NWChem User Manual

Release 7.3.1



W.R. Wiley Environmental Molecular Sciences Laboratory

Pacific Northwest National Laboratory

P.O. Box 999, Richland, WA 99352

Fri Nov 13 06:00:26 2026 13 2026

Educational Community License, Version 2.0 (ECL-2.0)

Table of Contents

1 NWChem: Open Source High-Performance Computational Chemistry

The NWChem software provides computational chemistry tools that can accurately tackle complex scientific problems. The software has been develop to efficiently use the aggregate resources available on supercomputers or computer clusters.

NWChem can handle:

- Biomolecules, nanostructures, and solid-state
- From quantum to classical, and all combinations
- Ground and excited-states
- Gaussian basis functions or plane-waves
- Scaling from one to thousands of processors
- Properties and relativistic effects

NWChem is actively developed by a consortium of developers and maintained by The Environmental Molecular Sciences Laboratory (EMSL) located at the Pacific Northwest National Laboratory (PNNL) in Washington State. Researchers interested in contributing to NWChem should review the Developers page. The code is distributed as open-source under the terms of the Educational Community License version 2.0 (ECL 2.0).

The NWChem development strategy is focused on providing new and essential scientific capabilities to its users in the areas of kinetics and dynamics of chemical transformations, chemistry at interfaces and in the condensed phase, and enabling innovative and integrated research at EMSL. At the same time continued development is needed to enable NWChem to effectively utilize architectures of tens of petaflops and beyond.

1.1 Latest NWChem release

NWChem version 7.3.1 is the latest release available for download from the link https://github.com/nwchemgit/nwchem/releases.

1.2 EMSL Arrows

Are you just learning how to use NWChem and would like to have an easy way to generate input decks, check your output decks against a large database of calculations, perform simple thermochemistry calculations, calculate the NMR and IR spectra of a modest size molecule, or just try out nwchem before installing it? EMSL Arrows scientific service can help. A web api to EMSL Arrows is now available for alpha testing.

For more information see EMSL Arrows - an easier way to use nwchem

EMSL Arrows API

1.3 NWChem Documentation

1.4 NWChem Citation

Please cite the following reference when publishing results obtained with NWChem:

E. Aprà, E. J. Bylaska, W. A. de Jong, N. Govind, K. Kowalski, T. P. Straatsma, M. Valiev, H. J. J. van Dam, Y. Alexeev, J. Anchell, V. Anisimov, F. W. Aquino, R. Atta-Fynn, J. Autschbach, N. P. Bauman, J. C. Becca, D. E. Bernholdt, K. Bhaskaran-Nair, S. Bogatko, P. Borowski, J. Boschen, J. Brabec, A. Bruner, E. Cauët, Y. Chen, G. N. Chuev, C. J. Cramer, J. Daily, M. J. O. Deegan, T. H. Dunning Jr., M. Dupuis, K. G. Dyall, G. I. Fann, S. A. Fischer, A. Fonari, H. Früchtl, L. Gagliardi, J. Garza, N. Gawande, S. Ghosh, K. Glaesemann, A. W. Götz, J. Hammond, V. Helms, E. D. Hermes, K. Hirao, S. Hirata, M. Jacquelin, L. Jensen, B. G. Johnson, H. Jónsson, R. A. Kendall, M. Klemm, R. Kobayashi, V. Konkov, S. Krishnamoorthy, M. Krishnan, Z. Lin, R. D. Lins, R. J. Littlefield, A. J. Logsdail, K. Lopata, W. Ma, A. V. Marenich, J. Martin del Campo, D. Mejia-Rodriguez, J. E. Moore, J. M. Mullin, T. Nakajima, D. R. Nascimento, J. A. Nichols, P. J.

Nichols, J. Nieplocha, A. Otero-de-la-Roza, B. Palmer, A. Panyala, T. Pirojsirikul, B. Peng, R. Peverati, J. Pittner, L. Pollack, R. M. Richard, P. Sadayappan, G. C. Schatz, W. A. Shelton, D. W. Silverstein, D. M. A. Smith, T. A. Soares, D. Song, M. Swart, H. L. Taylor, G. S. Thomas, V. Tipparaju, D. G. Truhlar, K. Tsemekhman, T. Van Voorhis, Á. Vázquez-Mayagoitia, P. Verma, O. Villa, A. Vishnu, K. D. Vogiatzis, D. Wang, J. H. Weare, M. J. Williamson, T. L. Windus, K. Woliński, A. T. Wong, Q. Wu, C. Yang, Q. Yu, M. Zacharias, Z. Zhang, Y. Zhao, and R. J. Harrison, “NWChem: Past, present, and future”, *The Journal of Chemical Physics* **152**, 184102 (2020). DOI: 10.1063/5.0004997

3 How to get NWChem

3.1 How to download and install NWChem

3.1.1 Source Download

The NWChem source is available for download from https://github.com/nwchemgit/nwchem/releases

Compilation instructions can be found at this link

3.1.2 NWChem availability in Linux distributions

Debian: https://packages.debian.org/search?keywords=nwchem

Ubuntu: https://launchpad.net/ubuntu/+source/nwchem

Fedora and EPEL: https://admin.fedoraproject.org/updates/search/nwchem

Good search engine for NWChem Linux packages: https://pkgs.org/search/?q=nwchem

3.1.2.1 Example of NWChem installation on Debian/Ubuntu

```
sudo apt-get  install nwchem
```

Parallel runs (using more than one process) can be performed with the command

```
/usr/bin/mpirun.openmpi -np 2 nwchem n2.nw
```

3.1.2.2 Example of NWChem RPM installation under Redhat/Centos 7 x86_64

```
sudo yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
sudo yum update
sudo yum install nwchem nwchem-openmpi Lmod
```

In order to run NWChem, you must type

```
module load mpi/openmpi-x86_64
```

The name of the NWChem executable is

```
nwchem_openmpi
```

Serial runs, using a single process, (on a input file named `n2.nw` in the following example) can be performed with the command

```
nwchem_openmpi n2.nw
```

Parallel runs (using more than one process) can be performed with the command

```
mpirun -np 2 nwchem_openmpi n2.nw
```

3.1.3 NWChem availability on macOS

NWChem can be installed from Homebrew, by executing the following commands

```
bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"

brew install nwchem
```

3.1.4 NWChem installation on Conda

NWChem can be installed on Linux or MacOS from the conda-forge channel of Conda with the commands

```
conda install -c conda-forge micromamba
micromamba install -c conda-forge nwchem
```

More details at https://github.com/conda-forge/nwchem-feedstock

3.2 Compiling NWChem from source

On this page, a step-by-step description of the build process and necessary and optional environment variables is outlined. In addition, based on the experiences of developers and users how-to's for various platforms have been created. These how-to's will be updated with additional platforms and better environment variables over time.

Download of the NWChem source is a step needed before compilation. Details for downloading as well as instructions for installing pre-compiled version of NWChem are available at the Download page.

3.2.1 Setting up the proper environment variables

- $NWCHEM_TOP defines the top directory of the NWChem source tree, e.g.

When dealing with source from a ***NWChem release*** (7.3.1 in this example)

```
export NWCHEM_TOP=<your path>/nwchem-7.3.1
```

- $NWCHEM_TARGET defines your target platform, e.g.

```
export NWCHEM_TARGET=LINUX64
```

The following platforms are available:

NWCHEM_TARGET	Platform	OS	Compilers
LINUX	x86	Linux	GNU, Intel
	ppc	Linux	GNU, IBM
	arm	Linux	GNU
LINUX64	x86_64	Linux	GNU, Intel, flang
	ppc64le	Linux	GNU, IBM
	aarch64	Linux	GNU, flang
MACX	x86	Darwin	GNU
MACX64	x86_64	Darwin	GNU
	arm64	Darwin	GNU

- $ARMCI_NETWORK must be defined in order to achieve best performance on high performance networks, e.g.

```
export ARMCI_NETWORK=MPI-PR
```

For a single processor system, this environment variable does not have to be defined. Supported combination of ARMCI_NETWORK and NWCHEM_TARGET variables:

ARMCI_NETWORK	NWCHEM_TARGET	Network	Protocol
OPENIB	LINUX, LINUX64	Mellanox InfiniBand	Verbs
MPI-PR	LINUX64	Any network	MPI
MPI-MT MPI-SPAWN	LINUX64	MPI supporting multi-threading multiple	MPI-2
MPI-TS MPI-PT	any	any network with MPI	MPI
BGMLMPI	BGL	IBM Blue Gene/L	BGLMPI
DC MFMPI	BGP	IBM Blue Gene/P	DCMF,MPI

Please see Choosing the ARMCI Library for additional information on choosing the right network options.

3.2.1.1 MPI variables

Variable	Description
USE_MPI	Set to “y” to indicate that NWChem should be compiled with MPI
USE_MPIF	Set to “y” for the NWPW module to use fortran-bindings of MPI. (Generally set when USE_MPI is set)
USE_MPIF4	Set to “y” for the NWPW module to use Integer*4 fortran-bindings of MPI. (Generally set when USE_MPI is set on most platforms)
LIBMPI (deprecated)	Name of the MPI library that should be linked with -l
MPI_LIB (deprecated)	Directory where the MPI library resides
MPI_INCLUDE (deprecated)	Directory where the MPI include files reside

3.2.1.1.1 Automatic detection of MPI variables with mpif90

New in NWChem 6.6: If the location of the mpif90 command is part of your PATH env. variable, NWChem will figure out the values of LIBMPI , MPI_LIB and MPI_INCLUDE (if they are not set). Therefore, we do**NOT** recommend to set LIBMPI , MPI_LIB and MPI_INCLUDE and add the location of mpif90 to the PATH variable, instead. Therefore, the next section can be considered obsolete.

3.2.1.1.2 OBSOLETE: How to set the MPI variables

The information of this section should not be used because of the automatic detection of MPI variables described in the previous section.

~~The output of the command~~

```
mpif90 -show
```

~~can be used to extract the values of LIBMPI, MPI_LIB and MPI_INCLUDE~~

~~E.g. for MPICH2, this might look like~~:

```
$ mpif90 -show
f95 -I/usr/local/mpich2.141p1/include -I/usr/local/mpich2.141p1/include -L/usr/local/mpich2.141p1/lib \
-lmpichf90 -lmpichf90 -lmpich -lopa -lmpl -lrt -lpthread
```

~~The corresponding environment variables are~~

```
% export USE_MPI=y
% export LIBMPI="-lmpich -lopa -lmpl -lpthread -lmpichf90 -lfmpich -lmpich"
% export MPI_LIB=/usr/local/mpich2.141p1/lib
% export MPI_INCLUDE='/usr/local/mpich2.141p1/include
```

3.2.1.1.3 How to start NWChem

When MPI is used, the appropriate MPI run command should be used to start an NWChem calculation, e.g.

```
% mpirun -np 8 $NWCHEM_TOP/bin/$NWCHEM_TARGET/nwchem h2o.nw
```

3.2.1.2 NWCHEM_MODULES

- $NWCHEM_MODULES defines the modules to be compiled, e.g.

```
export NWCHEM_MODULES="all python"
```

The following modules are available:

Module	Description
all	Everything useful
all python	Everything useful plus python
qm	All quantum mechanics modules
md	MD only build

Note that additional environment variables need to be defined to specify the location of the Python libraries, when the python module is compiled. See the optional environmental variables section for specifics.

3.2.2 Adding optional environmental variables

USE_NOFSCHECK can be set to avoid NWChem creating files for each process when testing the size of the scratch directory (a.k.a. creation of junk files), e.g.

```
export USE_NOFSCHECK=TRUE
```

USE_NOIO can be set to avoid NWChem 6.5 doing I/O for the ddscf, mp2 and ccsd modules (it automatically sets `USE_NOFSCHECK` , too). It is strongly recommended on large clusters or supercomputers or any computer lacking any fast and large local filesystem.

```
export USE_NOIO=TRUE
```

~~**LIB_DEFINES** can be set to pass additional defines to the C preprocessor (for both Fortran and C), e.g~~.

```
export LIB_DEFINES=-DDFLT_TOT_MEM=16777216
```

~~Note: `-DDFLT_TOT_MEM` sets the default dynamic memory available for NWChem to run, where the units are in doubles~~. However, it is recommended that, instead of manually defining this environment variable, the getmem.nwchem script to be executed as described in the related section

MRCC_METHODS can be set to request the multireference coupled cluster capability to be included in the code, e.g.

```
export MRCC_METHODS=TRUE
```

CCSDTQ can be set to request the CCSDTQ method and its derivatives to be included in the code, e.g.

```
export CCSDTQ=TRUE
```

3.2.2.1 Setting Python environment variables

Python programs may be embedded into the NWChem input and used to control the execution of NWChem. To build with Python, Python needs to be available on your machine. The software can be download from https://www.python.org . Follow the Python instructions for installation and testing. NWChem has been tested with Python versions up to 3.11

The following environment variables need to be set when compiling with Python, together with having the location of your installed python binary part of the `PATH` environment variable:

```
export PYTHONVERSION=3.8
```

Note that the third number in the version should not be kept: 3.8.1 should be set as 3.8

You will also need to set PYTHONPATH to include any modules that you are using in your input. Examples of Python within NWChem are in the `$NWCHEM_TOP/QA/tests/pyqa3` and `$NWCHEM_TOP/contrib/python` directories.

3.2.2.2 Optimized math libraries

By default NWChem uses its own basic linear algebra subroutines (BLAS). To include faster BLAS routines, the environment variable BLASOPT needs to be set before building the code. For example, with OpenBLAS

```
export BLASOPT="-lopenblas"
```

Good choices of optimized BLAS libraries on x86 (e.g. LINUX and LINUX64) hardware include:

BLIS	https://github.com/flame/blis
OpenBLAS	https://github.com/xianyi/OpenBLAS
Intel MKL	https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html
Cray LibSci	Available only on Cray hardware, it is automatically linked when compiling on Cray computers.
IBM ESSL	Available only on IBM hardware https://www.ibm.com/docs/en/essl/6.3

New since release 7.0.0 (after commit 6b0a971): If `BLASOPT` is defined, the `LAPACK_LIB` environment variable must be set up, too. `LAPACK_LIB` must provide the location of the library containing the LAPACK routines. For example, OpenBLAS provides the full suite of LAPACK routines, therefore, in this case, `LAPACK_LIB` can be set to the same value as BLASOPT

```
export BLASOPT=-lopenblas
export LAPACK_LIB=-lopenblas
```

NWChem can also take advantage of theScaLAPACK library if it is installed on your system. The following environment variables need to be set:

```
export USE_SCALAPACK=y
export SCALAPACK="location of Scalapack and BLACS library"
```

3.2.2.3 How to deal with integer size of Linear Algebra libraries

In the case of 64-bit platforms, most vendors optimized BLAS libraries cannot be used. This is due to the fact that while NWChem uses 64-bit integers (i.e. integer*8) on 64-bit platforms, most of the vendors optimized BLAS libraries used 32-bit integers. The same holds for the ScaLAPACK libraries, which internally use 32-bit integers.
The BLAS_SIZE environment variable is used at compile time to set the size of integer arguments in BLAS calls.

BLAS_SIZE	**size of integer arguments in BLAS routines**
4	32-bit (most common default)
8	64-bit

A method is available to link against the libraries mentioned above, using the following procedure:

```
cd $NWCHEM_TOP/src
make clean
make 64_to_32
make USE_64TO32=y  BLAS_SIZE=4 BLASOPT=" optimized BLAS" SCALAPACK="location of Scalapack and BLACS library"
```

E.g., for IBM64 this looks like

```
% make USE_64TO32=y BLAS_SIZE=4 BLASOPT="-lessl -lmass"
```

Notes:

- OpenBLAS can be installed with 64bit integers. This is accomplished by compiling the OpenBLAS library after having edited the OpenBLAS Makefile.rule file and un-commenting the line containing the INTERFACE64 definition. In other words, the line

```
#INTERFACE64 = 1
```

needs to be changed to

```
INTERFACE64 = 1
```

- AOCL and MKL can support 64-bit integers if the appropriate library is chosen. For MKL, one can choose the ILP64 Version of Intel® MKL and the correct recipe can be extracted from the website https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl-link-line-advisor.html.

3.2.2.4 Automated build of OpenBLAS/ScaLAPACK

New in NWChem 7.0.2:

1. The environment variable `BUILD_OPENBLAS` can be used to automatically build the OpenBLAS library during a NWChem compilation (either using `BLAS_SIZE=8` or `BLAS_SIZE=4`)
2. The environment variable `BUILD_SCALAPACK` can be used to automatically build the ScaLapack library during a NWChem compilation (either using `SCALAPACK_SIZE=8` or `SCALAPACK_SIZE=4`)

The following settings are strongly recommended over setting variables pointing to existing installations:

```
BUILD_OPENBLAS=1
BUILD_SCALAPACK=1
BLAS_SIZE=8
SCALAPACK_SIZE=8
```

3.2.2.5 Linking in NBO

The 5.0 (obsolete) version of NBO provides a utility to generate source code that can be linked into computational chemistry packages such as NWChem. To utilize this functionality, follow the instructions in the NBO 5 package to generate an nwnbo.f file. Linking NBO into NWChem can be done using the following procedure:

```
% cd $NWCHEM_TOP/src
% cp nwnbo.f $NWCHEM_TOP/src/nbo/.
% make nwchem_config NWCHEM_MODULES="all nbo"
% make
```

One can now use “task nbo” and incorporate NBO input into the NWChem input file directly:

```
nbo
  $NBO NRT $END
  ...
end

task nbo
```

3.2.3 Building the NWChem binary

Once all required and optional environment variables have been set, NWChem can be compiled:

```
% cd $NWCHEM_TOP/src

% make nwchem_config

% make >& make.log
```

The make above will use the standard compilers available on your system. To use compilers different from the default one can either set environment variables:

```
% export FC=<fortran compiler>
% export CC=<c compiler>
```

Or one can supply the compiler options to the make command (*ecommended* option), e.g:

```
% make FC=ifort
```

For example, on Linux FC could be set either equal to ifort, gfortran or pgf90

Nota bene: NWChem does NOT support usage of the full path in FC and CC variables. Please provide filenames only as in the examples above!

Note 1: If in a Linux environment, FC is set equal to anything other than the tested compilers, there is no guarantee of a successful installation, since the makefile structure has not been tested to process other settings. In other words, please avoid make FC="ifort -O3 -xhost" and stick to make FC="ifort", instead

Note 2: It's better to avoid redefining CC, since a) NWChem does not have C source that is a computational bottleneck and b) we typically test just the default C compiler. In other words, the recommendation is to compile with make FC=ifort

Note 3: It's better to avoid modifying the values of the FOPTIMIZE and COPTIMIZE variables. The reason is that the default values for FOPTIMIZE and COPTIMIZE have been tested by the NWChem developers (using the internal QA suites, among others), while any modification might produce incorrect results.

3.2.3.1 Setting the default memory values

It is strongly recommended to use, after a successful compilation, thegetmem.nwchem script in the $NWCHEM_TOP/contrib directory. The script will choose the default memory settings based on the available physical memory, recompile the appropriate files and relink. Here is an example of its usage:

```
cd $NWCHEM_TOP/src
../contrib/getmem.nwchem
```

If non default compiler are used, the getmem.nwchem script must be called, using bash shell, by first specifying the compiler environment variable. The example below uses ifort as Fortran compiler

```
cd $NWCHEM_TOP/src
FC=ifort ../contrib/getmem.nwchem
```

3.2.4 How-to: Linux platforms

- **Common environmental variables for building in serial or in parallel with MPI**

```
% export NWCHEM_TOP=<your path>/nwchem
% export NWCHEM_TARGET=LINUX64
% export NWCHEM_MODULES =all
```

- **Common environmental variables for building with MPI**

The following environment variables need to be set when NWChem is compiled with MPI:

```
% export USE_MPI=y
% export USE_MPIF=y
% export USE_MPIF4=y
```

New in NWChem 6.6: If the location of the mpif90 command is part of your PATH env. variable, NWChem will figure out the values of LIBMPI, MPI_LIB and MPI_INCLUDE (if they are not set). Therefore, we recommend not to set LIBMPI, MPI_LIB and MPI_INCLUDE and add the location of mpif90 to the PATH variable, instead.

- **Compiling the code once all variables are set**

```
% cd $NWCHEM_TOP/src
% make nwchem_config
% make FC=gfortran >& make.log
```

3.2.4.1 NWChem 6.8 on Centos 7.1/Fedora 27

Once you have added the EPEL repository to your Centos/Fedora/RedHat installation, you can have a more efficient NWChem build.

```
sudo rpm -Uvh http://download.fedoraproject.org/pub/epel/7/x86_64/Packages/e/epel-release-7-11.noarch.rpm
```

- Packages required:

```
python-devel gcc-gfortran openblas-devel openblas-serial64 openmpi-devel scalapack-openmpi-devel \
elpa-openmpi-devel tcsh openssh-clients which tar bzip2
```

- Settings

```
export USE_MPI=y
export NWCHEM_TARGET=LINUX64
export USE_PYTHONCONFIG=y
export PYTHONVERSION=2.7
export PYTHONHOME=/usr
export USE_64TO32=y
export BLAS_SIZE=4
export BLASOPT="-lopenblas -lpthread -lrt"
export LAPACK_LIB=$BLASOPT
export SCALAPACK_SIZE=4
export SCALAPACK="-L/usr/lib64/openmpi/lib -lscalapack"
export ELPA="-I/usr/lib64/gfortran/modules/openmpi -L/usr/lib64/openmpi/lib -lelpa"
export LD_LIBRARY_PATH=/usr/lib64/openmpi/lib/:$LD_LIBRARY_PATH
export PATH=/usr/lib64/openmpi/bin/:$PATH
```

- Compilation steps

```
cd $NWCHEM_TOP/src
make nwchem_config NWCHEM_MODULES="all python"
make 64_to_32
make
```

3.2.5 How to: Mac platforms

3.2.5.1 Compilation of NWChem 6.6 on Mac OS X 10.10 (Yosemite) x86_64

3.2.5.1.1 Method #1: using gfortran and openmpi from brew

- Download and unpack latest NWChem tarball to the directory of your choosing, say /Users/johndoe/nwchem
- Install Homebrew as described at http://brew.sh (more details at https://docs.brew.sh/Installation.html)

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

- Use Homebrew to install open-mpi

```
brew install open-mpi
```

- As usual, set the env. variables

```
export USE_MPI=y
export NWCHEM_TARGET=MACX64
export NWCHEM_TOP=/Users/johndoe/nwchem
export USE_INTERNALBLAS=y
```

- **Important**: set the following env. variable (GA will not compile otherwise)

```
export CFLAGS_FORGA "-DMPICH_NO_ATTR_TYPE_TAGS"
```

- Go to your source directory, configure, and compile

```
cd /Users/johndoe/nwchem/src
make nwchem_config`
make
```

WARNING: Please do not use the Mac OS X default BLAS and LAPACK libraries available (or brew's veclibfort), since they are causing NWChem to produce erroneous results

3.2.5.1.2 Method #2: using Intel compilers and MKL

The Intel compilers and MKL work just fine on Mac with the following options:

```
NWCHEM_TARGET=MACX64
CC=icc
FC=ifort
BLASOPT="-mkl -openmp"
USE_OPENMP=T
```

MPICH and ARMCI-MPI work reliably on Mac. See Choosing the ARMCI Library for details on ARMCI-MPI

3.2.6 How-to: Cray platforms

Common environmental variables for building and running on the Cray XT, XE, XC and XK:

```
% export NWCHEM_TOP=<your path>/nwchem
% export NWCHEM_TARGET=LINUX64
% export NWCHEM_MODULES=all
% export USE_MPI=y
% export USE_MPIF=y
% export USE_MPIF4=y
% export USE_SCALAPACK=y
% export USE_64TO32=y
% export LIBMPI=" "
```

- **Compiling the code on Cray once all variables (described below) are set**

```
% cd $NWCHEM_TOP/src
% make nwchem_config
% make 64_to_32
% make FC=ftn >& make.log
```

The step `make 64_to_32` is required only if either SCALAPACK_SIZE or BLAS_SIZE are set equal to 4.

3.2.6.1 Method #2: ARMCI_NETWORK=MPI-PR

This is a **new option available in NWChem 6.6.**

Set the environmental variables for compilation:

```
% export ARMCI_NETWORK=MPI-PR
```

3.2.6.2 Example: OLCF Titan

These are variables used for compilation on the OLCF Titan, a Cray XK7 We assume use of Portland Group compilers programming environment (`module load PrgEnv-pgi`)

```
NWCHEM_TARGET=LINUX64
ARMCI_NETWORK=MPI-PR
USE_64TO32=y
USE_MPI=y
BLAS_SIZE=4
LAPACK_SIZE=4
SCALAPACK_SIZE=4
SCALAPACK=-lsci_pgi_mp
BLASOPT=-lsci_pgi_mp
```

To enable the GPU part, set

```
TCE_CUDA=y
```

and load the cudatoolkit module

```
module load cudatoolkit
```

3.2.6.3 Aries, e.g. XC30/XC40

3.2.6.3.1 Method #1: ARMCI_NETWORK=MPI-PR

This is a **new option available in NWChem 6.6.**

Set the environmental variables for compilation:

```
% export ARMCI_NETWORK=MPI-PR
```

3.2.6.3.2 Example: HPE Cray

These are variables used for compilation a Cray HPE system using Intel compilers (i.e. after issuing the commands `module swap PrgEnv-gnu PrgEnv-intel`).

```
export NWCHEM_TARGET=LINUX64
export USE_MPI=y
export NWCHEM_TARGET=LINUX64
export ARMCI_NETWORK=MPI-PR
export USE_MPI=y
export USE_SCALAPACK=y
export SCALAPACK="-L$MKLROOT/lib/intel64 -lmkl_scalapack_ilp64 -lmkl_intel_ilp64 -lmkl_core -lmkl_sequential \
-lmkl_blacs_intelmpi_ilp64 -lpthread -lm"
export SCALAPACK_SIZE=8
export SCALAPACK_LIB="$SCALAPACK"
export BLAS_SIZE=8
export BLASOPT="-L$MKLROOT/lib/intel64 -lmkl_intel_ilp64 -lmkl_core -lmkl_sequential -lmkl_core -liomp5 -lpthread -ldmapp -lm"
export USE_NOIO=y
export CRAYPE_LINK_TYPE=dynamic
```

To compile

```
make nwchem_config
make FC=ftn
```

3.2.7 How-to: Intel Xeon Phi

This section describes both the newer KNL and older KNC hardware, in reverse chronological order.

- **Compiling NWChem on self-hosted Intel Xeon Phi Knights Landing processors**

NWChem 6.6 (and later versions) support OpenMP threading, which is essential to obtaining good performance with NWChem on Intel Xeon Phi many-core processors.
As of November 2016, the development version of NWChem contains threading support in the TCE coupled-cluster codes (primarily non-iterative triples in e.g. CCSD(T)), semi-direct CCSD(T), and plane-wave DFT (i.e. NWPW).

Required for compilation: Intel compilers, version 16+ (17+ is strongly recommended).

Environmental variables required for compilation:

```
% export USE_KNL=1
% export USE_OPENMP=1
% export USE_F90_ALLOCATABLE=T
% export USE_FASTMEM=T
```

The latter two options are required to allocate temporaries in MCDRAM when running in flat mode. Please do not use cache mode for NWChem CCSD(T) codes. Note that using Fortran heap allocations means the memory statistics generated by MA are no longer accurate, but we doubt that anyone has been relying on these anyways.

`USE_FASTMEM` requires the *memkind* library to be installed. An open source version of the memkind library can be downloaded from Github

Side note: With the exception of `USE_FASTMEM`, all of the options in the KNL section apply to Intel Xeon processors as well. OpenMP is certainly useful on multicore processors as a way to reduce the communication overhead and memory footprint of NWChem.

When using MKL and Intel 16+, please use the following settings

```
% export BLASOPT  ="-mkl -qopenmp"
% export SCALAPACK="-mkl -qopenmp -lmkl_scalapack_ilp64 -lmkl_blacs_intelmpi_ilp64"
```

The command require for compilation is

```
$ make FC=ifort CC=icc
```

Environmental variables recommended at runtime (assuming Intel OpenMP and MPI):

```
% export I_MPI_PIN=1
% export I_MPI_DEBUG=4
% export KMP_BLOCKTIME=1
% export KMP_AFFINITY=scatter,verbose
```

Once you are comfortable with the affinity settings, you can use these instead:

```
% export I_MPI_PIN=1
% export KMP_BLOCKTIME=1
% export KMP_AFFINITY=scatter
```

Please consult the Intel or similar documentation regarding MPI+OpenMP affinity on your system. This is a complicated issue that depends on the software you use; it is impossible to document all the different combinations of MPI and OpenMP implementations that may be used with NWChem.

If you encounter segfaults not related to ARMCI, you may need to set the following or recompile with `-heap-arrays`. Please

create thread in the Forum if you observe this.

```
% ulimit -s unlimited
% export OMP_STACKSIZE=32M
```

- **Compiling NWChem on hosts equipped with Intel Xeon Phi Knights Corner coprocessors**

NWChem 6.5 (and later versions) offers the possibility of using Intel Xeon Phi hardware to perform the most computationally intensive part of the CCSD(T) calculations (non-iterative triples corrections).

Required for compilation: Intel Composer XE version 14.0.3 (or later versions)

Environmental variables required for compilation:

```
% export USE_OPENMP=1
% export USE_OFFLOAD=1
```

When using MKL and Intel Composer XE version 14 (or later versions), please use the following settings

```
% export BLASOPT  ="-mkl -openmp   -lpthread -lm"
% export SCALAPACK="-mkl -openmp -lmkl_scalapack_ilp64 -lmkl_blacs_intelmpi_ilp64 -lpthread -lm"
```

The command require for compilation is

```
$ make FC=ifort
```

- **Examples of recommended configurations**

From our experience using the CCSD(T) TCE module, we have determined that the optimal configuration is to use a single Global Arrays ranks for offloading work to each Xeon Phi card.

On the EMSL cascade system, each node is equipped with two coprocessors, and NWChem can allocate one GA ranks per coprocessor. In the job scripts, we recommend spawning just 6 GA ranks for each node, instead of 16 (number that would match the number of physical cores). Therefore, 2 out 6 GA ranks assigned to a particular compute node will offload to the coprocessors, while the remaining 6 cores while be used for traditional CPU processing duties. Since during offload the host core is idle, we can double the number of OpenMP threads for the host (`OMP_NUM_THREADS=4`) in order to fill the idle core with work from another GA rank (4 process with 4 threads each will total 16 threads on each node).

NWChem itself automatically detects the available coprocessors in the system and properly partitions them for optimal use, therefore no action is required other than specifying the number of processes on each node (using the appropriate mpirun/mpiexec options) and setting the value of `OMP_NUM_THREADS` as in the example above.

Environmental variables useful at run-time:

OMP_NUM_THREADS is needed for the thread-level parallelization on the Xeon CPU hosts

```
% export OMP_NUM_THREADS=4
```

MIC_USE_2MB_BUFFER greatly improve communication between host and Xeon Phi card

```
% export MIC_USE_2MB_BUFFER=16K
```

Very important: when running on clusters equipped with Xeon Phi and Infiniband network hardware (requiring `ARMCI_NETWORK=OPENIB`), the following env. variable is required, even in the case when the Xeon Phi hardware is not utilized.

```
% export ARMCI_OPENIB_DEVICE=mlx4_0
```

3.2.8 How-to: IBM platforms

- **Compiling NWChem on BLUEGENE/L**

The following environment variables need to be set

```
% export NWCHEM_TOP=<your path>/nwchem
% export NWCHEM_TARGET=BGL
% export ARMCI_NETWORK=BGMLMPI
% export BGLSYS_DRIVER=/bgl/BlueLight/ppcfloor
% export BGLSYS_ROOT=${BGLSYS_DRIVER}/bglsys
% export BLRTS_GNU_ROOT=${BGLSYS_DRIVER}/blrts-gnu
% export BGDRIVER=${BGLSYS_DRIVER}
% export BGCOMPILERS=${BLRTS_GNU_ROOT}/bin
% export USE_MPI=y
% export LARGE_FILES=TRUE
% export MPI_LIB=${BGLSYS_ROOT}/lib
% export MPI_INCLUDE=${BGLSYS_ROOT}/include
% export LIBMPI="-lfmpich_.rts -lmpich.rts -lmsglayer.rts -lrts.rts -ldevices.rts"
% export BGMLMPI_INCLUDE=/bgl/BlueLight/ppcfloor/bglsys/include
% export BGMLLIBS=/bgl/BlueLight/ppcfloor/bglsys/lib
```

To compile, the following commands should be used:

```
% cd $NWCHEM_TOP/src
% make nwchem_config
% make FC=blrts_xlf >& make.log
```

- **Compiling NWChem on BLUEGENE/P**

The following environment variables need to be set

```
% export NWCHEM_TARGET=BGP
% export ARMCI_NETWORK=DCMFMPI
% export MSG_COMMS=DCMFMPI
% export USE_MPI=y
% export LARGE_FILES=TRUE
% export BGP_INSTALLDIR=/bgsys/drivers/ppcfloor
% export BGCOMPILERS=/bgsys/drivers/ppcfloor/gnu-linux/bin
% export BGP_RUNTIMEPATH=/bgsys/drivers/ppcfloor/runtime
% export ARMCIDRV=${BGP_INSTALLDIR}
% export BGDRIVER=${ARMCIDRV}
% export MPI_LIB=${BGDRIVER}/comm/lib
% export MPI_INCLUDE=${BGDRIVER}/comm/include
% export LIBMPI="-L${MPI_LIB} -lfmpich_.cnk -lmpich.cnk -ldcmfcoll.cnk -ldcmf.cnk -lpthread -lrt -L${BGP_RUNTIMEPATH}/SPI -lSPI.cna"
% export BGMLMPI_INCLUDE=${MPI_INCLUDE}
```

To compile, the following commands should be used:

```
% cd $NWCHEM_TOP/src
% make nwchem_config
% make FC=bgxlf >& make.log
```

- **Compiling NWChem on BLUEGENE/Q**

The following environment variables need to be set

```
% export NWCHEM_TARGET=BGQ
% export USE_MPI=y
% export USE_MPIF=y
% export USE_MPIF4=y
% export MPI_INCLUDE=/bgsys/drivers/ppcfloor/comm/xl/include
% export LIBMPI=" "
% export BLASOPT="/opt/ibmmath/essl/5.1/lib64/libesslbg.a -llapack -lblas -Wl,-zmuldefs "
% export BLAS_LIB="/opt/ibmmath/essl/5.1/lib64/libesslbg.a -zmuldefs "
% export BLAS_SIZE=4
% export USE_64TO32=y
% set path=(/bgsys/drivers/ppcfloor/gnu-linux/bin/ $path)
% export ARMCI_NETWORK=MPI-TS
% export DISABLE_GAMIRROR=y
```

To compile, the following commands should be used:

```
% module load bgq-xl
% make nwchem_config
% make 64_to_32 >& make6t3.log
% make >& make.log
```

WARNING: This is just a baseline port that we have tested and validated against our QA suite. There is large room for improvement both for serial performance (compiler options) and parallel performance (use of alternative ARMCI_NETWORKs other than MPI-TS)

- **Compiling NWChem on IBM PowerPC architectures**

The following environment variables should be set:

```
% export NWCHEM_TOP=<your path>/nwchem
% export NWCHEM_TARGET=IBM64
% export ARMCI_NETWORK=MPI-MT
% export OBJECT_MODE=64
% export USE_MPI=y
```

To compile, the following commands should be used:

```
% cd $NWCHEM_TOP/src
% make nwchem_config
% make FC=xlf >& make.log
```

3.2.9 How-to: Commodity clusters with Infiniband

Common environmental variables for building and running on most Infiniband clusters are:

```
export NWCHEM_TOP=<your path>/nwchem
export NWCHEM_TARGET=LINUX64
export NWCHEM_MODULES="all"
export USE_MPI=y
export USE_MPIF=y
export USE_MPIF4=y
```

- On Infiniband clusters with the OpenIB software stack, the following environment variables should be defined

```
export ARMCI_NETWORK=OPENIB
export IB_INCLUDE=<Location of Infiniband libraries>/include
```

- Compiling the code on an Infiniband cluster once all variables are set

```
cd $NWCHEM_TOP/src

make nwchem_config

make >& make.log
```

3.2.10 How-to: Commodity clusters with Intel Omni-Path

- On clusters with the Intel Omni-Path network, the following environment variables should be defined

```
export ARMCI_NETWORK=MPI-PR
```

The following setting is needed to avoid run-time errors

```
export PSM2_MEMORY=large
```

More details on this topic discussed a

- https://github.com/nwchemgit/nwchem/issues/284

- https://github.com/GlobalArrays/ga/issues/126

3.2.11 How-to: Windows Platforms

3.2.11.1 MingW

The current recommended approach for building a NWChem binary for a Windows platform is to build with the MinGW/Mingw32 environment. MinGW can be installed using a semi-automatic tool mingw-get-setup.exe (http://sourceforge.net/projects/mingw/files/Installer/). A basic MinGW installation is required (Basic Setup), plus pthreads-32, mingw32-gcc-fortran-dev of “All Packages” and the MSYS software.
More detailed MinGW/MSYS installation tips can be found in the following forum discussions

https://nwchemgit.github.io/Special_AWCforum/sp/id5124.html

https://nwchemgit.github.io/Special_AWCforum/sp/id6628.html

Another essential prerequisite step is to install Mpich, which can be found at the following URL

https://web.archive.org/web/20190105171104/http://www.mpich.org/static/tarballs/1.4.1p1/mpich2-1.4.1p1-win-ia32.msi

Once Mpich is installed, you should copy the installation files to a different location to avoid the failure of the tools compilation. You can use the following command

```
% cp -rp /c/Program\ Files\ \(x86\)/MPICH2/ ~/
```

You might want to install Python, too, by using the following installation file

https://www.python.org/ftp/python/2.7.8/python-2.7.8.msi

Next, you need to set the env.

```
% export NWCHEM_TOP=~/nwchem-6.8
% export NWCHEM_TARGET=LINUX
% export USE_MPI=y
% export MPI_LOC=~/MPICH2
% export MPI_INCLUDE=$MPI_LOC/include
% export MPI_LIB=$MPI_LOC/lib
% export LIBMPI="-lfmpich2g -lmpi"
% export PYTHONVERSION=27
% export DEPEND_CC=gcc
% export USE_INTERNALBLAS=y
% export NWCHEM_MODULES=all
```

Then, you can start the compilation by typing

```
% cd $NWCHEM_TOP/src
% make nwchem_config
% make FC=gfortran DEPEND_CC=gcc
```

3.2.11.1.1 MSYS2

https://github.com/msys2/msys2/wiki/MSYS2-installation

```
pacman -Syuu
pacman -S mingw32/mingw-w64-i686-gcc-fortran
pacman -S mingw32/mingw-w64-i686-python3
pacman -S msys/make
```

3.2.11.2 WSL on Windows 10

A good alternative only on Windows 10 is **Windows Subsystem for Linux** (WSL). This option gives the best performance on Windows when WLS 2 is used. **WSL** allows you to obtain a functional command line Linux 64-bit NWChem environment, either by compiling the NWChem code from scratch or by using the Ubuntu precompiled NWChem package. Here is a link to the install guide

https://learn.microsoft.com/en-us/windows/wsl/install

Once Ubuntu is installed, the quickest method to install NWChem is by fetching the Ubuntu NWChem package by typing

```
sudo apt install nwchem
```

3.2.12 General site installation

The build procedures outlined above will allow use of NWChem within the NWChem directory structure. The code will look for the basis set library file in a default place within that directory structure. To install the code in a general, public place (e.g., /usr/local/NWChem) the following procedure can be applied:

- Determine the local storage path for the install files. (e.g., /usr/local/NWChem).
- Make directories

```
mkdir /usr/local/NWChem
mkdir /usr/local/NWChem/bin
mkdir /usr/local/NWChem/data
```

- Copy binary

```
cp $NWCHEM_TOP/bin/${NWCHEM_TARGET}/nwchem /usr/local/NWChem/bin
cd /usr/local/NWChem/bin
chmod 755 nwchem
```

- Set links to data files (basis sets, force fields, etc.)

```
cd $NWCHEM_TOP/src/basis
cp -r libraries /usr/local/NWChem/data
cd $NWCHEM_TOP/src/
cp -r data /usr/local/NWChem
cd $NWCHEM_TOP/src/nwpw
cp -r libraryps /usr/local/NWChem/data
```

- Each user will need a .nwchemrc file to point to these default data files. A global one could be put in /usr/local/NWChem/data and a symbolic link made in each users `$HOME` directory is probably the best plan for new installations. Users would have to issue the following command prior to using NWChem:

```
ln -s /usr/local/NWChem/data/default.nwchemrc $HOME/.nwchemrc
```

Contents of the `default.nwchemrc` file based on the above information should be:

```
nwchem_basis_library /usr/local/NWChem/data/libraries/
nwchem_nwpw_library /usr/local/NWChem/data/libraryps/
ffield amber
amber_1 /usr/local/NWChem/data/amber_s/
amber_2 /usr/local/NWChem/data/amber_q/
amber_3 /usr/local/NWChem/data/amber_x/
amber_4 /usr/local/NWChem/data/amber_u/
spce    /usr/local/NWChem/data/solvents/spce.rst
charmm_s /usr/local/NWChem/data/charmm_s/
charmm_x /usr/local/NWChem/data/charmm_x/
```

Of course users can copy this file instead of making the symbolic link described above and change these defaults at their discretion.

It is can also be useful to use the `NWCHEM_BASIS_LIBRARY` environment variable when testing a new installation when an old one exists. This will allow you to overwrite the value of `nwchem_basis_library` in your `.nwchemrc` file and point to the new basis

library. For example:

```
% export NWCHEM_BASIS_LIBRARY="$NWCHEM/data-5.0/libraries/"
```

Do not forget the trailing “/”.

4 Introduction

4.1 Getting Started

This section provides an overview of NWChem input and program architecture, and the syntax used to describe the input. See Simple Input File and Water Molecule Input for examples of NWChem input files with detailed explanation.

NWChem consists of independent modules that perform the various functions of the code. Examples of modules include the input parser, SCF energy, SCF analytic gradient, DFT energy, etc.. Data is passed between modules and saved for restart using a disk-resident database or dumpfile (see NWChem Architecture).

The input to NWChem is composed of commands, called directives, which define data (such as basis sets, geometries, and filenames) and the actions to be performed on that data. Directives are processed in the order presented in the input file, with the exception of certain start-up directives (see Input File Structure) which provide critical job control information, and are processed before all other input. Most directives are specific to a particular module and define data that is used by that module only. A few directives (see Top-level Directives) potentially affect all modules, for instance by specifying the total electric charge on the system.

There are two types of directives. Simple directives consist of one line of input, which may contain multiple fields. Compound directives group together multiple simple directives that are in some way related and are terminated with an END directive. See the sample inputs (Simple Input File and Water Molecule Input) and the input syntax specification (Input Format and Syntax for Directives).

All input is free format and case is ignored except for actual data (e.g., names/tags of centers, titles). Directives or blocks of module-specific directives (i.e., compound directives) can appear in any order, with the exception of the TASK directive (see Input File Structure and Tasks) which is used to invoke an NWChem module. All input for a given task must precede the TASK directive. This input specification rule allows the concatenation of multiple tasks in a single NWChem input file.

To make the input as short and simple as possible, most options have default values. The user needs to supply input only for those items that have no defaults, or for items that must be different from the defaults for the particular application. In the discussion of each directive, the defaults are noted, where applicable.

The input file structure is described in the following sections, and illustrated with two examples. The input format and syntax for directives is also described in detail.

4.1.1 .nwchemrc for environment variables and libraries

Each user should have a `.nwchemrc` file to point to default data files, such as basis sets, pseudopotentials, and MD potentials.

Contents of the default.nwchemrc file based on the above information should be:

```
nwchem_basis_library <location of NWChem installation>/src/basis/libraries/
nwchem_nwpw_library <location of NWChem installation>/src/nwpw/libraryps/
ffield amber
amber_1 <location of NWChem installation>/src/data/amber_s/
amber_2 <location of NWChem installation>/src/data/amber_q/
amber_3 <location of NWChem installation>/src/data/amber_x/
amber_4 <location of NWChem installation>/src/data/amber_u/
spce   <location of NWChem installation>/src/data/solvents/spce.rst
charmm_s <location of NWChem installation>/src/data/charmm_s/
charmm_x <location of NWChem installation>/src/data/charmm_x/
```

It is can also be useful to use the NWCHEM_BASIS_LIBRARY environment variable when testing a new libraries in your own directory. This will allow you to overwrite the value of nwchem_basis_library in your .nwchemrc file and point to the new basis library. For example:

```
% setenv NWCHEM_BASIS_LIBRARY "$NWCHEM/data-5.0/libraries/"
```

Do not forget the trailing "/" .

4.1.2 Input File Structure

The structure of an input file reflects the internal structure of NWChem. At the beginning of a calculation, NWChem needs to determine how much memory to use, the name of the database, whether it is a new or restarted job, where to put scratch/permanent files, etc.. It is not necessary to put this information at the top of the input file, however. NWChem will read through the entire input file looking for the start-up directives. In this first pass, all other directives are ignored.

The start-up directives are

```
START
RESTART
SCRATCH_DIR
PERMANENT_DIR
MEMORY
ECHO
```

After the input file has been scanned for the start-up directives, it is rewound and read sequentially. Input is processed either by the top-level parser (for the directives listed in Top-level Directives, such as TITLE, SET, …) or by the parsers for specific computational modules (e.g., SCF, DFT, …). Any directives that have already been processed (e.g., MEMORY) are ignored. Input is read until a TASK directive (see Tasks) is encountered. A TASK directive requests that a calculation be performed and specifies the level of theory and the operation to be performed. Input processing then stops and the specified task is executed. The position of the TASK directive in effect marks the end of the input for that task. Processing of the input resumes upon the successful completion of the task, and the results of that task are available to subsequent tasks in the same input file.

The name of the input file is usually provided as an argument to the execute command for NWChem. That is, the execute command looks something like the following

```
nwchem input_file
```

The default name for the input file is `nwchem.nw` . If an input file name `input_file` is specified without an extension, the code assumes .nw as a default extension, and the input filename becomes `input_file.nw` . If the code cannot locate a file named either `input_file` or `input_file.nw` (or nwchem.nw if no file name is provided), an error is reported and execution terminates. The following section presents two input files to illustrate the directive syntax and input file format for NWChem applications.

4.1.3 Simple Input File – SCF geometry optimization

A simple example of an NWChem input file is an SCF geometry optimization of the nitrogen molecule, using a Dunning cc-pvdz basis set. This input file contains the bare minimum of information the user must specify to run this type of problem – fewer than ten lines of input, as follows:

```
title "Nitrogen cc-pvdz SCF geometry optimization"
geometry
  n 0 0 0
  n 0 0 1.08
end
basis
  n library cc-pvdz
end
task scf optimize
```

Examining the input line by line, it can be seen that it contains only four directives; TITLE, GEOMETRY, BASIS, and TASK. The TITLE directive is optional, and is provided as a means for the user to more easily identify outputs from different jobs. An initial geometry is specified in Cartesian coordinates and Angstrøms by means of the GEOMETRY directive. The Dunning cc-pvdz basis is obtained from the NWChem basis library, as specified by the BASIS directive input. The TASK directive requests an SCF geometry optimization.

The GEOMETRY directive defaults to Cartesian coordinates and Angstrøms (options include atomic units and Z-matrix format). The input blocks for the BASIS and GEOMETRY directives are structured in similar fashion, i.e., name, keyword, …, end (In this simple example, there are no keywords). The BASIS input block must contain basis set information for every atom type in the geometry with which it will be used. Refer to Basis for a description of available basis sets and a discussion of how to define new ones.

The last line of this sample input file (task scf optimize) tells the program to optimize the molecular geometry by minimizing the SCF energy. (For a description of possible tasks and the format of the TASK directive, refer to Tasks)

If the input is stored in the file `n2.nw` , the command to run this job on a typical UNIX workstation is as follows:

```
nwchem n2
```

NWChem output is to UNIX standard output, and error messages are sent to both standard output and standard error.

4.1.4 Water Molecule Sample Input File

A more complex sample problem is the optimization of a positively charged water molecule using second-order Møller-Plesset perturbation theory (MP2), followed by a computation of frequencies at the optimized geometry. A preliminary SCF geometry optimization is performed using a computationally inexpensive basis set (STO-3G). This yields a good starting guess for the optimal geometry, and any Hessian information generated will be used in the next optimization step. Then the optimization is finished using MP2 and a basis set with polarization functions. The final task is to calculate the MP2 vibrational frequencies. The input file to accomplish these three tasks is as follows:

```
start h2o_freq
charge 1
geometry units angstroms
 O      0.0  0.0  0.0
 H      0.0  0.0  1.0
 H      0.0  1.0  0.0
end
basis
 H library sto-3g
 O library sto-3g
end
scf
 uhf; doublet
 print low
end
title "H2O+ : STO-3G UHF geometry optimization"
task scf optimize
basis
 H library 6-31g**
 O library 6-31g**
end
title "H2O+ : 6-31g** UMP2 geometry optimization"
task mp2 optimize
mp2; print none; end
scf; print none; end
title "H2O+ : 6-31g** UMP2 frequencies"
task mp2 freq
```

The START directive (START/RESTART tells NWChem that this run is to be started from the beginning. This directive need not be at the beginning of the input file, but it is commonly placed there. Existing database or vector files are to be ignored or overwritten. The entry h2o_freq on the START line is the prefix to be used for all files created by the calculation. This convention allows different jobs to run in the same directory or to share the same scratch directory SCRATCH_DIR/PERMANENT_DIR, as long as they use different prefix names in this field.

As in the first sample problem, the geometry is given in Cartesian coordinates. In this case, the units are specified as Angstrøms. (Since this is the default, explicit specification of the units is not actually necessary, however.) The CHARGE directive defines the total charge of the system. This calculation is to be done on an ion with charge +1.

A small basis set (STO-3G) is specified for the intial geometry optimization. Next, the multiple lines of the first SCF directive in the scf …end block specify details about the SCF calculation to be performed. Unrestricted Hartree-Fock is chosen here (by specifying the keyword uhf), rather than the default, restricted open-shell high-spin Hartree-Fock (ROHF). This is necessary for the subsequent MP2 calculation, because only UMP2 is currently available for open-shell systems (see Section 4). For open-shell systems, the spin multiplicity has to be specified (using doublet in this case), or it defaults to singlet. The print level is set to low to avoid verbose output for the starting basis calculations.

All input up to this point affects only the settings in the runtime database. The program takes its information from this database, so the sequence of directives up to the first TASK directive is irrelevant. An exchange of order of the different blocks or directives would not affect the result. The TASK directive, however, must be specified after all relevant input for a given problem. The TASK directive causes the code to perform the specified calculation using the parameters set in the preceding directives. In this case, the first task is an SCF calculation with geometry optimization, specified with the input scf and optimize. (See Tasks for a list of available tasks and operations.)

After the completion of any task, settings in the database are used in subsequent tasks without change, unless they are overridden by new input directives. In this example, before the second task (task mp2 optimize), a better basis set (6-31G**) is defined and the title is changed. The second TASK directive invokes an MP2 geometry optimization.

Once the MP2 optimization is completed, the geometry obtained in the calculation is used to perform a frequency calculation. This task is invoked by the keyword freq in the final TASK directive, task mp2 freq. The second derivatives of the energy are calculated as numerical derivatives of analytical gradients. The intermediate energies and gradients are not of interest in this case, so output from the SCF and MP2 modules is disabled with the PRINT directives.

4.1.5 Input Format and Syntax for Directives

This section describes the input format and the syntax used in the rest of this documentation to describe the format of directives. The input format for the directives used in NWChem is similar to that of UNIX shells, which is also used in other chemistry packages, most notably GAMESS-UK. An input line is parsed into whitespace (blanks or tabs) separating tokens or fields. Any token that contains whitespace must be enclosed in double quotes in order to be processed correctly. For example, the basis set with the descriptive name modified Dunning DZ must appear in a directive as “modified Dunning DZ”, since the name consists of three separate words.

4.1.5.1 Input Format

A (physical) line in the input file is terminated with a newline character (also known as a `return’ or `enter’ character). A semicolon (;) can be also used to indicate the end of an input line, allowing a single physical line of input to contain multiple logical lines of input. For example, five lines of input for the GEOMETRY directive can be entered as follows;

```
geometry
 O 0 0    0
 H 0  1.430 1.107
 H 0 -1.430 1.107
end
```

These same five lines could be entered on a single line, as

```
geometry; O 0 0 0; H 0 1.430 1.107; H 0 -1.430 1.107; end
```

This one physical input line comprises five logical input lines. Each logical or physical input line must be no longer than 1023 characters.

In the input file:

- a string, token, or field is a sequence of ASCII characters (NOTE: if the string includes blanks or tabs (i.e., white space), the entire string must be enclosed in double quotes).
- `\` (backslash) at the end of a line concatenates it with the next line. Note that a space character is automatically inserted at this point so that it is not possible to split tokens across lines. A backslash is also used to quote special characters such as whitespace, semi-colons, and hash symbols so as to avoid their special meaning (NOTE: these special symbols must be quoted with the backslash even when enclosed within double quotes).
- `;` (semicolon) is used to mark the end of a logical input line within a physical line of input.
- `#` (the hash or pound symbol) is the comment character. All characters following `#` (up to the end of the physical line) are ignored.
- If any input line (excluding Python programs begins with the string INCLUDE (ignoring case) and is followed by a valid file name, then the data in that file are read as if they were included into the current input file at the current line. Up to three levels of nested include files are supported. The user should note that inputting a basis set from the standard basis library (Basis Sets) uses one level of include.
- Data is read from the input file until an end-of-file is detected, or until the string EOF (ignoring case) is encountered at the beginning of an input line.

4.1.5.2 Format and syntax of directives

Directives consist of a directive name, keywords, and optional input, and may contain one line or many. Simple directives consist of a single line of input with one or more fields. Compound directives can have multiple input lines, and can also include other optional simple and compound directives. A compound directive is terminated with an END directive. The directives START (see START/RESTART) and ECHO (see ECHO) are examples of simple directives. The directive GEOMETRY (see Geometry) is an example of a compound directive.

Some limited checking of the input for self-consistency is performed by the input module, but most defaults are imposed by the application modules at runtime. It is therefore usually impossible to determine beforehand whether or not all selected options are consistent with each other.

In the rest of this document, the following notation and syntax conventions are used in the generic descriptions of the NWChem input.

- a directive name always appears in all-capitals, and in computer typeface (e.g., GEOMETRY, BASIS, SCF). Note that the case of directives and keywords is ignored in the actual input.
- a keyword always appears in lower case, in computer typeface (e.g., swap, print, units, bqbq).
- variable names always appear in lower case, in computer typeface, and enclosed in angle brackets to distinguish them from keywords (e.g., `<input_filename>`, `<basisname>`, `<tag>`).
- `$variable$` is used to indicate the substitution of the value of a variable.
- `()` is used to group items (the parentheses and other special symbols should not appear in the input).
- `|` separate exclusive options, parameters, or formats.
- `[]` enclose optional entries that have a default value.
- `<>` enclose a type, a name of a value to be specified, or a default value, if any.
- `\` is used to concatenate lines in a description.
- `...` is used to indicate indefinite continuation of a list.

An input parameter is identified in the description of the directive by prefacing the name of the item with the type of data expected, i.e.,

- string - an ASCII character string
- integer - integer value(s) for a variable or an array
- logical - true/false logical variable
- real - real floating point value(s) for a variable or an array
- double - synonymous with real

If an input item is not prefaced by one of these type names, it is assumed to be of type ``string''.

In addition, integer lists may be specified using Fortran triplet notation, which interprets lo:hi:inc as lo, lo+inc, lo+2*inc, …, hi. For example, where a list of integers is expected in the input, the following two lines are equivalent

```
7 10 21:27:2 1:3 99
7 10 21 23 25 27 1 2 3 99
```

(In Fortran triplet notation, the increment, if unstated, is 1; e.g., 1:3 = 1:3:1.)

The directive VECTORS is presented here as an example of an NWChem input directive. The general form of the directive is as follows:

```
VECTORS [input (<string input_movecs default atomic>) || \
            (project <string basisname> <string filename>)] \
        [swap [(alpha||beta)] <integer vec1 vec2> ...] \
        [output <string output_movecs default $file_prefix$.movecs>]
```

This directive contains three optional keywords, as indicated by the three main sets of square brackets enclosing the keywords input, swap, and output. The keyword input allows the user to specify the source of the molecular orbital vectors. There are two mutually exclusive options for specifying the vectors, as indicated by the || symbol separating the option descriptions;

```
(<string input_movecs default atomic>) || \
            (project <string basisname> <string filename>) \
```

The first option, `<string input_movecs default atomic>` , can be used to specify an ASCII character string for the parameter `input_movecs` . If no entry is specified, the code uses the default `atomic` (i.e., atomic guess). The second option, `project <string basisname> <string filename>` , contains the keyword `project` , which takes two string arguments. When this keyword is used, the vectors in file `<filename>` will be projected from the (smaller) basis `<basisname>` into the current atomic orbital (AO) basis.

The second keyword, `swap` , can be used to re-order the starting vectors, specifying the pairs of vectors to be swapped. As many pairs as the user wishes to have swapped can be listed for . The optional keywords alpha and beta allow the user to swap the alpha or beta spin orbitals.

The third keyword, `output` , allows the user to tell the code where to store the vectors, by specifying an ASCII string for the parameter `output_movecs` . If no entry is specified for this parameter, the default is to write the vectors back into either the user-specified MO vectors input file or, if this is not available, the file `$file_prefix$.movecs` .

A particular example of the VECTORS directive is shown below. It specifies both the input and output keywords, but does not use the swap option.

```
vectors input project "small basis" small_basis.movecs \
      output large_basis.movecs
```

This directive tells the code to generate input vectors by projecting from vectors in a smaller basis named “small basis”, which is stored in the file `small_basis.movecs` . The output vectors will be stored in the file `large_basis.movecs` .

The order of keyed optional entries within a directive should not matter, unless noted otherwise in the specific instructions

for a particular directive.

4.2 NWChem Architecture

As described in the Getting Started section, NWChem consists of independent modules that perform the various functions of the code. Examples include the input parser, self-consistent field (SCF) energy, SCF analytic gradient, and density functional theory (DFT) energy modules. The independent NWChem modules can share data only through a disk-resident database, which is similar to the GAMESS-UK dumpfile or the Gaussian checkpoint file. This allows the modules to share data, or to share access to files containing data.

It is not necessary for the user to be intimately familiar with the contents of the database in order to run NWChem. However, a nodding acquaintance with the design of the code will help in clarifying the logic behind the input requirements, especially when restarting jobs or performing multiple tasks within one job.

As detailed in the section describing the (nput file structure), all start-up directives are processed at the beginning of the job by the main program, and then the input module is invoked. Each input directive usually results in one or more entries being made in the database. When a TASK directive is encountered, control is passed to the appropriate module, which extracts relevant data from the database and any associated files. Upon completion of the task, the module will store significant results in the database, and may also modify other database entries in order to affect the behavior of subsequent computations.

4.2.1 Database Structure

Data is shared between modules of NWChem by means of the database. Three main types of information are stored in the data base: (1) arrays of data, (2) names of files that contain data, and (3) objects. Arrays are stored directly in the database, and contain the following information:

1. the name of the array, which is a string of ASCII characters (e.g., “reference energies”)
2. the type of the data in the array (i.e., real, integer, logical, or character)
3. the number (N) of data items in the array (Note: A scalar is stored as an array of unit length.)
4. the N items of data of the specified type

It is possible to enter data directly into the database using the SET directive. For example, to store a (64-bit precision) three-element real array with the name “reference energies” in the database, the directive is as follows:

```
set "reference energies" 0.0 1.0 -76.2
```

NWChem determines the data to be real (based on the type of the first element, 0.0), counts the number of elements in the array, and enters the array into the database.

Much of the data stored in the database is internally managed by NWChem and should not be modified by the user. However, other data, including some NWChem input options, can be freely modified.

Objects are built in the database by storing associated data as multiple entries, using an internally consistent naming convention. This data is managed exclusively by the subroutines (or methods) that are associated with the object. Currently, the code has two main objects: basis sets and geometries. GEOMETRY and BASIS present a complete discussion of the input to describe these objects.

As an illustration of what comprises a geometry object, the following table contains a partial listing of the database contents for a water molecule geometry named “test geom”. Each entry contains the field test geom, which is the unique name of the object.

```
Contents of RTDB h2o.db
-----------------------
Entry                          Type[nelem]
-------------------------  ----------------------
geometry:test geom:efield          double[3]
geometry:test geom:coords           double[9]
geometry:test geom:ncenter             int[1]
geometry:test geom:charges          double[3]
geometry:test geom:tags             char[6]
...
```

Using this convention, multiple instances of objects may be stored with different names in the same database. For example, if a user needed to do calculations considering alternative geometries for the water molecule, an input file could be constructed containing all the geometries of interest by storing them in the database under different names.

The runtime database contents for the file h2o.db listed above were generated from the user-specified input directive,

```
geometry "test geom"
  O     0.00000000    0.00000000    0.00000000
  H     0.00000000    1.43042809   -1.10715266
  H     0.00000000   -1.43042809   -1.10715266
end
```

The GEOMETRY directive allows the user to specify the coordinates of the atoms (or centers), and identify the geometry with a unique name.

Unless a specific name is defined for the geometry, (such as the name "test geom" shown in the example), the default name of geometry is assigned. This is the geometry name that computational modules will look for when executing a calculation. The SET directive can be used in the input to force NWChem to look for a geometry with a name other than geometry. For example, to specify use of the geometry with the name "test geom" in the example above, the SET directive is as follows:

```
set geometry "test geom"
```

NWChem will automatically check for such indirections when loading geometries. Storage of data associated with basis sets, the other database resident object, functions in a similar fashion, using the default name "ao basis" .

4.2.2 Persistence of data and restart

The database is persistent, meaning that all input data and output data (calculation results) that are not destroyed in the course of execution are permanently stored. These data are therefore available to subsequent tasks or jobs. This makes the input for restart jobs very simple, since only new or changed data must be provided. It also makes the behavior of successive restart jobs identical to that of multiple tasks within one job.

Sometimes, however, this persistence is undesirable, and it is necessary to return an NWChem module to its default behavior by restoring the database to its input-free state. In such a case, the UNSET directive can be used to delete all database entries associated with a given module (including both inputs and outputs).

5 Functionalities

5.1 Overview

NWChem provides many methods for computing the properties of molecular and periodic systems using standard quantum mechanical descriptions of the electronic wavefunction or density. Its classical molecular dynamics capabilities provide for the simulation of macromolecules and solutions, including the computation of free energies using a variety of force fields. These approaches may be combined to perform mixed quantum-mechanics and molecular-mechanics simulations.

The specific methods for determining molecular electronic structure, molecular dynamics, and pseudopotential plane-wave electronic structure and related attributes are listed in the following sections.

5.2 Molecular Electronic Structure

Methods for determining energies and analytic first derivatives with respect to atomic coordinates include the following:

- Hartree-Fock (RHF, UHF, high-spin ROHF)
- Gaussian orbital-based density functional theory (DFT) using many local and non-local exchange-correlation potentials (LDA, LSDA)
- second-order perturbation theory (MP2) with RHF and UHF references
- complete active space self-consistent field theory (CASSCF).

Analytic second derivatives with respect to atomic coordinates are available for RHF and UHF, and closed-shell DFT with all functionals.

The following methods are available to compute energies only:

- iterative CCSD, CCSDT, and CCSDTQ methods and their EOM-CC counterparts for RHF, ROHF, and UHF references
- active-space CCSDt and EOM-CCSDt approaches
- completely renormalized CR-CCSD(T), and CR-EOM-CCSD(T) correction to EOM-CCSD excitation energies
- locally renormalized CCSD(T) and CCSD(TQ) approaches
- non-iterative approaches based on similarity transformed Hamiltonian: the CCSD(2)T and CCSD(2) formalisms.
- MP2 with RHF reference and resolution of the identity integral approximation MP2 (RI-MP2) with RHF and UHF references
- selected CI with second-order perturbation correction.

For all methods, the following may be performed:

- single point energy calculations
- geometry optimization with constraints (minimization and transition state)
- molecular dynamics on the fully ab initio potential energy surface
- automatic computation of numerical first and second derivatives
- normal mode vibrational analysis in Cartesian coordinates
- ONIOM hybrid calculations
- Conductor-Like Screening Model (COSMO) calculations
- electrostatic potential from fit of atomic partial charges
- spin-free one-electron Douglas-Kroll calculations
- electron transfer (ET)
- vibrational SCF and DFT.

At the SCF and DFT level of theory various (response) properties are available, including NMR shielding tensors and indirect spin-spin coupling.

5.3 Quantum Mechanics/Molecular Mechanics (QM/MM)

The QM/MM module in NWChem provides a comprehensive set of capabilities to study ground and excited state properties of large-molecular systems. The QM/MM module can be used with practically any quantum mechanical method available in NWChem. The following tasks are supported

- single point energy and property calculations

- excited states calculation
- optimizations and transition state search
- dynamics
- free energy calculations.

5.4 Pseudopotential Plane-Wave Electronic Structure

The NWChem Plane-Wave (NWPW) module uses pseudopotentials and plane-wave basis sets to perform DFT calculations. This method's efficiency and accuracy make it a desirable first principles method of simulation in the study of complex molecular, liquid, and solid-state systems. Applications for this first principles method include the calculation of free energies, search for global minima, explicit simulation of solvated molecules, and simulations of complex vibrational modes that cannot be described within the harmonic approximation.

The NWPW module is a collection of three modules:

- PSPW (PSeudopotential Plane-Wave) A gamma point code for calculating molecules, liquids, crystals, and surfaces.
- Band A band structure code for calculating crystals and surfaces with small band gaps (e.g. semi-conductors and metals).
- PAW (Projector Augmented Wave) a gamma point projector augmented plane-wave code for calculating molecules, crystals, and surfaces.

These capabilities are available:

- constant energy and constant temperature Car-Parrinello molecular dynamics (extended Lagrangian dynamics)
- LDA, PBE96, and PBE0, exchange-correlation potentials (restricted and unrestricted)
- SIC, pert-OEP, Hartree-Fock, and hybrid functionals (restricted and unrestricted)
- Hamann, Troullier-Martins, Hartwigsen-Goedecker-Hutter norm-conserving pseudopotentials with semicore corrections
- geometry/unit cell optimization, frequency, transition-states
- fractional occupation of molecular orbitals for metals
- AIMD/MM capability in PSPW
- constraints needed for potential of mean force (PMF) calculation
- wavefunction, density, electrostatic, Wannier plotting
- band structure and density of states generation

5.5 Molecular Dynamics

The NWChem Molecular Dynamics (MD) module can perform classical simulations using the AMBER and CHARMM force fields, quantum dynamical simulations using any of the quantum mechanical methods capable of returning gradients, and mixed quantum mechanics molecular dynamics simulation and molecular mechanics energy minimization.

Classical molecular simulation functionality includes the following methods:

- single configuration energy evaluation
- energy minimization
- molecular dynamics simulation
- free energy simulation (MCTI and MSTP with single or dual topologies, double-wide sampling, and separation-shifted scaling).

The classical force field includes the following elements:

- effective pair potentials
- first-order polarization
- self-consistent polarization
- smooth particle mesh Ewald
- twin-range energy and force evaluation
- periodic boundary conditions
- SHAKE constraints
- constant temperature and/or pressure ensembles
- dynamic proton hopping using the Q-HOP methodology
- advanced system setup capabilities for biomolecular membranes.

6 Top-level Directives

6.1 Overview

Top-level directives are directives that can affect all modules in the code. Some specify molecular properties or other data that should apply to all subsequent calculations with the current database. However, most top-level directives provide the user with the means to manage the resources for a calculation and to start computations. As the first step in the execution of a job, NWChem scans the entire input file looking for start-up directives, which NWChem must process before all other input. The input file is then rewound and processed sequentially, and each directive is processed in the order in which it is encountered. In this second pass, start-up directives are ignored.

The following sections describe each of the top-level directives in detail, noting all keywords, options, required input, and defaults.

- START/RESTART
- PERMANENT_DIR
- SCRATCH_DIR
- MEMORY
- ECHO
- TITLE
- PRINT / NOPRINT
- SET
- UNSET
- STOP
- TASK
- ECCE_PRINT

7 Geometries

7.1 Overview

The GEOMETRY directive is a compound directive that allows the user to define the geometry to be used for a given

calculation. The directive allows the user to specify the geometry with a relatively small amount of input, but there are a large number of optional keywords and additional subordinate directives that the user can specify, if needed. The directive therefore appears to be rather long and complicated when presented in its general form, as follows:

```
GEOMETRY [<string name default geometry>] \
       [units <string units default angstroms>] \
       [(angstrom_to_au || ang2au) \
            <real angstrom_to_au default 1.8897265>] \
       [print [xyz] || noprint] \
       [center || nocenter] \
       [bqbq] \
       [autosym [real tol default 1d-2] || noautosym] \
       [autoz || noautoz] \
       [adjust] \
       [(nuc || nucl || nucleus) <string nucmodel>]
 [SYMMETRY [group] <string group_name> [print] \
       [tol <real tol default 1d-2>]]
 [ [LOAD] [format xyz||pdb]  [frame <int frame>] \
       [select [not] \
           [name <string atomname>] \
           [rname <string residue-name>]
           [id  <int atom-id>|<int range atom-id1:atom-id2> ... ]
           [resi <int residue-id>|<int range residue-id1:residue-id2> ... ]
       ]
 <string filename> ]

 <string tag> <real x y z> [vx vy vz] [charge <real charge>] \
       [mass <real mass>] \
       [(nuc || nucl || nucleus) <string nucmodel>]
 ... ]
 [ZMATRIX || ZMT || ZMAT
     <string tagn> <list_of_zmatrix_variables>
     ...
     [VARIABLES
         <string symbol> <real value>
         ... ]
     [CONSTANTS
         <string symbol> <real value>
         ... ]
  (END || ZEND)]
  [ZCOORD
      CVR_SCALING <real value>
      BOND    <integer i> <integer j> \
            [<real value>] [<string name>] [constant]
      ANGLE   <integer i> <integer j> \
                [<real value>] [<string name>] [constant]
      TORSION <integer i> <integer j> <integer k> <integer l> \
            [<real value>] [<string name>] [constant]
   END]

   [SYSTEM surface  <molecule polymer surface crystal default molecule>
       lat_a <real lat_a> lat_b <real lat_b> lat_c <real lat_c>
       alpha <real alpha> beta <real beta> gamma <real gamma>
   END]
 END
```

The three main parts of the GEOMETRY directive are:

- keywords on the first line of the directive (to specify such optional input as the geometry name, input units, and print level for the output)
- symmetry information
- Cartesian coordinates or Z-matrix input to specify the locations of the atoms and centers
- lattice parameters (needed only for periodic systems)

The following sections present the input for this compound directive in detail, describing the options available and the usages of the various keywords in each of the three main parts.

- Keywords for the GEOMETRY directive
- SYMMETRY: Symmetry Group Input
- Names of 3-dimensional space groups
- Cartesian coordinate input
- ZMATRIX: Z-matrix input

- ZCOORD: Forcing internal coordinates
- SYSTEM: Lattice parameters for periodic systems
- LOAD: Load geometry from XYZ file

8 Basis sets

8.1 Overview

NWChem currently supports basis sets consisting of generally contracted Cartesian Gaussian functions up to a maximum angular momentum of six (*h* functions), and also *sp* (or L) functions. The BASIS directive is used to define these, and also to specify use of an Effective Core Potential that is associated with a basis set.

The basis functions to be used for a given calculation can be drawn from a standard set in the Basis set library that is included in the release of NWChem. Alternatively, the user can specify particular functions explicitly in the input, to define a particular basis set.

8.2 BASIS directive

The general form of the BASIS directive is as follows:

```
BASIS [<string name default "ao basis">] \
    [(spherical || cartesian) default cartesian] \
    [(print || noprint) default print]  \
    [rel] [bse]
  <string tag> library [<string tag_in_lib>] \
            <string standard_set> [file <filename>] \
            [except <string tag list>] [rel]
    ...
  <string tag> <string shell_type> [rel]
    <real exponent> <real list_of_coefficients>
    ...
END
```

The following sections examine the keywords on the first line of the BASIS directive:

8.2.1 Basis set NAME

NAME ：
By default, the basis set is stored in the database with the name “ao basis”. Another name may be specified in the BASIS directive, thus, multiple basis sets may be stored simultaneously in the database. Also, the DFT and RI-MP2 modules and the Dyall-modified-Dirac relativistic method require multiple basis sets with specific names. The user can associate the “ao basis” with another named basis using the SET directive (see SET).

8.2.2 SPHERICAL or CARTESIAN

SPHERICAL || CARTESIAN ：
The keywords spherical and cartesian offer the option of using either spherical-harmonic (5 *d*, 7 *f*, 9 *g*, …) or Cartesian (6*d*, 10 *f*, 15 *g*, …) angular functions. The default is Cartesian. Note that the correlation-consistent basis sets were designed using spherical harmonics and to use these, the spherical keyword should be present in the BASIS directive. The use of spherical functions also helps eliminate problems with linear dependence.

	Cartesian	Spherical
1	s	s
1	p_x	p_x
2	p_y	p_y
3	p_z	p_z
1	d_{xx}	d_{xy}
2	d_{xy}	d_{yz}
3	d_{xz}	$d_{z^2-x^2-y^2}$
4	d_{yy}	d_{xz}
5	d_{yz}	$d_{x^2-y^2}$
6	d_{zz}	
1	f_{xxx}	$f_{xxy-yyy}$
2	f_{xxy}	f_{xyz}
3	f_{xxz}	$f_{yzz-xxy-yyy}$
4	f_{xyy}	$f_{zzz-xxz-yyz}$
5	f_{xyz}	$f_{-xzz+xxx+xyy}$
6	f_{xzz}	$f_{xxz-yyz}$
7	f_{yyy}	$f_{xyy-xxx}$
8	f_{yyz}	
9	f_{yzz}	
10	f_{zzz}	

Order of functions.

8.2.3 PRINT keyword

PRINT or NOPRINT：
The default is for the input module to print all basis sets encountered. Specifying the keyword noprint allows the user to suppress this output.

8.2.4 REL keyword

REL：
This keyword marks the entire basis as a relativistic basis for the purposes of the Dyall-modified-Dirac relativistic integral code. The marking of the basis set is necessary for the code to make the proper association between the relativistic shells

in the ao basis and the shells in the large and/or small component basis. This is only necessary for basis sets which are to be used as the ao basis. The user is referred to the section on Dyall's modified Dirac-Hamiltonian approximation for more details.

8.2.5 BSE keyword

BSE ***New in NWChem 7.2.0***:
This keyword loads the basis set library using data in $NWCHEM_TOP/src/basis/libraries.bse from basisetexchanger.org. CAVEAT: use of this keyword will also use the spherical/cartesian keywords from the basis library files.

8.3 Basis sets tags

Basis sets are associated with centers by using the tag of a center in a geometry that has either been input by the user or is available elsewhere. Each atom or center with the same tag will have the same basis set. All atoms must have basis functions assigned to them – only dummy centers (X or Bq) may have no basis functions. To facilitate the specification of the geometry and the basis set for any chemical system, the matching process of a basis set tag to a geometry tag first looks for an exact match. If no match is found, NWChem will attempt to match, ignoring case, the name or symbol of the element. E.g., all hydrogen atoms in a system could be labeled " H1 ", " H2 ", …, in the geometry but only one basis set specification for " H " or " hydrogen " is necessary. If desired, a special basis may be added to one or more centers (e.g., " H1 ") by providing a basis for that tag. If the matching mechanism fails then NWChem stops with an appropriate error message.

A special set of tags, "*" and tags ending with a "*" (E.g. " H* ") can be used in combination with the keyword library. These tags facilitate the definition of a certain type of basis set of all atoms, or a group of atoms, in a geometry using only a single or very few basis set entries. The "*" tag will not place basis sets on dummy atoms, Bq* can be used for that if necessary.

Examined next is how to reference standard basis sets in the basis set library, and finally, how to define a basis set using exponents and coefficients.

8.4 Basis set library

The keyword library associated with each specific tag entry specifies that the calculation will use the standard basis set in NWChem for that center. The string <standard_set> is the name that identifies the functions in the library. The names of standard basis sets are not case sensitive. For a complete list of basis sets and associated ECPs in the NWChem library see the available basis sets or the Basis Set Exchange for naming conventions and their specifications.

The general form of the input line requesting basis sets from the NWChem basis set library is:

```
<string tag> library [<string tag_in_lib>] \
        <string standard set> [file < filename> \
        [except <string tag list>] [rel]
  ...
```

For example, the NWChem basis set library contains the Dunning cc-pvdz basis set. These may be used as follows

```
basis
  oxygen library cc-pvdz
  hydrogen library cc-pvdz
end
```

A default path of the NWChem basis set libraries is provided on installation of the code, but a different path can be defined by specifying the keyword file, and one can explicitly name the file to be accessed for the basis functions. For example,

```
basis
  o  library 3-21g file /usr/d3g681/nwchem/library
  si library 6-31g file /usr/d3g681/nwchem/libraries/
end
```

This directive tells the code to use the basis set 3-21g in the file /usr/d3g681/nwchem/library for atom o and to use the basis set 6-31g in the directory /usr/d3g681/nwchem/libraries/ for atom si, rather than look for them in the default libraries. When a directory is defined the code will search for the basis set in a file with the name 6-31g.

The "*" tag can be used to efficiently define basis set input directives for large numbers of atoms. An example is:

```
basis
  * library 3-21g
end
```

This directive tells the code to assign the basis sets 3-21g to all the atom tags defined in the geometry. If one wants to place a different basis set on one of the atoms defined in the geometry, the following directive can be used:

```
basis
  * library 3-21g except H
end
```

This directive tells the code to assign the basis sets 3-21g to all the atoms in the geometry, except the hydrogen atoms. Remember that the user will have to explicitly define the hydrogen basis set in this directive! One may also define tags that end with a "*":

```
basis
  oxy* library 3-21g
end
```

This directive tells the code to assign the basis sets 3-21g to all atom tags in the geometry that start with "oxy".

If standard basis sets are to be placed upon a dummy center, the variable <tag_in_lib> must also be entered on this line, to identify the correct atom type to use from the basis function library (see the ghost atom example in SET and below). For example: To specify the cc-pvdz basis for a calculation on the water monomer in the dimer basis, where the dummy oxygen and dummy hydrogen centers have been identified as bqo and bqh respectively, the BASIS directive is as follows:

```
basis
  o   library cc-pvdz
  h   library cc-pvdz
  bqo library o cc-pvdz
  bqh library h cc-pvdz
end
```

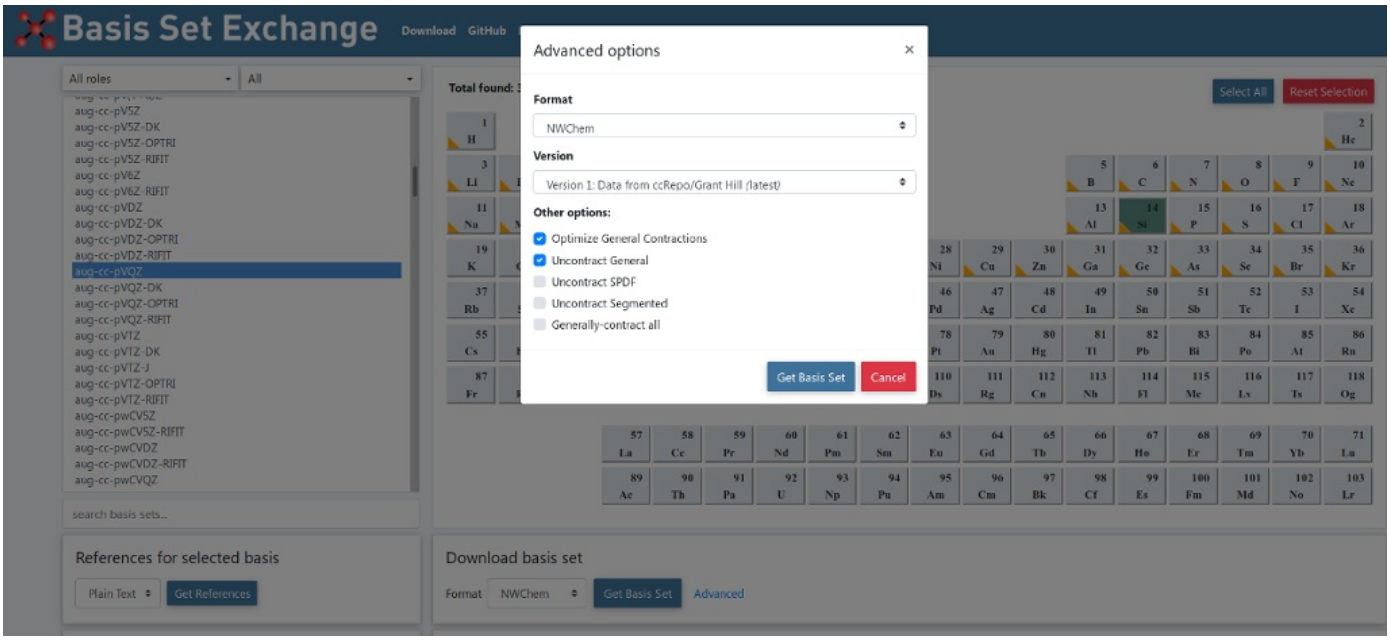
A special dummy center tag is bq*, which will assign the same basis set to all bq centers in the geometry. Just as with the * tag, the except list can be used to assign basis sets to unique dummy centers.

The library basis sets can also be marked as relativistic by adding the rel keyword to the tag line. See the section on relativistic all-electron approximations for more details. The correlation consistent basis sets have been contracted for relativistic effects and are included in the standard library.

There are also contractions in the standard library for both a point nucleus and a finite nucleus of Gaussian shape. These are usually distinguished by the suffix _pt and _fi. It is the user's responsibility to ensure that the contraction matches the nuclear type specified in the geometry object. The specification of a finite nucleus basis set does NOT automatically set the nuclear type for that atom to be finite. See Geometries for information.

8.5 How to use basis files from https://www.basissetexchange.org (NEW in 2019)

In order to ensure compatibility with the existing basis libraries available in NWChem, we suggest the user to select the "Advanced Options" menu and tick the boxes "Optimize General Contractions" and "Uncontract General", as in the image below, when downloading basis files from www.basissetexchange.org



As an alternative, basis set files downloaded from the basissetexchange.org website are available in the NWChem source code (after release 7.0.0). In order to switch from the default basis libraries to the library formed by files downloaded from www.basissetexchange.org, the following environment variable setting is required

```
NWCHEM_BASIS_LIBRARY=$NWCHEM_TOP/src/basis/libraries.bse/
```

8.6 Explicit basis set definition

If the basis sets in the library or available in other external files are not suitable for a given calculation, the basis set may be explicitly defined. A generally contracted Gaussian basis function is associated with a center using an input line of the following form:

```
<string tag> <string shell_type> [rel]
  <real exponent> <real list_of_coefficients>
  ...
```

The variable identifies the angular momentum of the shell, *s*, *p*, *d*, ... NWChem is configured to handle up to*h* shells. The keyword `rel` marks the shell as relativistic – see the Section on relativistic all-electron approximations for more details. Subsequent lines define the primitive function exponents and contraction coefficients. General contractions are specified by including multiple columns of coefficients.

The following example defines basis sets for the water molecule:

```
basis spherical
  oxygen s
    11720.0000    0.000710  -0.000160
     1759.0000    0.005470  -0.001263
      400.8000    0.027837  -0.006267
      113.7000    0.104800  -0.025716
       37.0300    0.283062  -0.070924
       13.2700    0.448719  -0.165411
        5.0250    0.270952  -0.116955
        1.0130    0.015458   0.557368
        0.3023   -0.002585   0.572759
  oxygen s
        0.3023    1.000000
  oxygen p
       17.7000    0.043018
        3.8540    0.228913
        1.0460    0.508728
        0.2753    0.460531
  oxygen p
        0.2753    1.000000
  oxygen d
        1.1850    1.000000
  hydrogen s
       13.0100    0.019685
        1.9620    0.137977
        0.4446    0.478148
        0.1220    0.501240
  hydrogen s
        0.1220    1.000000
  hydrogen p
        0.7270    1.000000
  oxygen s
        0.01      1.0
  hydrogen s
        0.02974   1.0
  hydrogen p
        0.141     1.0
end
```

Explicit basis set specifications are available from thebasis set exchange.

8.7 Combinations of library and explicit basis set input

The user can use a mixture of library basis and explicit basis set input to define the basis sets used on the various atoms.

For example, the following BASIS directive augments the Dunning cc-pvdz basis set for the water molecule with a diffuse s-shell on oxygen and adds the aug-cc-pVDZ diffuse functions onto the hydrogen.

```
basis spherical
  oxygen library cc-pvdz
  hydrogen library cc-pvdz
  oxygen s
    0.01 1.0
  hydrogen library "aug-cc-pVDZ Diffuse"
end
```

The resulting basis set defined is identical to the one defined above in the explicit basis set input.

9 Effective Core Potentials

9.1 Overview

Effective core potentials (ECPs) are a useful means of replacing the core electrons in a calculation with an effective potential, thereby eliminating the need for the core basis functions, which usually require a large set of Gaussians to describe them. In addition to replacing the core, they may be used to represent relativistic effects, which are largely confined to the core. In this context, both the scalar (spin-free) relativistic effects and spin-orbit (spin-dependent) relativistic effects may be included in effective potentials. NWChem has the facility to use both, and these are described in the next two sections.

A brief recapitulation of the development of RECPs is given here, following L.F. Pacios and P.A. Christiansen, J. Chem. Phys. 82, 2664 (1985). The process can be viewed as starting from an atomic Dirac-Hartree-Fock calculation, done in jj coupling, and producing relativistic effective potentials (REPs) for each l and j value, $U^{\rm REP}_{lj}$ which for example contains the Coulomb potential of the core electrons balanced by the part of the nuclear attraction which cancels the core electron charge. The residue is expressed in a semi-local form,

$$U^{REP} = U^{REP}_{LJ}(r) + \sum_{l=0}^{L-1} \sum_{j=|l-1/2|}^{l+1/2} \left[U^{REP}_{lj}(r) - U^{REP}_{LJ}(r)] \right] \sum_{m=-j}^j \vert lj\ m \rangle \langle lj\ m \vert$$

where L is one larger than the maximum angular momentum in the atom. The scalar potential is obtained by averaging the REPs for each j for a given l to give an averaged relativistic effective potential, or AREP

$$U^{AREP}_l(r) = \frac{1}{2l+1} \left[lU^{REP}_{l-1/2}(r) + (l+1) U^{\rm REP}_{l+1/2}(r) \right]$$

These are summed into the full potential

$$U^{AREP}(r) = U^{AREP}_L(r) + \sum_{l=0}^L \sum_{m=-l}^l \left[U^{AREP}_{l}(r) - U^{AREP}_{L}(r) \right] \vert l\ m \rangle \langle l\ m \vert$$

The spin-orbit potential is obtained from the difference between the REPs for the two j values for a given l, and may be represented in terms of an effective spin-orbit operator,

$$H^{SO} = \mathbf{\hat{s}} \cdot \sum_{l=1}^{L-1} \frac{2}{2l+1} \Delta U^{REP}_l \sum_{mm'}\vert {lm} \rangle \langle{lm} \vert \mathbf{\hat{l}} \vert {lm'} \rangle \langle{lm'} \vert$$

where

$$\Delta U^{REP}_{l} = U^{REP}_{l+1/2}(r) - U^{REP}_{l-1/2}(r)$$

The relavistic potential U^{REP} is the sum of H^{SO} and U^{AREP}.

The spin-orbit integrals generated by NWChem are the integrals over the sum, including the factor of $(2/(2l+1)$ as an effective spin-orbit operator without further factors introduced.

The effective potentials, both scalar and spin-orbit, are fitted to Gaussians with the form

$$U_{l}(r) = r^{-2} \sum_{k} A_{lk} r^{n_{lk}} e^{-B_{lk}r^{2}}$$

where A_{lk} is the contraction coefficient, n_{lk} is the exponent of the r term (r-exponent), and B_{lk} is the Gaussian exponent. The n_{lk} exponent is shifted by 2, in accordance with most of the ECP literature and implementations, i.e., $n_{lk} = 0$ implies r^{-2}. The current implementation allows n_{lk} values of only 0, 1, or 2.

9.2 Scalar ECPs

The optional directive ECP allows the user to describe an effective core potential (ECP) in terms of contracted Gaussian functions as given above. Potentials using these functions must be specified explicitly by user input in the ECP directive. This directive has essentially the same form and properties as the standard BASIS directive, except for essential differences required for ECPs. Because of this, the ECP is treated internally as a basis set. The form of the input for the ECP directive is as follows:

```
ECP [<string name default "ecp basis">] \
    [print || noprint default print]
  <string tag> library [<string tag_in_lib>] \
           <string standard_set> [file <filename>] \
           [except<string tag list>]
  <string tag> [nelec] <integer number_of_electrons_replaced>
    ...
  <string tag> <string shell_type>
  <real r-exponent> <real Gaussian-exponent> <real list_of_coefficients>
    ...
END
```

ECPs are automatically segmented, even if general contractions are input. The projection operators defined in an ECP are spherical by default, so there is no need to include the CARTESIAN or SPHERICAL keyword as there is for a standard basis set. ECPs are associated with centers in geometries through tags or names of centers. These tags must match in the same manner as for basis sets the tags in a GEOMETRY and ECP directives, and are limited to sixteen (16) characters. Each center with the same tag will have the same ECP. By default, the input module prints each ECP that it encounters. The NOPRINT option can be used to disable printing. There can be only one active ECP, even though several may exist in the input deck. The ECP modules load ecp basis inputs along with any ao basis inputs present. ECPs may be used in both energy and gradient calculations.

ECPs are named in the same fashion as geometries or regular basis sets, with the default name being “ecp basis”. It should be clear from the above discussion on geometries and database entries how indirection is supported. All directives that are in common with the standard Gaussian basis set input have the same function and syntax.

As for regular basis sets, ECPs may be obtained from the standard library. For a complete list of basis sets and associated ECPs in the NWChem library see the available basis sets or the Basis Set Exchange for naming conventions and their specifications.

The keyword nelec allows the user to specify the number of core electrons replaced by the ECP. Additional input lines define the specific coefficients and exponents. The variable <shell_type> is used to specify the components of the ECP. The keyword ul entered for <shell_type> denotes the local part of the ECP. This is equivalent to the highest angular momentum functions specified in the literature for most ECPs. The standard entries (*s*, *p*, *d*, etc.) for shell_type specify the angular momentum projector onto the local function. The shell type label of *s* indicates the *ul-s* projector input, *p* indicates the *ul-p*, etc.

For example, the Christiansen, Ross and Ermler ARECPs are available in the standard basis set library named crenbl_ecp .
To perform a calculation on uranyl UO_2^{2+} with all-electron oxygen (aug-cc-pvdz basis), and uranium with an ARECP and using the corresponding basis the following input can be used

```
geometry
  U 0 0  0
  O 0 0  1.65
  O 0 0 -1.65
end
basis
  U library crenbl_ecp
  O library aug-cc-pvdz
end
ecp
  U library crenbl_ecp
end
```

The following is an example of explicit input of an ECP for H_2CO.
It defines an ECP for the carbon and oxygen atoms in the molecule.

```
ecp
 C nelec 2 # ecp replaces 2 electrons on C
 C ul     # d
       1      80.0000000      -1.60000000
       1      30.0000000      -0.40000000
       2       0.5498205      -0.03990210
 C s      # s - d
       0       0.7374760       0.63810832
       0     135.2354832      11.00916230
       2       8.5605569      20.13797020
 C p      # p - d
       2      10.6863587      -3.24684280
       2      23.4979897       0.78505765
 O nelec 2 # ecp replaces 2 electrons on O
 O ul     # d
       1      80.0000000      -1.60000000
       1      30.0000000      -0.40000000
       2       1.0953760      -0.06623814
 O s      # s - d
       0       0.9212952       0.39552179
       0      28.6481971       2.51654843
       2       9.3033500      17.04478500
 O p      # p - s
       2      52.3427019      27.97790770
       2      30.7220233     -16.49630500
end
```

Various ECPs without a local function are available, including those of the Stuttgart group. For those, no$_{ul}$ part needs to be defined. To define the absence of the local potential, simply specify one contraction with a zero coefficient:

```
 <string tag> ul
 2    1.00000    0.00000
```

9.3 Spin-orbit ECPs

The Spin-orbit ECPs can be used with the Density Functional Approach, but one has to run the calculations without symmetry. Note: when a Hartree-Fock method is specified the spin-orbit input will be ignored.

Spin-orbit ECPs are fitted in precisely the same functional form as the scalar RECPs and have the same properties, with the exception that there is no local potential u_l, no s potential and no effective charge has to be defined. Spin-orbit potentials are specified in the same way as ECPs except that the directive SO is used instead of ECP. Note that there currently are no spin-orbit ECPs defined in the standard NWChem library. The SO directive is as follows:

```
SO [<string name default "so basis">] \
    [print || noprint default print]
  <string tag> library [<string tag_in_lib>] \
           <string standard_set> [file <filename>]
           [except `<string tag list>]
    ...
  <string tag> <string shell_type>
  <real r-exponent> <real Gaussian-exponent> <real list_of_coefficients>
    ...
END
```

Note: in the literature the coefficients of the spin-orbit potentials are NOT always defined in the same manner. The NWChem code assumes that the spin-orbit potential defined in the input is of the form:

For example, in the literature (most of) the Stuttgart potentials are defined as\(\Delta U_{l}\) and, hence, have to be multiplied by \(2/(2l+1)\) (Note: On the Stuttgart/Köln web pages
https://www.tc.uni-koeln.de/PP/clickpse.en.html,
spin-orbit potentials have already been corrected by the appropriate scaling factor and can be used as is). On the other hand, the CRENBL potentials in the published papers are defined as \(\,\!\frac{l}{2l+1} \Delta U_{l}\) have been corrected with the \(2/l\) factor, so make sure the appropriate scaling is applied).

For example, to use the Stuttgart/Köln ECP and SO-ECP for Hg (ECP60MDF) in NWChem.
The following URL will display bot the the ECP and SO parts.
http://www.tc.uni-koeln.de/cgi-bin/pp.pl?language=en,format=molpro,element=Hg,job=getecp,ecp=ECP60MDF
The highlighted section (last four lines) below is the SO part.

The un-highlighted part (first five lines) is the ECP.

```
!  Q=20., MEFIT, MCDHF+Breit, Ref 37.
ECP,Hg,60,5,4;
1; 2,1.000000,0.000000;
2; 2,12.413071,275.774797; 2,6.897913,49.267898;
4; 2,11.310320,80.506984; 2,10.210773,161.034824; 2,5.939804,9.083416; 2,5.019755,18.367773;
4; 2,8.407895,51.137256; 2,8.214086,76.707459; 2,4.012612,6.561821; 2,3.795398,9.818070;
2; 2,3.273106,9.429001; 2,3.208321,12.494856;
2; 2,4.485296,-6.338414; 2,4.513200,-8.099863;
4; 2,11.310320,-161.013967;2,10.210773,161.034824;2,5.939804,-18.166832;2,5.019755,18.367773;
4; 2,8.407895,-51.137256; 2,8.214086,51.138306; 2,4.012612,-6.561821; 2,3.795398,6.545380;
2; 2,3.273106,-6.286001; 2,3.208321,6.247428;
2; 2,4.485296,3.169207; 2,4.513200,-3.239945;
! References:
! [37] D. Figgen, G. Rauhut, M. Dolg, H. Stoll, Chem. Phys. 311, 227 (2005).
```

The corresponding NWChem input is

```
ecp
Hg nelec 60
Hg ul
2      1.0000000             0.0000000
Hg S
2     12.4130710           275.7747970
2      6.8979130            49.2678980
Hg P
2     11.3103200            80.5069840
2     10.2107730           161.0348240
2      5.9398040             9.0834160
2      5.0197550            18.3677730
Hg D
2      8.4078950            51.1372560
2      8.2140860            76.7074590
2      4.0126120             6.5618210
2      3.7953980             9.8180700
Hg F
2      3.2731060             9.4290010
2      3.2083210            12.4948560
Hg G
2      4.4852960            -6.3384140
2      4.5132000            -8.0998630
end

so
Hg P
2  11.310320  161.013967
2  10.210773  161.034824
2   5.939804  -18.166832
2   5.019755   18.367773
Hg D
2   8.407895  -51.137256
2   8.214086   51.138306
2   4.012612   -6.561821
2   3.795398    6.545380
Hg F
2   3.273106   -6.286001
2   3.208321    6.247428
Hg G
2   4.485296    3.169207
2   4.513200   -3.239945
end
```

9.3.1 Websites with Spin-Orbits ECPs

- https://www.tc.uni-koeln.de/PP/clickpse.en.html
- https://people.clarkson.edu/~pchristi/reps.html
- http://www.qchem.pnpi.spb.ru/recp
- https://pseudopotentiallibrary.org

10 Relativistic all-electron approximations

10.1 Overview

All methods which include treatment of relativistic effects are ultimately based on the Dirac equation, which has a four component wave function. The solutions to the Dirac equation describe both positrons (the “negative energy” states) and electrons (the “positive energy” states), as well as both spin orientations, hence the four components. The wave function may be broken down into two-component functions traditionally known as the large and small components; these may further be broken down into the spin components.

The implementation of approximate all-electron relativistic methods in quantum chemical codes requires the removal of the negative energy states and the factoring out of the spin-free terms. Both of these may be achieved using a transformation of the Dirac Hamiltonian known in general as a Foldy-Wouthuysen (FW) transformation. Unfortunately this transformation cannot be represented in closed form for a general potential, and must be approximated. One popular approach is that originally formulated by Douglas and Kroll[1] and developed by Hess[2,3]. This approach decouples the positive and negative energy parts to second order in the external potential (and also fourth order in the fine structure constant, α). Other approaches include the Zeroth Order Regular Approximation (ZORA)[4,5,6,7] and modification of the Dirac equation by Dyall[8], and involves an exact FW transformation on the atomic basis set level[9,10].

Since these approximations only modify the integrals, they can in principle be used at all levels of theory. At present the Douglas-Kroll and ZORA implementations can be used at all levels of theory whereas Dyall’s approach is currently available at the Hartree-Fock level. The derivatives have been implemented, allowing both methods to be used in geometry optimizations and frequency calculations.

10.2 RELATIVISTIC directive

The RELATIVISTIC directive provides input for the implemented relativistic approximations and is a compound directive that encloses additional directives specific to the approximations:

```
RELATIVISTIC
  [DOUGLAS-KROLL [<string (ON||OFF) default ON> \
            <string (FPP||DKH||DKFULL||DK3||DK3FULL) default DKH>]  ||
   ZORA [ (ON || OFF) default ON ] ||
   DYALL-MOD-DIRAC [ (ON || OFF) default ON ]  ||
            [ (NESC1E || NESC2E) default NESC1E ] ]  ||
   X2C [ (ON || OFF) default ON ]
  [CLIGHT <real clight default 137.0359895>]
END
```

Only one of the methods may be chosen at a time. If both methods are found to be on in the input block, NWChem will stop and print an error message. There is one general option for both methods, the definition of the speed of light in atomic units:

```
CLIGHT <real clight default 137.0359895>
```

The following sections describe the optional sub-directives that can be specified within the RELATIVISTIC block.

10.2.1 Douglas-Kroll approximation

The spin-free and spin-orbit one-electron Douglas-Kroll approximation have been implemented. The use of relativistic effects from this Douglas-Kroll approximation can be invoked by specifying:

```
DOUGLAS-KROLL [<string (ON||OFF) default ON> \
            <string (FPP||DKH||DKFULL|DK3|DK3FULL) default DKH>]
```

The ON|OFF string is used to turn on or off the Douglas-Kroll approximation. By default, if the DOUGLAS-KROLL keyword is found, the approximation will be used in the calculation. If the user wishes to calculate a non-relativistic quantity after turning on Douglas-Kroll, the user will need to define a new RELATIVISTIC block and turn the approximation OFF. The user could also simply put a blank RELATIVISTIC block in the input file and all options will be turned off.

The FPP is the approximation based on free-particle projection operators[11] whereas the DKH and DKFULL approximations are based on external-field projection operators[12]. The latter two are considerably better approximations than the former. DKH is the Douglas-Kroll-Hess approach and is the approach that is generally implemented in quantum chemistry codes. DKFULL includes certain cross-product integral terms ignored in the DKH approach (see for example Häberlen and Rösch[13]). The third-order Douglas-Kroll approximation has been implemented by T. Nakajima and K. Hirao[14][15]. This approximation can be called using DK3 (DK3 without cross-product integral terms) or DK3FULL (DK3 with cross-product integral terms).

The contracted basis sets used in the calculations should reflect the relativistic effects, i.e. one should use contracted basis sets which were generated using the Douglas-Kroll Hamiltonian. Basis sets that were contracted using the non-relativistic (Schödinger) Hamiltonian WILL PRODUCE ERRONEOUS RESULTS for elements beyond the first row. See appendix A for available basis sets and their naming convention.

NOTE: we suggest that spherical basis sets are used in the calculation. The use of high quality cartesian basis sets can lead to numerical inaccuracies.

In order to compute the integrals needed for the Douglas-Kroll approximation the implementation makes use of a fitting basis set (see literature given above for details). The current code will create this fitting basis set based on the given ao basis by simply uncontracting that basis. This again is what is commonly implemented in quantum chemistry codes that include the Douglas-Kroll method. Additional flexibility is available to the user by explicitly specifying a Douglas-Kroll fitting basis set. This basis set must be named D-K basis (see Basis Sets).

10.2.2 Zeroth Order regular approximation (ZORA)

The spin-free and spin-orbit one-electron zeroth-order regular approximation (ZORA) have been implemented. ZORA can be accessed only via the DFT and SO-DFT modules. The use of relativistic effects with ZORA can be invoked by specifying:

```
ZORA [<string (ON||OFF) >
```

The ON | OFF string is used to turn on or off ZORA. No default is present, therefore ZORA keyword need to be followed by ON in order for the approximation to be used in the calculation. If the user wishes to calculate a non-relativistic quantity after turning on ZORA, the user will need to define a new RELATIVISTIC block and turn the approximation OFF. The user can also simply put a blank RELATIVISTIC block in the input file and all options will be turned off.

To increase the accuracy of ZORA calculations, the following settings may be used in the relativistic block

```
relativistic
   zora on
   zora:cutoff 1d-30
end
```

To invoke the relativistic ZORA model potential approach due to van Wullen (references[16] and [17]).

For model potentials constructed from 4-component densities:

```
relativistic
   zora on
   zora:cutoff 1d-30
   modelpotential  1
end
```

For model potentials constructed from 2-component densities:

```
relativistic
   zora on
   zora:cutoff 1d-30
   modelpotential  2
end
```

Both approaches are comparable in accuracy and depends on the system.

10.2.3 Dyall's Modified Dirac Hamitonian approximation

The approximate methods described in this section are all based on Dyall's modified Dirac Hamiltonian. This Hamiltonian is entirely equivalent to the original Dirac Hamiltonian, and its solutions have the same properties. The modification is achieved by a transformation on the small component, extracting out $\sigma \cdot \mathbf{p}/2mc$. This gives the modified small component the same symmetry as the large component, and in fact it differs from the large component only at order α^2. The advantage of the modification is that the operators now resemble the operators of the Breit-Pauli Hamiltonian, and can be classified in a similar fashion into spin-free, spin-orbit and spin-spin terms. It is the spin-free terms which have been implemented in NWChem, with a number of further approximations.

The first is that the negative energy states are removed by a normalized elimination of the small component (NESC), which is equivalent to an exact Foldy-Wouthuysen (EFW) transformation. The number of components in the wave function is thereby effectively reduced from 4 to 2. NESC on its own does not provide any advantages, and in fact complicates things because the transformation is energy-dependent. The second approximation therefore performs the elimination on an atom-by-atom basis, which is equivalent to neglecting blocks which couple different atoms in the EFW transformation. The advantage of this approximation is that all the energy dependence can be included in the contraction coefficients of the basis set. The tests which have been done show that this approximation gives results well within chemical accuracy. The third approximation neglects the commutator of the EFW transformation with the two-electron Coulomb interaction, so that the only corrections that need to be made are in the one-electron integrals. This is the equivalent of the Douglas-Kroll(-Hess) approximation as it is usually applied.

The use of these approximations can be invoked with the use of the DYALL-MOD-DIRAC directive in the RELATIVISTIC directive block. The syntax is as follows.

```
DYALL-MOD-DIRAC [ (ON || OFF) default ON ]
        [ (NESC1E || NESC2E) default NESC1E ]
```

The ON | OFF string is used to turn on or off the Dyall's modified Dirac approximation. By default, if the DYALL-MOD-DIRAC keyword is found, the approximation will be used in the calculation. If the user wishes to calculate a non-relativistic quantity after turning on Dyall's modified Dirac, the user will need to define a new RELATIVISTIC block and turn the approximation OFF. The user could also simply put a blank RELATIVISTIC block in the input file and all options will be turned off.

Both one- and two-electron approximations are available NESC1E || NESC2E, and both have analytic gradients. The one-electron approximation is the default. The two-electron approximation specified by NESC2E has some sub options which are placed on the same logical line as the DYALL-MOD-DIRAC directive, with the following syntax:

```
NESC2E [ (SS1CENT [ (ON || OFF) default ON ] || SSALL) default SSALL ]
    [ (SSSS [ (ON || OFF) default ON ] || NOSSSS) default SSSS ]
```

The first sub-option gives the capability to limit the two-electron corrections to those in which the small components in any density must be on the same center. This reduces the *(LL|SS)* contributions to at most three-center integrals and the *(SS|SS)* contributions to two centers. For a case with only one relativistic atom this option is redundant. The second controls the inclusion of the *(SS|SS)* integrals which are of order α^4. For light atoms they may safely be neglected, but for heavy atoms they should be included.

In addition to the selection of this keyword in the RELATIVISTIC directive block, it is necessary to supply basis sets in addition to the ao basis. For the one-electron approximation, three basis sets are needed: the *atomic FW* basis set, the *large component* basis set and the *small* component basis set. The atomic FW basis set should be included in the ao basis. The large and small components should similarly be incorporated in basis sets named large component and small component, respectively. For the two-electron approximation, only two basis sets are needed. These are the large component and the small component. The large component should be included in the ao basis and the small component is specified separately as small component, as for the one-electron approximation. This means that the two approximations can not be run correctly without changing the ao basis, and it is up to the user to ensure that the basis sets are correctly specified.

There is one further requirement in the specification of the basis sets. In the ao basis, it is necessary to add the el keyword either to the basis directive or the library tag line (See below for examples). The former marks the basis functions specified

by the tag as relativistic, the latter marks the whole basis as relativistic. The marking is actually done at the unique shell level, so that it is possible not only to have relativistic and nonrelativistic atoms, it is also possible to have relativistic and nonrelativistic shells on a given atom. This would be useful, for example, for diffuse functions or for high angular momentum correlating functions, where the influence of relativity was small. The marking of shells as relativistic is necessary to set up a mapping between the ao basis and the large and/or small component basis sets. For the one-electron approximation the large and small component basis sets MUST be of the same size and construction, i.e. differing only in the contraction coefficients.

It should also be noted that the relativistic code will NOT work with basis sets that contain sp shells, nor will it work with ECPs. Both of these are tested and flagged as an error.

10.2.4 Examples for DYALL-MOD-DIRAC

Some examples follow. The first example sets up the data for relativistic calculations on water with the one-electron approximation and the two-electron approximation, using the library basis sets.

```
start h2o-dmd
geometry units bohr
symmetry c2v
  O       0.000000000    0.000000000   -0.009000000
  H       1.515260000    0.000000000   -1.058900000
  H      -1.515260000    0.000000000   -1.058900000
end
basis "fw" rel
  oxygen library cc-pvdz_pt_sf_fw
  hydrogen library cc-pvdz_pt_sf_fw
end
basis "large"
  oxygen library cc-pvdz_pt_sf_lc
  hydrogen library cc-pvdz_pt_sf_lc
end
basis "large2" rel
  oxygen library cc-pvdz_pt_sf_lc
  hydrogen library cc-pvdz_pt_sf_lc
end
basis "small"
  oxygen library cc-pvdz_pt_sf_sc
  hydrogen library cc-pvdz_pt_sf_sc
end
set "ao basis" fw
set "large component" large
set "small component" small
relativistic
  dyall-mod-dirac
end
task scf
set "ao basis" large2
unset "large component"
set "small component" small
relativistic
  dyall-mod-dirac nesc2e
end
task scf
```

The second example has oxygen as a relativistic atom and hydrogen nonrelativistic.

```
start h2o-dmd2
geometry units bohr
symmetry c2v
  O      0.000000000    0.000000000   -0.009000000
  H      1.515260000    0.000000000   -1.058900000
  H     -1.515260000    0.000000000   -1.058900000
end
basis "ao basis"
  oxygen library cc-pvdz_pt_sf_fw rel
  hydrogen library cc-pvdz
end
basis "large component"
  oxygen library cc-pvdz_pt_sf_lc
end
basis "small component"
  oxygen library cc-pvdz_pt_sf_sc
end
relativistic
  dyall-mod-dirac
end
task scf
```

10.2.5 X2C: exact two-component relativistic Hamiltonian

The exact two-component Hamiltonian[18][19] has been implemented in the DFT module of NWChem[20][21][22].

```
X2C [<string (ON||OFF) default ON>
```

The ON | OFF string is used to turn on or off X2C. By default, if the X2C keyword is found, the approximation will be used in the calculation. If the user wishes to calculate a non-relativistic quantity after turning on X2C, the user will need to define a new RELATIVISTIC block and turn the approximation OFF. The user can also simply put a blank RELATIVISTIC block in the input file and all options will be turned off.

To increase the accuracy of X2C calculations, the following settings may be used in the relativistic block

```
relativistic
   x2c on
   x2c:cutoff 1d-15
end
```

10.3 References

1. Douglas, M.; Kroll, N.M. (1974). “Quantum electrodynamical corrections to the fine structure of helium”. Annals of Physics 82: 89-155. DOI:10.1016/0003-4916(74)90333-9.
2. Hess, B.A. (1985). “Applicability of the no-pair equation with free-particle projection operators to atomic and molecular structure calculations”. Physical Review A 32: 756-763. DOI:10.1103/PhysRevA.32.756.
3. Hess, B.A. (1986). “Relativistic electronic-structure calculations employing a two-component no-pair formalism with external-field projection operators”. Physical Review A 33: 3742-3748. DOI:10.1103/PhysRevA.33.3742.
4. Chang, C; Pelissier, M; Durand, M (1986). “Regular Two-Component Pauli-Like Effective Hamiltonians in Dirac Theory”. Physica Scripta 34: 394. DOI:10.1088/0031-8949/34/5/007.
5. van Lenthe, E (1996). “The ZORA Equation” (in English).
6. Faas, S.; Snijders, J.G.; van Lenthe, J.H.; van Lenthe, E.; Baerends, E.J. (1995). “The ZORA formalism applied to the Dirac-Fock equation”. Chemical Physics Letters 246: 632-640. DOI:10.1016/0009-2614(95)01156-0.
7. Nichols, P.; Govind, N.; Bylaska, E.J.; de Jong, W.A. (2009). “Gaussian Basis Set and Planewave Relativistic Spin-Orbit Methods in NWChem”. Journal of Chemical Theory and Computation 5: 491-499. DOI:10.1021/ct8002892.
8. Dyall, K.G. (1994). “An exact separation of the spin-free and spin-dependent terms of the Dirac–Coulomb–Breit Hamiltonian”. The Journal of Chemical Physics 100: 2118-2127. DOI:10.1063/1.466508.
9. Dyall, K.G. (1997). “Interfacing relativistic and nonrelativistic methods. I. Normalized elimination of the small component in the modified Dirac equation”. The Journal of Chemical Physics 106: 9618-9626. DOI:10.1063/1.473860.
10. Dyall, K.G.; Enevoldsen, T. (1999). “Interfacing relativistic and nonrelativistic methods. III. Atomic 4-spinor expansions and integral approximations”. The Journal of Chemical Physics 111: 10000-10007. DOI:10.1063/1.480353.
11. Hess, B.A. (1985). “Applicability of the no-pair equation with free-particle projection operators to atomic and molecular structure calculations”. Physical Review A 32: 756-763. DOI:10.1103/PhysRevA.32.756.

12. Hess, B.A. (1986). “Relativistic electronic-structure calculations employing a two-component no-pair formalism with external-field projection operators”. Physical Review A 33: 3742-3748. DOI:10.1103/PhysRevA.33.3742.
13. Haeberlen, O.D.; Roesch, N. (1992). “A scalar-relativistic extension of the linear combination of Gaussian-type orbitals local density functional method: application to AuH, AuCl and Au2”. Chemical Physics Letters 199: 491-496. DOI:10.1016/0009-2614(92)87033-L.
14. Nakajima, T.; Hirao, K. (2000). “Numerical illustration of third-order Douglas-Kroll method: atomic and molecular properties of superheavy element 112”. Chemical Physics Letters 329: 511-516. DOI:10.1016/S0009-2614(00)01035-6.
15. Nakajima, T.; Hirao, K. (2000). “The higher-order Douglas–Kroll transformation”. The Journal of Chemical Physics 113: 7786-7789. DOI:10.1063/1.1316037.
16. van Wullen, C. (1998). “Molecular density functional calculations in the regular relativistic approximation: Method, application to coinage metal diatomics, hydrides, fluorides and chlorides, and comparison with first-order relativistic calculations”. The Journal of Chemical Physics 109: 392-399 DOI:10.1063/1.476576
17. van Wullen, C.; Michauk, C. (2005). “Accurate and efficient treatment of two-electron contributions in quasirelativistic high-order Douglas-Kroll density-functional calculations”. The Journal of Chemical Physics 123, 204113 DOI:10.1063/1.2133731
18. Liu, W.; Peng, D. (2009). J. Chem. Phys. 2009, 131, 031104 DOI:10.1063/1.3159445
19. Saue, T. (2011). “Relativistic Hamiltonians for Chemistry: A Primer”. ChemPhysChem, 12, 3077–3094 DOI:10.1002/cphc.201100682
20. Autschbach, J.; Peng, D; Reiher, M. (2012). J. Chem. Theory Comput. 2012, 8, 4239–4248 DOI:10.1021/ct300623j
21. Peng, D.; Reiher, M. (2012). Theor. Chem. Acc. 131, 1081 DOI:10.1007/s00214-011-1081-y
22. Autschbach, J. (2021). Quantum Theory for Chemical Applications:From Basic Concepts to Advanced Topics, Oxford, University Press, Chapter 24 DOI:10.1093/oso/9780190920807.001.0001

11 Quantum-Mechanical-Methods

11.1 Hartree-Fock

11.1.1 Overview

The NWChem self-consistent field (SCF) module computes closed-shell restricted Hartree-Fock (RHF) wavefunctions, restricted high-spin open-shell Hartree-Fock (ROHF) wavefunctions, and spin-unrestricted Hartree-Fock (UHF) wavefunctions. The Hartree-Fock equations are solved using a conjugate-gradient method with an orbital Hessian based preconditioner[1]. The module supports both replicated data and distributed data Fock builders[2].

The SCF directive provides input to the SCF module and is a compound directive that encloses additional directives specific to the SCF module:

```
SCF
  ...
END
```

11.1.2 Wavefunction type

A spin-restricted, closed shell RHF calculation is performed by default. An error results if the number of electrons is inconsistent with this assumption. The number of electrons is inferred from the total charge on the system and the sum of the effective nuclear charges of all centers (atoms and dummy atoms, see GEOMETRY). The total charge on the system is zero by default, unless specified at some value by input on the CHARGE directive Total system charge.

The options available to define the SCF wavefunction and multiplicity are as follows:

```
SINGLET
DOUBLET
TRIPLET
QUARTET
QUINTET
SEXTET
SEPTET
OCTET
NOPEN <integer nopen default 0>
RHF
ROHF
UHF
```

The optional keywords SINGLET, DOUBLET, …, OCTET and NOPEN allow the user to specify the number of singly occupied orbitals for a particular calculation. SINGLET is the default, and specifies a closed shell; DOUBLET specifies one singly occupied orbital; TRIPLET specifies two singly occupied orbitals; and so forth. If there are more than seven singly occupied orbitals, the keyword NOPEN must be used, with the integer nopen defining the number of singly occupied orbitals (sometimes referred to as open shells).

If the multiplicity is any value other than SINGLET, the default calculation will be a spin-restricted, high-spin, open-shell SCF calculation (keyword ROHF). The open-shell orbitals must be the highest occupied orbitals. If necessary, any starting vectors may be rearranged through the use of the SWAP keyword on the VECTORS directive to accomplish this.

A spin-unrestricted solution can also be performed by specifying the keyword UHF. In UHF calculations, it is assumed that the number of singly occupied orbitals corresponds to the difference between the number of alpha-spin and beta-spin orbitals. For example, a UHF calculation with 2 more alpha-spin orbitals than beta-spin orbitals can be obtained by specifying

```
scf
  triplet ; uhf    # (Note: two logical lines of input)
  ...
end
```

The user should be aware that, by default, molecular orbitals are symmetry adapted in NWChem. This may not be desirable for fully unrestricted wavefunctions. In such cases, the user has the option of defeating the defaults by specifying the keywords ADAPT OFF and SYM OFF .

The keywords RHF and ROHF are provided in the code for completeness. It may be necessary to specify these in order to modify the behavior of a previous calculation (see NWChem Architecture for restart behavior).

11.1.3 SYM: use of symmetry

```
SYM <string (ON||OFF) default ON>
```

This directive enables/disables the use of symmetry to speed up Fock matrix construction (via the petite-list or skeleton algorithm) in the SCF, if symmetry was used in the specification of the geometry. Symmetry adaptation of the molecular orbitals is not affected by this option. The default is to use symmetry if it is specified in the geometry directive.

For example, to disable use of symmetry in Fock matrix construction:

```
sym off
```

11.1.4 ADAPT: symmetry adaptation of MOs

```
ADAPT <string (ON||OFF) default ON>
```

The default in the SCF module calculation is to force symmetry adaption of the molecular orbitals. This does not affect the speed of the calculation, but without explicit adaption the resulting orbitals may be symmetry contaminated for some problems. This is especially likely if the calculation is started using orbitals from a distorted geometry.

The underlying assumption in the use of symmetry in Fock matrix construction is that the density is totally symmetric. If the orbitals are symmetry contaminated, this assumption may not be valid – which could result in incorrect energies and poor convergence of the calculation. It is thus advisable when specifying ADAPT OFF to also specify SYM OFF (Use of Symmetry).

11.1.5 TOL2E: integral screening threshold

```
TOL2E <real tol2e default min(10e-7 , 0.01*thresh)>
```

The variable tol2e is used in determining the integral screening threshold for the evaluation of the energy and related Fock-like matrices. The Schwarz inequality is used to screen the product of integrals and density matrices in a manner that results in an accuracy in the energy and Fock matrices that approximates the value specified for tol2e .

It is generally not necessary to set this parameter directly. Specify instead the required precision in the wavefunction, using the THRESH directive (Convergence threshold). The default threshold is the minimum of 10^{-7} and 0.01 times the requested convergence threshold for the SCF calculation (Convergence threshold).

The input to specify the threshold explicitly within the SCF directive is, for example:

```
tol2e 1e-9
```

For very diffuse basis sets, or for high-accuracy calculations it might be necessary to set this parameter. A value of 10^{-12} is sufficient for nearly all such purposes.

11.1.6 VECTORS: input/output of MO vectors

```
VECTORS [[input] (<string input_movecs default atomic>) || \
            (project <string basisname> <string filename>) || \
            (fragment <string file1> [<string file2> ...])] \
      [swap [alpha||beta] <integer vec1 vec2> ...] \
      [reorder <integer atom1 atom2> ...] \
      [output <string output_filename default input_movecs>] \
      [lock]
      [rotate  <string input_geometry> <string input_movecs>]
```

The VECTORS directive allows the user to specify the source and destination of the molecular orbital vectors. In a startup calculation (see START), the default source for guess vectors is a diagonalized Fock matrix constructed from a superposition of the atomic density matrices for the particular problem. This is usually a very good guess. For a restarted calculation, the default is to use the previous MO vectors.

The optional keyword INPUT allows the user to specify the source of the input molecular orbital vectors as any of the following:

- ATOMIC – eigenvectors of a Fock-like matrix formed from a superposition of the atomic densities (the default guess). See Atomic guess and Accuracy of initial guess.
- HCORE – eigenvectors of the bare-nucleus Hamiltonian or the one-electron Hamiltonian.
- filename – the name of a file containing the MO vectors from a previous calculation. Note that unless the path is fully qualified, or begins with a dot (“.”), then it is assumed to reside in the directory for permanent files (see File directories).
- PROJECT basisname filename – projects the existing MO vectors in the file filename from the smaller basis with name basisname into the current basis. The definition of the basis basisname must be available in the current database, and the basis must be smaller than the current basis. In addition, the geometry used for the previous calculations must have the atoms in the same order and in the same orientation as the current geometry.
- FRAGMENT file1 ... – assembles starting MO vectors from previously performed calculations on fragments of the system and is described in more detail in Superposition of fragment molecular orbitals. Even though there are some significant restrictions in the use of the initial implementation of this method, this is the most powerful initial guess option within the code. It is very effective for open shell metallic systems.

- ROTATE input_geometry input_movecs – rotates MO vectors generated at a previous geometry to the current active geometry.

The molecular orbitals are saved every iteration if more than 600 seconds have elapsed, and also at the end of the calculation. At completion (converged or not), the SCF module always canonically transforms the molecular orbitals by separately diagonalizing the closed-closed, open-open, and virtual-virtual blocks of the Fock matrix.

The name of the file used to store the MO vectors is determined as follows:

- if the OUTPUT keyword was specified on the VECTORS directive, then the filename that follows this keyword is used, or
- if the input vectors were read from a file, this file is reused for the output vectors (overwriting the input vectors); else,
- a default file name is generated in the directory for permanent files (File directories) by prepending ".movecs" with the file prefix, i.e., ".movecs".

The name of this file is stored in the database so that a subsequent SCF calculation will automatically restart from these MO vectors.

Applications of this directive are illustrated in the following examples.

Example 1:

```
vectors output h2o.movecs
```

Assuming a start-up calculation, this directive will result in use of the default atomic density guess, and will output the vectors to the file h2o.movecs.

Example 2:

```
vectors input initial.movecs output final.movecs
```

This directive will result in the initial vectors being read from the file "initial.movecs". The results will be written to the file final.movecs. The contents of "initial.movecs" will not be changed.

Example 3:

```
vectors input project "small basis" small.movecs
```

This directive will cause the calculation to start from vectors in the file "small.movecs" which are in a basis named "small basis". The output vectors will be written to the default file "".

11.1.6.1 VECTORS SWAP keyword

Once starting vectors have been obtained using any of the possible options, they may be reordered through use of the SWAP keyword. This optional keyword requires a list of orbital pairs that will be swapped. For UHF calculations, separate SWAP keywords may be provided for the alpha and beta orbitals, as necessary.

An example of use of the SWAP directive:

```
vectors input try1.movecs swap 173 175 174 176 output try2.movecs
```

This directive will cause the initial orbitals to be read from the file "try1.movecs". The vectors for the orbitals within the pairs 173-175 will be swapped with those within 174-176, so the resulting order is 175, 176, 173, 174. The final orbitals obtained in the calculation will be written to the file "try2.movecs".

The swapping of orbitals occurs as a sequential process in the order (left to right) input by the user. Thus, regarding each pair as an elementary transposition it is possible to construct arbitrary permutations of the orbitals. For instance, to apply the permutation (6 7 8 9) we note that this permutation is equal to (6 7)(7 8)(8 9) , and thus may be specified as

```
vectors swap 8 9  7 8  6 7
```

Another example, now illustrating this feature for a UHF calculation, is the directive

```
vectors swap beta 4 5 swap alpha 5 6
```

This input will result in the swapping of the 5-6 alpha orbital pair and the 4-5 beta orbital pair. (All other items in the input use the default values.)

11.1.6.2 VECTORS LOCK keyword

The LOCK keyword allows the user to specify that the ordering of orbitals will be locked to that of the initial vectors, insofar as possible. The default is to order by ascending orbital energies within each orbital space. One application where locking might be desirable is a calculation where it is necessary to preserve the ordering of a previous geometry, despite flipping of the orbital energies. For such a case, the LOCK directive can be used to prevent the SCF calculation from changing the ordering, even if the orbital energies change.

11.1.6.3 VECTORS REORDER keyword

The mapping of the MO's to the nuclei can be changed using the REORDER keyword. Once starting vectors have been obtained using any of the possible options, the REORDER keyword moves the MO coefficients between atoms listed in the integer list. This keyword is particularly useful for calculating localized electron and hole states.

This optional keyword requires a list containing the new atom ordering. It is not necessary to provide separate lists for alpha and beta orbitals.

An example of use of the REORDER keyword:

```
vectors input try1.movecs reorder 2 1 output try2.movecs
```

This directive will cause the initial orbitals to be read from the file "try1.movecs". The MO coefficients for the basis functions on atom 2 will be swapped with those on atom 1. The final orbitals obtained in the calculation will be written to the file "try2.movecs".

11.1.6.4 VECTORS ROTATE keyword

The following example shows how the ROTATE keyword can be used to rotate MO vectors calculated at geometry$_{\text{geom1}}$ to geometry $_{\text{geom2}}$, which has a different rotational orientation:

```
set geometry geom1
dft
 vectors input atomic output geom1.mo
end
task dft
set geometry geom2
dft
 vectors input rotate geom1 geom1.mo output geom2.mo
end
task dft
```

11.1.6.5 VECTORS FRAGMENT: Superposition of fragment molecular orbitals

The fragment initial guess is particularly useful in the following instances:

- The system naturally decomposes into molecules that can be treated individually, e.g., a cluster.
- One or more fragments are particularly hard to converge and therefore much time can be saved by converging them independently.

- A fragment (e.g., a metal atom) must be prepared with a specific occupation. This can often be readily accomplished with a calculation on the fragment using dummy charges to model a ligand field.
- The molecular occupation predicted by the atomic initial guess is often wrong for systems with heavy metals which may have partially occupied orbitals with lower energy than some doubly occupied orbitals. The fragment initial guess avoids this problem.

```
VECTORS [input] fragment <string file1> [<string file2> ...]
```

The molecular orbitals are formed by superimposing the previously generated orbitals of fragments of the molecule being studied. These fragment molecular orbitals must be in the same basis as the current calculation. The input specifies the files containing the fragment molecular orbitals. For instance, in a calculation on the water dimer, one might specify

```
vectors fragment h2o1.movecs h2o2.movecs
```

where h2o1.movecs contains the orbitals for the first fragment, and h2o2.movecs contains the orbitals for the second fragment.

A complete example of the input for a calculation on the water dimer using the fragment guess is as follows:

```
start dimer
title "Water dimer SCF using fragment initial guess"
geometry dimer
  O   -0.595   1.165  -0.048
  H    0.110   1.812  -0.170
  H   -1.452   1.598  -0.154
  O    0.724  -1.284   0.034
  H    0.175  -2.013   0.348
  H    0.177  -0.480   0.010
end
geometry h2o1
  O   -0.595   1.165  -0.048
  H    0.110   1.812  -0.170
  H   -1.452   1.598  -0.154
end
geometry h2o2
  O    0.724  -1.284   0.034
  H    0.175  -2.013   0.348
  H    0.177  -0.480   0.010
end
basis
  o library 3-21g
  h library 3-21g
end
set geometry h2o1
scf; vectors input atomic output h2o1.movecs; end
task scf
set geometry h2o2
scf; vectors input atomic output h2o2.movecs; end
task scf
set geometry dimer
scf
vectors input fragment h2o1.movecs h2o2.movecs \
        output dimer.movecs
end
task scf
```

First, the geometry of the dimer and the two monomers are specified and given names. Then, after the basis specification, calculations are performed on the fragments by setting the geometry to the appropriate fragment (SET) and redirecting the output molecular orbitals to an appropriately named file. Note also that use of the atomic initial guess is forced, since the default initial guess is to use any existing MOs which would not be appropriate for the second fragment calculation. Finally, the dimer calculation is performed by specifying the dimer geometry, indicating use of the fragment guess, and redirecting the output MOs.

The following points are important in using the fragment initial guess:

1. The fragment calculations must be in the same basis set as the full calculation.
2. The order of atoms in the fragments and the order in which the fragment files are specified must be such that when the fragment basis sets are concatenated all the basis functions are in the same order as in the full system. This is readily

accomplished by first generating the full geometry with atoms for each fragment contiguous, splitting this into numbered fragments and specifying the fragment MO files in the correct order on the VECTORS directive.

3. The occupation of orbitals is preserved when they are merged from the fragments to the full molecule and the resulting occupation must match the requested occupation for the full molecule. E.g., a triplet ROHF calculation must be comprised of fragments that have a total of exactly two open-shell orbitals.
4. Because of these restrictions, it is not possible to introduce additional atoms (or basis functions) into fragments for the purpose of cleanly breaking real bonds. However, it is possible, and highly recommended, to introduce additional point charges to simulate the presence of other fragments.
5. MO vectors of partially occupied or strongly polarized systems are very sensitive to orientation. While it is possible to specify the same fragment MO vector file multiple times in the VECTORS directive, it is usually much better to do a separate calculation for each fragment.
6. Linear dependencies which were present in a fragment calculation may be magnified in the full calculation. When this occurs, some of the fragment's highest virtual orbitals will not be copied to the full system, and a warning will be printed.

A more involved example is now presented. We wish to model the sextet state of Fe(III) complexed with water, imidazole and a heme with a net unit positive charge. The default atomic guess does not give the correct d^5 occupation for the metal and also gives an incorrect state for the double anion of the heme. The following performs calculations on all of the fragments. Things to note are:

1. The use of a dummy *+2* charge in the initial guess on the heme which in part simulates the presence of the metal ion, and also automatically forces an additional two electrons to be added to the system (the default net charge being zero).
2. The iron fragment calculation (charge +3, d^5, sextet) will yield the correct open-shell occupation for the full system. If, instead, the d-orbitals were partially occupied (e.g., the doublet state) it would be useful to introduce dummy charges around the iron to model the ligand field and thereby lift the degeneracy to obtain the correct occupation.
3. C_s symmetry is used for all of the calculations. It is not necessary that the same symmetry be used in all of the calculations, provided that the order and orientation of the atoms is preserved.
4. The unset scf:* directive is used immediately before the calculation on the full system so that the default name for the output MO vector file can be used, rather than having to specify it explicitly.

```
start heme6a1
title  "heme-H2O (6A1) from M.Dupuis"
############################################################
# Define the geometry of the full system and the fragments #
############################################################
geometry full-system
  symmetry cs
  H     0.438   -0.002    4.549
  C     0.443   -0.001    3.457
  C     0.451   -1.251    2.828
  C     0.452    1.250    2.828
  H     0.455    2.652    4.586
  H     0.461   -2.649    4.586
  N1    0.455   -1.461    1.441
  N1    0.458    1.458    1.443
  C     0.460    2.530    3.505
  C     0.462   -2.530    3.506
  C     0.478    2.844    1.249
  C     0.478    3.510    2.534
  C     0.478   -2.848    1.248
  C     0.480   -3.513    2.536
  C     0.484    3.480    0.000
  C     0.485   -3.484    0.000
  H     0.489    4.590    2.664
  H     0.496   -4.592    2.669
  H     0.498    4.573    0.000
  H     0.503   -4.577    0.000
  H    -4.925    1.235    0.000
  H    -4.729   -1.338    0.000
  C    -3.987    0.685    0.000
  N    -3.930   -0.703    0.000
  C    -2.678    1.111    0.000
  C    -2.622   -1.076    0.000
  H    -2.284    2.126    0.000
  H    -2.277   -2.108    0.000
  N    -1.838    0.007    0.000
```

```
  Fe    0.307    0.000    0.000
  O     2.673   -0.009    0.000
  H     3.238   -0.804    0.000
  H     3.254    0.777    0.000
end
geometry ring-only
  symmetry cs
  H     0.438   -0.002    4.549
  C     0.443   -0.001    3.457
  C     0.451   -1.251    2.828
  C     0.452    1.250    2.828
  H     0.455    2.652    4.586
  H     0.461   -2.649    4.586
  N1    0.455   -1.461    1.441
  N1    0.458    1.458    1.443
  C     0.460    2.530    3.505
  C     0.462   -2.530    3.506
  C     0.478    2.844    1.249
  C     0.478    3.510    2.534
  C     0.478   -2.848    1.248
  C     0.480   -3.513    2.536
  C     0.484    3.480    0.000
  C     0.485   -3.484    0.000
  H     0.489    4.590    2.664
  H     0.496   -4.592    2.669
  Bq    0.307    0.0      0.0    charge 2  # simulate the iron
end
geometry imid-only
  symmetry cs
  H     0.498    4.573    0.000
  H     0.503   -4.577    0.000
  H    -4.925    1.235    0.000
  H    -4.729   -1.338    0.000
  C    -3.987    0.685    0.000
  N    -3.930   -0.703    0.000
  C    -2.678    1.111    0.000
  C    -2.622   -1.076    0.000
  H    -2.284    2.126    0.000
  H    -2.277   -2.108    0.000
  N    -1.838    0.007    0.000
end
geometry fe-only
  symmetry cs
  Fe    .307    0.000    0.000
end
geometry water-only
  symmetry cs
  O     2.673   -0.009    0.000
  H     3.238   -0.804    0.000
  H     3.254    0.777    0.000
end
############################
# Basis set for everything #
############################
basis nosegment
 O  library 6-31g*
 N  library 6-31g*
 C  library 6-31g*
 H  library 6-31g*
 Fe  library "Ahlrichs pVDZ"
end
#############################################################
# SCF on the fragments for initial guess for full system #
#############################################################
scf; thresh 1e-2; end
set geometry ring-only
scf; vectors atomic swap 80 81 output ring.mo; end
task scf
set geometry water-only
scf; vectors atomic output water.mo; end
task scf
set geometry imid-only
scf; vectors atomic output imid.mo; end
task scf
charge 3
set geometry fe-only
scf; sextet; vectors atomic output fe.mo; end
task scf
#########################
# SCF on the full system #
#########################
unset scf:*     # This restores the defaults
charge 1
set geometry full-system
scf
 sextet
 vectors fragment ring.mo imid.mo fe.mo water.mo
 maxiter 50
```

```
end
task scf
```

11.1.6.6 Example of projecting smaller basis into larger basis

Key ingredient needed: definition of both the smaller and the larger basis set, plus mention of the small basis set in the "input project" line.

```
start he

geometry
 he 0 0 0
 symmetry oh
end

basis  small
 * library sto-3g
end
basis  large
 * library 3-21g
end

set "ao basis" small
scf
 vectors input atomic output small.mos
end
task scf

set "ao basis" large
scf
 vectors input project small small.mos output large.mos
end
task scf
```

11.1.6.7 Atomic guess orbitals with charged atoms

As noted above, the default guess vectors are based on superimposing the density matrices of the neutral atoms. If some atoms are significantly charged, this default guess may be improved upon by modifying the atomic densities. This is done by setting parameters that add fractional charges to the occupation of the valence atomic orbitals. Since the atomic SCF program does not have its own input block, the SET directive (SET) must be used to set these parameters.

The input specifies a list of tags (i.e., names of atoms in a geometry, see GEOMETRY) and the charges to be added to those centers. Two parameters must be set as follows:

```
set atomscf:tags_z <string list_of_tags>
set atomscf:z      <real list_of_charges>
```

The array of strings atomscf:tags_z should be set to the list of tags, and the array atomscf:z should be set to the list of charges which must be real numbers (not integers). All atoms that have a tag specified in the list of tags will be assigned the corresponding charge from the list of charges.

For example, the following specifies that all oxygen atoms with tag O be assigned a charge of -1 and all iron atoms with tag Fe be assigned a charge of +2

```
set atomscf:z       -1  2.0
set atomscf:tags_z    O  Fe
```

There are some limitations to this feature. It is not possible to add electrons to closed shell atoms, nor is it possible to remove all electrons from a given atom. Attempts to do so will cause the code to report an error, and it will not report further errors in the input for modifying the charge even when they are detected.

Finally, recall that the database is persistent (Data persistence) and that the modified settings will be used in subsequent atomic guess calculations unless the data is deleted from the database with the UNSET directive (UNSET).

11.1.7 Accuracy of initial guess

For SCF, the initial Fock-matrix construction from the atomic guess is performed to a default precision of 1e-7. However, other wavefunctions, notably DFT, use a lower precision. In charged, or diffuse basis sets, this precision may not be sufficient and could result in incorrect ordering of the initial orbitals. The accuracy may be increased with the following directive which should be inserted in the top-level of input (i.e., outside of the SCF input block) and before the TASK directive.

```
set tolguess 1e-7
```

11.1.8 THRESH – convergence threshold

```
THRESH  <real thresh default 1.0e-4>
```

This directive specifies the convergence threshold for the calculation. The convergence threshold is the norm of the orbital gradient, and has a default value in the code of 10^{-4}.

The norm of the orbital gradient corresponds roughly to the precision available in the wavefunction, and the energy should be converged to approximately the square of this number. It should be noted, however, that the precision in the energy will not exceed that of the integral screening tolerance. This tolerance (Integral screening threshold) is automatically set from the convergence threshold, so that sufficient precision is usually available by default.

The default convergence threshold suffices for most SCF energy and geometry optimization calculations, providing about 6-8 decimal places in the energy, and about four significant figures in the density and energy derivative with respect to nuclear coordinates. However, greater precision may be required for calculations involving weakly interacting systems, floppy molecules, finite-difference of gradients to compute the Hessian, and for post-Hartree-Fock calculations. A threshold of 10^{-6} is adequate for most such purposes, and a threshold of 10^{-8} might be necessary for very high accuracy or very weak interactions. A threshold of 10^{-8} should be regarded as the best that can be attained in most circumstances.

11.1.9 MAXITER – iteration limit

```
MAXITER <integer maxiter default 8>
```

The maximum number of iterations for the SCF calculation defaults to 20 for both ROHF/RHF and UHF calculations. For most molecules, this number of iterations is more than sufficient for the quadratically convergent SCF algorithm to obtain a solution converged to the default threshold (see Convergence threshold above). If the SCF program detects that the quadratically convergent algorithm is not efficient, then it will resort to a linearly convergent algorithm and increase the maximum number of iterations by 10.

Convergence may not be reached in the maximum number of iterations for many reasons, including input error (e.g., an incorrect geometry or a linearly dependent basis), a very low convergence threshold, a poor initial guess, or the fact that the system is intrinsically hard to converge due to the presence of many states with similar energies.

The following sets the maximum number of SCF iterations to 50:

```
maxiter 50
```

11.1.10 PROFILE – performance profile

This directive allows the user to obtain timing and parallel execution information about the SCF module. It is specified by the simple keyword

```
PROFILE
```

This option can be helpful in understanding the computational performance of an SCF calculation. However, it can introduce a significant overhead on machines that have expensive timing routines, such as the SUN.

11.1.11 DIIS – DIIS convergence

This directive allows the user to specify DIIS convergence rather than second-order convergence for the SCF calculation. The form of the directive is as follows:

```
DIIS
```

The implementation of this option is currently fairly rudimentary. It does not have level-shifting and damping, and does not support open shells or UHF. It is provided on an “as is” basis, and should be used with caution.

When the DIIS directive is specified in the input, the user has the additional option of specifying the size of the subspace for the DIIS extrapolation. This is accomplished with the DIISBAS directive, which is of the form:

```
DIISBAS <integer diisbas default 5>
```

The default of 5 should be adequate for most applications, but may be increased if convergence is poor. On large systems, it may be necessary to specify a lower value for diisbas, to conserve memory.

11.1.12 DIRECT and SEMIDIRECT: recomputation of integrals

In the context of SCF calculations direct means that all integrals are recomputed as required and none are stored. The other extreme are disk- or memory-resident (sometimes termed conventional) calculations in which all integrals are computed once and stored. Semi-direct calculations are between these two extremes with some integrals being precomputed and stored, and all other integrals being recomputed as necessary.

The default behavior of the SCF module is

- If enough memory is available, the integrals are computed once and are cached in memory.
- If there is not enough memory to store all the integrals at once, then 95% of the available disk space in the scratch directory (see File directories) is assumed to be available for this purpose, and as many integrals as possible are cached on disk (with no memory being used for caching). Some attempt is made to store the most expensive integrals in the cache.
- If there is not enough room in memory or on disk for all the integrals, then the ones that are not cached are recomputed in a semidirect fashion.

The integral file is deleted at the end of a calculation, so it is not possible to restart a semidirect calculation when the integrals are cached in memory or on disk. Many modern computer systems clear the fast scratch space at the end of each job, adding a further complication to the problem of restarting a parallel semidirect calculation.

A fully direct calculation (with recomputation of the integrals at each iteration) is forced by specifying the directive

```
DIRECT
```

Alternatively, the SEMIDIRECT directive can be used to control the default semidirect calculation by defining the amount of disk space and the cache memory size. The form of this directive is as follows:

```
SEMIDIRECT [filesize <integer filesize default disksize>]
      [memsize  <integer memsize default available>]
      [filename <string filename default $file_prefix.aoints$]
```

The keyword FILESIZE allows the user to specify the amount of disk space to be used per process for storing the integrals in 64-bit words. Similarly, the keyword MEMSIZE allows the user to specify the number of 64-bit words to be used per process for caching integrals in memory. (Note: If the amount of storage space specified by the entry for memsize is not available, the code cuts the value in half and checks again for available space. This process is repeated until the request is satisfied.)

By default, the integral files are placed into the scratch directory (see File directories). Specifying the keyword FILENAME overrides this default. The user-specified name entered in the string filename has the process number appended to it, so that each process has a distinct file but with a common base-name and directory. Therefore, it is not possible to use this keyword to specify different disks for different processes. The SCRATCH_DIR directive (see File directories) can be used for this purpose.

For example, to force full recomputation of all integrals:

```
direct
```

Exactly the same result could be obtained by entering the directive:

```
semidirect filesize 0 memsize 0
```

To disable the use of memory for caching integrals and limit disk usage by each process to 100 megawords (MW):

```
semidirect memsize 0 filesize 100000000
```

The integral records are typically 32769 words long and any non-zero value for filesize or memsize should be enough to hold at least one record.

11.1.12.1 Integral File Size and Format for the SCF Module

The file format is rather complex, since it accommodates a variety of packing and compression options and the distribution of data. This section presents some information that may help the user understand the output, and illustrates how to use the output information to estimate file sizes.

If integrals are stored with a threshold of greater than 10^{10}, then the integrals are stored in a 32-bit fixed-point format (with appropriate treatment for large values to retain precision). If integrals are stored with a threshold less than 10^{-10}, however, the values are stored in 64-bit floating-point format. If a replicated-data calculation is being run, then 8 bits are used for each basis function label, unless there are more than 256 functions, in which case 16 bits are used. If distributed data is being used, then the labels are always packed to 8 bits (the distributed blocks always being less than 256; labels are relative to the start of the block).

Thus, the number (*W*) of 64-bit words required to store *N* integrals, may be computed as

no. 64-bit words	labels	values
N	8-bit	32-bit
1.5N	16-bit	32-bit
1.5N	8-bit	64-bit
2N	16-bit	64-bit

Table 1: number (*W*) of 64-bit words required to store *N* integrals

The actual number of words required can exceed this computed value by up to one percent, due to bookkeeping overhead, and because the file itself is organized into fixed-size records.

With at least the default print level, all semidirect (not direct) calculations will print out information about the integral file and the number of integrals computed. The form of this output is as follows:

```
Integral file         = ./c6h6.aoints.0
Record size in doubles =  32769        No. of integs per rec  =  32768
Max. records in memory =      3        Max. records in file   =      5
No. of bits per label  =      8        No. of bits per value  =     32
#quartets = 2.0D+04  #integrals = 7.9D+05  direct = 63.6%  cached = 36.4%
```

The file information above relates only to process 0. The line of information about the number of quartets, integrals, etc., is a sum over all processes.

When the integral file is closed, additional information of the following form is printed:

```
------------------------------------------------------------------
EAF file 0: "./c6h6.aoints.0" size=262152 bytes
------------------------------------------------------------------
             write      read    awrite     aread      wait
             -----      ----    ------     -----      ----
    calls:       6        12         0         0         0
  data(b): 1.57e+06  3.15e+06  0.00e+00  0.00e+00
  time(s): 1.09e-01  3.12e-02                    0.00e+00
rate(mb/s): 1.44e+01  1.01e+02
------------------------------------------------------------------
 Parallel integral file used       4 records with       0 large values
```

Again, the detailed file information relates just to process 0, but the final line indicates the total number of integral records stored by all processes.

This information may be used to optimize subsequent calculations, for instance by assigning more memory or disk space.

11.1.13 SCF Convergence Control Options

Note to users: It is desired that the SCF program converge reliably with the default options for a wide variety of molecules. In addition, it should be guaranteed to converge for any system, with sufficient iterations.

The SCF program uses a preconditioned conjugate gradient (PCG) method that is unconditionally convergent. Basically, a search direction is generated by multiplying the orbital gradient (the derivative of the energy with respect to the orbital rotations) by an approximation to the inverse of the level-shifted orbital Hessian. In the initial iterations (see Controlling the Newton-Raphson), an inexpensive one-electron approximation to the inverse orbital Hessian is used. Closer to convergence, the full orbital Hessian is used, which should provide quadratic convergence. For both the full or one-electron orbital Hessians, the inverse-Hessian matrix-vector product is formed iteratively. Subsequently, an approximate line search is performed along the new search direction. If the exact Hessian is being employed, then the line search should require a single step (of unity). Preconditioning with approximate Hessians may require additional steps, especially in the initial iterations. It is the (approximate) line search that provides the convergence guarantee. The iterations required to solve the linear equations are referred to as micro-iterations. A macro-iteration comprises both the iterative solution and a line search.

Level-shifting plays the same role in this algorithm as it does in the conventional iterative solution of the SCF equations. The approximate Hessian used for preconditioning should be positive definite. If this is not the case, then level-shifting by a positive constant (Δ) serves to make the preconditioning matrix positive definite, by adding Δ to all of its eigenvalues. The level-shifts employed for the RHF orbital Hessian should be approximately four times (only twice for UHF) the value that one would employ in a conventional SCF. Level-shifting is automatically enabled in the early iterations, and the default options suffice for most test cases.

So why do things go wrong and what can be done to fix convergence problems? Most problems encountered so far arise either poor initial guesses or from small or negative eigenvalues of the orbital Hessian. The atomic orbital guess is usually very good. However, in calculations on charged systems, especially with open shells, incorrect initial occupations may result. The SCF might then converge very slowly since very large orbital rotations might be required to achieve the correct occupation or move charge large distances in the molecule. Possible actions are

- Modify the atomic guess by assigning charges to the atoms known to carry substantial charges (Atomic guess)
- Examining an analysis of the initial orbitals (Printing) and then swapping them to attain the desired occupation (VECTORS).
- Converging the calculation in a minimal basis set, which is usually easier, and then projecting into a larger basis set (VECTORS).
- Using the fragment orbital initial guess (Fragment molecular orbitals).

Small or negative Hessian eigenvalues can occur even though the calculation seem to be close to convergence (as measured by the gradient norm, or the off-diagonal Fock matrix elements). Small eigenvalues will cause the iterative linear equation solver to converge slowly, resulting in an excessive number of micro-iterations. This makes the SCF expensive in terms of computation time, and it is possible to exceed the maximum number of iterations without achieving the accuracy required for quadratic convergence – which causes more macro-iterations to be performed.

Two main options are available when a problem will not converge: Newton-Raphson can be disabled temporarily or permanently (see Controlling the Newton-Raphson), and level-shifting can be applied to the matrix (see Level-shifting). In some cases, both options may be necessary to achieve final convergence.

If there is reason to suspect a negative eigenvalue, the first course is to disable the Newton-Raphson iteration until the solution is closer to convergence. It may be necessary to disable it completely. At some point close to convergence, the Hessian will be positive definite, so disabling Newton-Raphson should yield a solution with approximately the same convergence rate as DIIS.

If temporarily disabling Newton-Raphson is not sufficient to achieve convergence, it may be necessary to disable it entirely and apply a small level-shift to the approximate Hessian. This should improve the convergence rate of the micro-iterations and stabilize the macro-iterations. The level-shifting will destroy exact quadratic convergence, but the optimization process is automatically adjusted to reflect this by enforcing conjugacy and reducing the accuracy to which the linear equations are solved. The net result of this is that the solution will do more macro-iterations, but each one should take less time than it would with the unshifted Hessian.

The following sections describe the directives needed to disable the Newton-Raphson iteration and specify level-shifting.

11.1.14 NR: controlling the Newton-Raphson

```
NR <real nr_switch default 0.1>
```

The exact orbital Hessian is adopted as the preconditioner when the maximum element of the orbital gradient is below the value specified for nr_switch. The default value is 0.1, which means that Newton-Raphson will be disabled until the maximum value of the orbital gradient (twice the largest off-diagonal Fock matrix element) is less than 0.1. To disable the Newton-Raphson entirely, the value of nr_switch must be set to zero. The directive to accomplish this is as follows:

```
nr 0
```

11.1.15 LEVEL: level-shifting the orbital Hessian

This directive allows the user to specify level-shifting to obtain a positive-definite preconditioning matrix for the SCF solution procedure. Separate level shifts can be set for the first-order convergent one-electron approximation to the Hessian used with the preconditioned conjugate gradient (PCG) method, and for the full Hessian used with the Newton-Raphson (NR) approach. It is also possible to change the level-shift automatically as the solution attains some specified accuracy. The form of the directive is as follows:

```
LEVEL [pcg <real initial default 20.0> \
      [<real tol default 0.5> <real final default 0.0>]] \
     [nr <real initial default 0.0> \
      [<real tol default 0.0> <real final default 0.0>]]
```

This directive contains only two keywords: one for the PCG method and the other for the exact Hessian (Newton Raphson, or NR). Use of PCG or NR is determined by the input specified for nr_switch on the NR directive, Controlling the Newton-Raphson above.

Specifying the keyword pcg on the LEVEL directive allows the user to define the level shifting for the approximate (i.e., PCG) method. Specifying the keyword nr allows the user to define the level shifting for the exact Hessians. In both options, the initial level shift is defined by the value specified for the variable initial. Optionally, tol can be specified independently with each keyword to define the level of accuracy that must be attained in the solution before the level shifting is changed to the value specified by input in the real variable final. Level shifts and gradient thresholds are specified in atomic units.

For the PCG method (as specified using the keyword pcg), the defaults for this input are 20.0 for initial, 0.5 for tol, and 0.0 for final. This means that the approximate Hessian will be shifted by 20.0 until the maximum element of the gradient falls below 0.5, at which point the shift will be set to zero.

For the exact Hessian (as specified using the keyword nr), the defaults are all zero. The exact Hessian is usually not shifted since this destroys quadratic convergence. An example of an input directive that applies a shift of 0.2 to the exact Hessian is as follows:

```
level nr 0.2
```

To apply this shift to the exact Hessian only until the maximum element of the gradient falls below 0.005, the required input directive is as follows:

```
level nr 0.2 0.005 0
```

Note that in both of these examples, the parameters for the PCG method are at the default values. To obtain values different from the defaults, the keyword pcg must also be specified. For example, to specify the level shifting in the above example for the exact Hessian and non-default shifting for the PCG method, the directive would be something like the following:

```
level pcg 20 0.3 0.0 nr 0.2 0.005 0.0
```

This input will cause the PCG method to be level-shifted by 20.0 until the maximum element of the gradient falls below 0.3, then the shift will be zero. For the exact Hessian, the level shifting is initially 0.2, until the maximum element falls below 0.005, after which the shift is zero.

The default options correspond to

```
level pcg 20 0.5 0 nr 0 0 0
```

11.1.16 Orbital Localization

The SCF module includes an experimental implementation of orbital localization, including Foster-Boys and Pipek-Mezey which only works for closed-shell (RHF) wavefunctions. There is currently no input in the SCF block to control this so the SET directive (SET) must be used.

The directive

```
set scf:localize t
```

will separately localize the core, valence, and virtual orbital spaces using the Pipek-Mezey algorithm. If the additional directive

```
set scf:loctype FB
```

is included, then the Foster-boys algorithm is used. The partitioning of core-orbitals is performed using the atomic information described in the section describing how to freeze the orbitals .

In the next release, this functionality will be extended to included all wavefunctions using molecular orbitals.

11.1.17 Printing Information from the SCF Module

All output from the SCF module is controlled using the PRINT directive described in Print control. The following list describes the items from SCF that are currently under direct print control, along with the print level for each one.

Name	Print Level	Description
“atomic guess density”	debug	guess density matrix
“atomic scf”	debug	details of atomic SCF
“mo guess”	default	brief info from mo guess
“information”	low	results
“initial vectors”	debug	
“intermediate vectors”	debug	
“final vectors”	debug	
“final vectors analysis”	default	
“initial vectors analysis”	never	
“intermediate evals”	debug	
“final evals”	default	
“schwarz”	high	integral screening info stats at completion
“screening statistics”	debug	display stats after every Fock build
“geometry”	high	
“symmetry”	debug	detailed symmetry info
“basis”	high	
“geombas”	debug	detailed basis map info
“vectors i/o”	default	report vectors I/O
“parameters”	default	convergence parameters
“convergence”	default	info each iteration
“mulliken ao”	never	Mulliken population of basis functions

Table 2: SCF Print Control Specifications

11.1.18 Hartree-Fock or SCF, MCSCF and MP2 Gradients

The input for this directive allows the user to adjust the print control for the SCF, UHF, ROHF, MCSCF and MP2 gradients. The form of the directive is as follows:

```
GRADIENTS
  [print || noprint] ...
END
```

The complementary keyword pair print and noprint allows the user some additional control on the information that can be included in the print output from the SCF calculation. Currently, only a few items can be explicitly invoked via print control. These are as follows:

Name	Print Level	Description
“information”	low	calculation info
“geometry”	high	geometry information
“basis”	high	basis set(s) used
“forces”	low	details of force components
“timing”	default	timing for each phase

Table 3: Gradient Print Control Specifications

11.1.19 References

1. Wong, A. T. and Harrison, R. J. (1995) “Approaches to large-scale parallel self-consistent field calculation”, *J. Comp. Chem.* **16**, 1291-1300, DOI: 10.1002/jcc.540161010
2. Foster, I. T.; Tilson, J. L.; Wagner, A. F.; Shepard, R. L.; Harrison, R. J.; Kendall, R. A. and Littlefield, R. J. (1996) “Toward high-performance computational chemistry: I. Scalable Fock matrix construction algorithms”, *J. Comp. Chem.* **17**, 109-123, DOI: 10.1002/(SICI)1096-987X(19960115)17:1<109::AID-JCC9>3.0.CO;2-V

11.2 Density Functional Theory (DFT)

11.2.1 Overview

The NWChem density functional theory (DFT) module uses the Gaussian basis set approach to compute closed shell and open shell densities and Kohn-Sham orbitals in the:

- local density approximation (LDA),
- non-local density approximation (NLDA),
- local spin-density approximation (LSD),
- non-local spin-density approximation (NLSD),
- non-local meta-GGA approximation (metaGGA),
- any empirical mixture of local and non-local approximations (including exact exchange), and
- asymptotically corrected exchange-correlation potentials.
- spin-orbit effects

The formal scaling of the DFT computation can be reduced by choosing to use auxiliary Gaussian basis sets to fit the charge density (CD) and/or fit the exchange-correlation (XC) potential.

DFT input is provided using the compound DFT directive

```
DFT
  ...
END
```

The actual DFT calculation will be performed when the input module encounters theTASK directive.

```
TASK DFT
```

Once a user has specified a geometry and a Kohn-Sham orbital basis set the DFT module can be invoked with no input directives (defaults invoked throughout). There are sub-directives which allow for customized application; those currently provided as options for the DFT module are:

```
VECTORS [[input] (<string input_movecs default atomic>) || \
                (project <string basisname> <string filename>)] \
        [swap [alpha||beta] <integer vec1 vec2> ...] \
        [output <string output_filename default input_movecs>] \
XC [[acm] [b3lyp] [beckehandh] [pbe0]\
   [becke97]  [becke97-1] [becke97-2] [becke97-3] [becke97-d] [becke98] \
    [hcth] [hcth120] [hcth147] [hcth147@tz2p]\
    [hcth407] [becke97gga1]  [hcth407p]\
    [mpw91] [mpw1k] [xft97] [cft97] [ft97] [op] [bop] [pbeop]\
    [xpkzb99] [cpkzb99] [xtpss03] [ctpss03] [xctpssh]\
    [b1b95] [bb1k] [mpw1b95] [mpwb1k] [pw6b95] [pwb6k] [m05] [m05-2x] [vs98] \
    [m06] [m06-hf] [m06-L] [m06-2x] \
    [HFexch <real prefactor default 1.0>] \
    [becke88 [nonlocal] <real prefactor default 1.0>] \
    [xperdew91 [nonlocal] <real prefactor default 1.0>] \
    [xpbe96 [nonlocal] <real prefactor default 1.0>] \
    [gill96 [nonlocal] <real prefactor default 1.0>] \
    [lyp <real prefactor default 1.0>] \
    [perdew81 <real prefactor default 1.0>] \
    [perdew86 [nonlocal] <real prefactor default 1.0>] \
    [perdew91 [nonlocal] <real prefactor default 1.0>] \
    [cpbe96 [nonlocal] <real prefactor default 1.0>] \
    [pw91lda <real prefactor default 1.0>] \
    [slater <real prefactor default 1.0>] \
    [vwn_1 <real prefactor default 1.0>] \
    [vwn_2 <real prefactor default 1.0>] \
    [vwn_3 <real prefactor default 1.0>] \
    [vwn_4 <real prefactor default 1.0>] \
    [vwn_5 <real prefactor default 1.0>] \
    [vwn_1_rpa <real prefactor default 1.0>] \
    [xtpss03 [nonlocal] <real prefactor default 1.0>] \
    [ctpss03 [nonlocal] <real prefactor default 1.0>] \
    [bc95 [nonlocal] <real prefactor default 1.0>] \
    [xpw6b95 [nonlocal] <real prefactor default 1.0>] \
    [xpwb6k [nonlocal] <real prefactor default 1.0>] \
    [xm05 [nonlocal] <real prefactor default 1.0>] \
    [xm05-2x [nonlocal] <real prefactor default 1.0>] \
    [cpw6b95 [nonlocal] <real prefactor default 1.0>] \
    [cpwb6k [nonlocal] <real prefactor default 1.0>] \
    [cm05 [nonlocal] <real prefactor default 1.0>] \
    [cm05-2x [nonlocal] <real prefactor default 1.0>]] \
    [xvs98 [nonlocal] <real prefactor default 1.0>]] \
    [cvs98 [nonlocal] <real prefactor default 1.0>]] \
    [xm06-L [nonlocal] <real prefactor default 1.0>]] \
    [xm06-hf [nonlocal] <real prefactor default 1.0>]] \
    [xm06 [nonlocal] <real prefactor default 1.0>]] \
    [xm06-2x [nonlocal] <real prefactor default 1.0>]] \
    [cm06-L [nonlocal] <real prefactor default 1.0>]] \
    [cm06-hf [nonlocal] <real prefactor default 1.0>]] \
    [cm06 [nonlocal] <real prefactor default 1.0>]] \
    [cm06-2x [nonlocal] <real prefactor default 1.0>]]
CONVERGENCE [[energy <real energy default 1e-7>] \
            [density <real density default 1e-5>] \
            [gradient <real gradient default 5e-4>] \
            [dampon <real dampon default 0.0>] \
            [dampoff <real dampoff default 0.0>] \
            [diison <real diison default 0.0>] \
            [diisoff <real diisoff default 0.0>] \
            [levlon <real levlon default 0.0>] \
            [levloff <real levloff default 0.0>] \
            [ncydp <integer ncydp default 2>] \
            [ncyds <integer ncyds default 30>] \
            [ncysh <integer ncysh default 30>] \
            [damps <integer npdamps default 0> ] [no damping] \
```

```
        [damp <integer ndamp default 0>] [nodamping] \
        [diis [nfock <integer nfock default 10>]] \
        [nodiis] [lshift <real lshift default 0.5>] \
        [nolevelshifting] \
        [hl_tol <real hl_tol default 0.01>] \
        [rabuck [n_rabuck <integer n_rabuck default 25>]\
        [fast] ]
GRID [(xcoarse||coarse||medium||fine||xfine||huge) default medium] \
   [(gausleg||lebedev ) default lebedev ] \
   [(becke||erf1||erf2||ssf) default erf1] \
   [(euler||mura||treutler) default mura] \
   [rm <real rm default 2.0>] \
   [nodisk]
TOLERANCES [[tight] [tol_rho <real tol_rho default 1e-10>] \
        [accCoul <integer accCoul default 8>] \
        [radius <real radius default 25.0>]]
[(LB94||CS00 <real shift default none>)]
DECOMP
ODFT
DIRECT
SEMIDIRECT [filesize <integer filesize default disksize>]
        [memsize  <integer memsize default available>]
        [filename <string filename default $file_prefix.aoints$>]
INCORE
ITERATIONS <integer iterations default 30>
MAX_OVL
IMOM
CGMIN
RODFT
MULLIKEN
DISP
XDM [ a1 <real a1> ]   [ a2 <real a2> ]
MULT <integer mult default 1>
NOIO
PRINT||NOPRINT
SYM <string (ON||OFF) default ON>
ADAPT <string (ON||OFF) default ON>
```

The following sections describe these keywords and optional sub-directives that can be specified for a DFT calculation in NWChem.

11.2.2 Specification of Basis Sets for the DFT Module

The DFT module requires at a minimum the basis set for the Kohn-Sham molecular orbitals. This basis set must be in the default basis set named “ao basis”, or it must be assigned to this default name using the SET directive.

In addition to the basis set for the Kohn-Sham orbitals, the charge density fitting basis set can also be specified in the input directives for the DFT module. This basis set is used for the evaluation of the Coulomb potential in the Dunlap scheme[1,2]. The charge density fitting basis set must have the name `cd basis`. This can be the actual name of a basis set, or a basis set can be assigned this name using the SET directive. If this basis set is not defined by input, the $O(N^4)$ exact Coulomb contribution is computed.

The user also has the option of specifying a third basis set for the evaluation of the exchange-correlation potential. This basis set must have the name `xc basis`. If this basis set is not specified by input, the exchange contribution (XC) is evaluated by numerical quadrature. In most applications, this approach is efficient enough, so the “xc basis” basis set is not required.

For the DFT module, the input options for defining the basis sets in a given calculation can be summarized as follows:

- `ao basis` - Kohn-Sham molecular orbitals; required for all calculations
- `cd basis` - charge density fitting basis set; optional, but recommended for evaluation of the Coulomb potential
- “xc basis” - exchange-correlation (XC) fitting basis set; optional, and not recommended

11.2.3 ADFT *New in NWChem 7.2.0*:

Use of the auxiliary density functional theory method (ADFT)[3,4] can be triggered by means of the `adft` keyword. This can result in a large speed-up when using “pure” GGA functionals (e.g. PBE96) and Laplacian-dependent mGGA functionals (e.g. SCAN-L). The speed-up comes from the use of the fitted density obtained with the charge density fitting technique to

approximate both the Coulomb and Exchange-Correlation contributions.

The ADFT method is similar in spirit to the exchange-correlation fitting technique triggered by specifying an xc basis without the adft keyword. It is important to note that, different to straight exchange-correlation fitting, *energy derivatives are well-defined* within the ADFT framework. As a consequence, geometry optimizations and harmonic vibrational frequencies are well-behaved.

The ADFT method requires a charge density fitting basis set (see DFT basis set section). If not cd basis set is provided, the weigend coulomb fitting basis set will be loaded.

11.2.4 VECTORS and MAX_OVL/IMOM: KS-MO Vectors

The VECTORS directive is the same as that in the SCF module. Currently, the LOCK keyword is not supported by the DFT module, however the directive

```
MAX_OVL
```

has the same effect.
Instead, the IMOM keyword applies the maximum overlap condition with respect to the initial SCF orbitals[5] (while max_ovl [6,7,8] uses the orbitals from the previous cycle).

11.2.5 XC and DECOMP: Exchange-Correlation Potentials

```
XC [[acm] [b3lyp] [beckehandh] [pbe0] [bhlyp]\
  [becke97]  [becke97-1] [becke97-2] [becke97-3] [becke98] [hcth] [hcth120] [hcth147] [hcth147@tz2p] \
   [hcth407] [becke97gga1] [hcth407p] \
   [optx] [hcthp14] [mpw91] [mpw1k] [xft97] [cft97] [ft97] [op] [bop] [pbeop]\
   [m05] [m05-2x] [m06] [m06-l] [m06-2x] [m06-hf] [m08-hx] [m08-so] [m11] [m11-l]\
   [HFexch <real prefactor default 1.0>] \
   [becke88 [nonlocal] <real prefactor default 1.0>] \
   [xperdew91 [nonlocal] <real prefactor default 1.0>] \
   [xpbe96 [nonlocal] <real prefactor default 1.0>] \
   [gill96 [nonlocal] <real prefactor default 1.0>] \
   [lyp <real prefactor default 1.0>] \
   [perdew81 <real prefactor default 1.0>] \
   [perdew86 [nonlocal] <real prefactor default 1.0>] \
   [perdew91 [nonlocal] <real prefactor default 1.0>] \
   [cpbe96 [nonlocal] <real prefactor default 1.0>] \
   [pw91lda <real prefactor default 1.0>] \
   [slater <real prefactor default 1.0>] \
   [vwn_1 <real prefactor default 1.0>] \
   [vwn_2 <real prefactor default 1.0>] \
   [vwn_3 <real prefactor default 1.0>] \
   [vwn_4 <real prefactor default 1.0>] \
   [vwn_5 <real prefactor default 1.0>] \
   [vwn_1_rpa <real prefactor default 1.0>]]
```

The user has the option of specifying the exchange-correlation treatment in the DFT Module (see table below for full list of functionals). The default exchange-correlation functional is defined as the local density approximation (LDA) for closed shell systems and its counterpart the local spin-density (LSD) approximation for open shell systems. Within this approximation, the exchange functional is the Slater $\rho^{1/3}$ functional[9,10], and the correlation functional is the Vosko-Wilk-Nusair (VWN) functional (functional V)[11]. The parameters used in this formula are obtained by fitting to the Ceperley and Alder Quantum Monte-Carlo solution of the homogeneous electron gas.

These defaults can be invoked explicitly by specifying the following keywords within the DFT module input directive, XC slater vwn_5 .

That is, this statement in the input file

```
dft
 XC slater vwn_5
end
task dft
```

is equivalent to the simple line

```
task dft
```

The DECOMP directive causes the components of the energy corresponding to each functional to be printed, rather than just the total exchange-correlation energy that is the default. You can see an example of this directive in the sample input.

Many alternative exchange and correlation functionals are available to the user as listed in the table below. The following sections describe how to use these options.

11.2.5.1 Libxc interface ***New in NWChem 7.2.0***:

If NWChem is compiled by linking it with thelibxc DFT library (as described in theInterfaces with External Software section), the user will be able to use most of the XC functionals available in libxc.
The input syntax requires to use the xc keyword followed by the functionals name from list available in Libxc

For example, the following input for the NWChem libxc interface

```
dft
 xc gga_x_pbe 1.0 gga_x_pbe 1.0
end
```

while trigger use of the same PBE96 functionals as in the NWChem built-in interface

```
dft
 xc xpbe96 1.0 cpbe96 1.0
end
```

11.2.5.2 Exchange-Correlation Functionals

There are several Exchange and Correlation functionals in addition to the defaultslater and vwn_5 functionals. These are either local or gradient-corrected functionals (GCA); a full list can be found in the table below.

The Hartree-Fock exact exchange functional, (which has $O(N^4)$ computation expense), is invoked by specifying

```
XC HFexch
```

Note that the user also has the ability to include only the local or nonlocal contributions of a given functional. In addition, the user can specify a multiplicative prefactor (the variable in the input) for the local/nonlocal component or total. An example of this might be,

```
XC becke88 nonlocal 0.72
```

The user should be aware that the Becke88 local component is simply the Slater exchange and should be input as such.

Any combination of the supported exchange functional options can be used. For example, the popular Gaussian B3 exchange could be specified as:

```
XC slater 0.8 becke88 nonlocal 0.72 HFexch 0.2
```

Any combination of the supported correlation functional options can be used. For example, B3LYP could be specified as:

```
XC vwn_1_rpa 0.19 lyp 0.81 HFexch 0.20  slater 0.80 becke88 nonlocal 0.72
```

and X3LYP as:

```
xc vwn_1_rpa 0.129 lyp 0.871 hfexch 0.218 slater 0.782 \
becke88 nonlocal 0.542  xperdew91 nonlocal 0.167
```

11.2.5.3 Setting up common exchange-correlation functionals

- B3LYP: `xc b3lyp`
- PBE0: `xc pbe0`
- PBE96: `xc xpbe96 cpbe96`
- PW91: `xc xperdew91 perdew91`
- BHLYP: `xc bhlyp`
- Becke Half and Half: `xc beckehandh`
- BP86: `xc becke88 perdew86`
- BP91: `xc becke88 perdew91`
- BLYP: `xc becke88 lyp`

Minnesota Functionals

- xc m05
- xc m05-2x
- xc m06
- xc m06-l
- xc m06-2x
- xc m06-hf
- xc m08-hx
- xc m08-so
- xc m11
- xc m11-l

Analytic second derivatives are not supported with the Minnesota functionals yet.

11.2.5.4 Combined Exchange and Correlation Functionals

In addition to the options listed above for the exchange and correlation functionals, the user has the alternative of specifying combined exchange and correlation functionals.

The available hybrid functionals (where a Hartree-Fock Exchange component is present) consist of the Becke “half and half”[12], the adiabatic connection method[13], Becke 1997 (“Becke V” paper[14]).

The keyword `beckehandh` specifies that the exchange-correlation energy will be computed as

$E_{XC} \approx ½E_X^{HF} + ½E_X^{Slater} + ½E_C^{PW91LDA}$

We know this is NOT the correct Becke prescribed implementation that requires the XC potential in the energy expression. But this is what is currently implemented as an approximation to it.

The keyword `acm` specifies that the exchange-correlation energy is computed as

$E_{XC} = a_0E_X^{HF} + (1 - a_0)E_X^{Slater} + a_X\delta E_X^{Becke88} + E_C^{VWN} + a_C\delta E_C^{Perdew91}$

where

$a_0 = 0.20$, $a_X = 0.72$, $a_C = 0.81$

and δ stands for a non-local component.

The keyword b3lyp specifies that the exchange-correlation energy is computed as

$E_{XC} = a_0 E_X^{HF} + (1 - a_0) E_X^{Slater} + a_X \delta E_X^{Becke88} + (1 - a_C) E_C^{VWN_1_RPA} + a_C \delta E_C^{LYP}$

where

$a_0 = 0.20$, $a_X = 0.72$, $a_C = 0.81$

11.2.6 XC Functionals Summary

Keyword	X	C	GGA	Meta	Hybr.	2nd	Ref.
slater	*					Y	910
vwn_1		*				Y	11
vwn_2		*				Y	11
vwn_3		*				Y	11
vwn_4		*				Y	11
vwn_5		*				Y	11
vwn_1_rpa		*				Y	11
perdew81		*				Y	15
pw91lda		*				Y	16
xbecke86b	*		*			N	17
becke88	*		*			Y	18
xperdew86	*		*			N	19
xperdew91	*		*			Y	16
xpbe96	*		*			Y	2021
gill96	*		*			Y	22
optx	*		*			N	23
mpw91	*		*			Y	2425
xft97	*		*			N	2627
rpbe	*		*			Y	28
revpbe	*		*			Y	29
xpw6b95	*		*			N	30
xpwb6k	*		*			N	30
perdew86		*	*			Y	19
lyp		*	*			Y	31

Keyword	X	C	GGA	Meta	Hybr.	2nd	Ref.
perdew91		*	*			Y	[3233]
cpbe96		*	*			Y	[2021]
cft97		*	*			N	[2627]
op		*	*			N	[34]
hcth	*	*	*			N	[35]
hcth120	*	*	*			N	[36]
hcth147	*	*	*			N	[36]
hcth147@tz2p	*	*	*			N	[37]
hcth407	*	*	*			N	[38]
becke97gga1	*	*	*			N	[39]
hcthp14	*	*	*			N	[40]
ft97	*	*	*			N	[2627]
htch407p	*	*	*			N	[41]
bop	*	*	*			N	[34]
pbeop	*	*	*			N	[42]
xpkzb99	*			*		N	[43]
cpkzb99		*		*		N	[43]
xtpss03	*			*		N	[44]
ctpss03		*		*		N	[44]
bc95		*		*		N	[28]
cpw6b95		*		*		N	[30]
cpwb6k		*		*		N	[30]
xm05	*			*	*	N	[4546]
cm05		*		*		N	[4546]
m05-2x	*	*		*	*	N	[47]
xm05-2x	*			*	*	N	[47]
cm05-2x		*		*		N	[47]
xctpssh				*	*	N	[48]
bb1k				*	*	N	[29]
mpw1b95				*	*	N	[49]

Keyword	X	C	GGA	Meta	Hybr.	2nd	Ref.
mpwb1k				*	*	N	[49]
pw6b95				*	*	N	[30]
pwb6k				*	*	N	[30]
m05				*	*	N	[45]
vs98				*	*	N	[50]
xvs98	*			*		N	[50]
cvs98		*		*		N	[50]
m06-L	*	*		*		N	[51]
xm06-L	*			*		N	[51]
cm06-L		*		*		N	[51]
m06-hf				*	*	N	[52]
xm06-hf	*			*	*	N	[52]
cm06-hf		*		*		N	[52]
m06				*	*	N	[53]
xm06	*			*	*	N	[53]
cm06		*		*		N	[53]
m06-2x				*	*	N	[51]
xm06-2x	*			*	*	N	[51]
cm06-2x		*		*		N	[51]
cm08-hx		*		*		N	[54]
xm08-hx	*			*		N	[54]
m08-hx	*	*		*	*	N	[54]
cm08-so		*		*		N	[54]
xm08-so	*			*		N	[54]
m08-so	*	*		*	*	N	[54]
cm11		*		*		N	[55]
xm11	*			*		N	[55]
m11	*	*		*	*	N	[55]
cm11-l		*		*		N	[56]
xm11-l	*			*		N	[56]

Keyword	X	C	GGA	Meta	Hybr.	2nd	Ref.
m11-l	*	*		*		N	[56]
csogga		*	*			N	[54]
xsogga	*		*			N	[54]
sogga	*	*	*			N	[54]
csogga11		*	*			N	[57]
xsogga11	*		*			N	[57]
sogga11	*	*	*			N	[57]
csogga11-x		*				N	[58]
xsogga11-x	*		*			N	[58]
sogga11-x	*	*	*		*	N	[58]
dldf	*	*		*	*	N	[59]
beckehandh	*	*			*	Y	[12]
b3lyp	*	*	*		*	Y	[13]
acm	*	*	*		*	Y	[13]
becke97	*	*	*		*	N	[14]
becke97-1	*	*	*		*	N	[35]
becke97-2	*	*	*		*	N	[60]
becke97-3	*	*	*		*	N	[61]
becke97-d	*	*	*		*	N	[62]
becke98	*	*	*		*	N	[63]
pbe0	*	*	*		*	Y	[64]
mpw1k	*	*	*		*	Y	[65]
xmvs15	*			*		N	[66]
hle16	*	*	*		*	Y	[67]
scan	*	*	*	*		N	[68]
scanl	*	*	*	*		N	[69]
revm06-L	*	*	*	*		N	[70]
revm06	*	*	*	*	*	N	[71]
wb97x	*	*	*		*	N	[72]
wb97x-d3	*	*	*		*	N	[73]

Keyword	X	C	GGA	Meta	Hybr.	2nd	Ref.
rscan	*	*	*	*		N	[74]
r2scan	*	*	*	*		N	[75]
r2scan0	*	*	*	*	*	N	[76]
r2scanl	*	*	*	*		N	[77][78]
ncap	*	*	*			Y	[79]

Table 1: Available Exchange (X) and Correlation (C) functionals. GGA is

11.2.6.1 Meta-GGA Functionals

One way to calculate meta-GGA energies is to use orbitals and densities from fully self-consistent GGA or LDA calculations and run them in one iteration in the meta-GGA functional. It is expected that meta-GGA energies obtained this way will be close to fully self consistent meta-GGA calculations.

It is possible to calculate metaGGA energies both ways in NWChem, that is, self-consistently or with GGA/LDA orbitals and densities. However, since second derivatives are not available for metaGGAs, in order to calculate frequencies, one must use task dft freq numerical. A sample file with this is shown below, in Sample input file. In this instance, the energy is calculated self-consistently and geometry is optimized using the analytical gradients.

(For more information on metaGGAs, see Kurth et al 1999[80] for a brief description of meta-GGAs, and citations 14-27 therein for thorough background)

Note: both TPSS and PKZB correlation require the PBE GGA CORRELATION (which is itself dependent on an LDA). The decision has been made to use these functionals with the accompanying local PW91LDA. The user cannot set the local part of these metaGGA functionals.

11.2.6.2 Range-Separated Functionals

Using the Ewald decomposition

$$\frac{1}{r_{12}} = \frac{\alpha + \beta \texttt{erf}(\mu r_{12})}{r_{12}} + \frac{1 - [\alpha + \beta \texttt{erf}(\mu r_{12})]}{r_{12}}$$

we can split the the Exchange interaction as

$$E_{X} = E_X^{LR} + E_X^{SR}$$

Therefore, the long-range HF Exchange energy becomes

$$E_X^{LR} = \alpha E_X^{HF} - \frac{\beta}{2} \sum_i \sum_j \int \int \phi_i(r_1)\phi_j(r_1)\frac{\texttt{erf}(\mu r_{12})}{r_{12}} \phi_i(r_2)\phi_j(r_2)$$

11.2.6.2.1 Input parameters for Range-Separated functionals

```
cam <real cam> cam_alpha <real cam_alpha> cam_beta <cam_beta>
```

`cam` represents the attenuation parameter μ, `cam_alpha` and `cam_beta` are the α and β parameters that control the amount of short-range DFT and long-range HF Exchange according to the Ewald decomposition. As $r_{12} \rightarrow 0$, the HF exchange fraction is α, while the DFT exchange fraction is 1 - α. As $r_{12} \rightarrow \infty$, the HF exchange fraction approaches α + β and the DFT exchange fraction approaches 1 - α - β. In the HSE functional, the HF part is short-ranged and DFT is long-ranged.

Range separated functionals (or long-range corrected or LC) can be specified as follows:

CAM-B3LYP:

```
xc xcamb88 1.00 lyp 0.81 vwn_5 0.19 hfexch 1.00
cam 0.33 cam_alpha 0.19 cam_beta 0.46
```

LC-BLYP:

```
xc xcamb88 1.00 lyp 1.0 hfexch 1.00
cam 0.33 cam_alpha 0.0 cam_beta 1.0
```

LC-PBE:

```
xc xcampbe96 1.0 cpbe96 1.0 HFexch 1.0
cam 0.30 cam_alpha 0.0 cam_beta 1.0
```

LC-PBE0 or CAM-PBE0:

```
xc xcampbe96 1.0 cpbe96 1.0 HFexch 1.0
cam 0.30 cam_alpha 0.25 cam_beta 0.75
```

BNL (Baer, Neuhauser, Lifshifts):

```
xc xbnl07 0.90 lyp 1.00 hfexch 1.00
cam 0.33 cam_alpha 0.0 cam_beta 1.0
```

LC-wPBE:

```
xc xwpbe 1.00 cpbe96 1.0 hfexch 1.00
cam 0.4 cam_alpha 0.00 cam_beta 1.00
```

LRC-wPBEh:

```
xc xwpbe 0.80 cpbe96 1.0 hfexch 1.00
cam 0.2 cam_alpha 0.20 cam_beta 0.80
```

QTP-00

```
xc xcamb88 1.00 lyp 0.80 vwn_5 0.2 hfexch 1.00
cam 0.29 cam_alpha 0.54 cam_beta 0.37
```

rCAM-B3LYP

```
xc xcamb88 1.00 lyp 1.0 vwn_5 0. hfexch 1.00 becke88 nonlocal 0.13590
cam 0.33 cam_alpha 0.18352 cam_beta 0.94979
```

HSE03 functional: 0.25*Ex(HF-SR) - 0.25*Ex(PBE-SR) + Ex(PBE) + Ec(PBE), where gamma(HF-SR) = gamma(PBE-SR)

```
xc hse03
```

or it can be explicitly set as

```
xc xpbe96 1.0 xcampbe96 -0.25 cpbe96 1.0 srhfexch 0.25
cam 0.33 cam_alpha 0.0 cam_beta 1.0
```

HSE06 functional:

```
xc xpbe96 1.0 xcampbe96 -0.25 cpbe96 1.0 srhfexch 0.25
cam 0.11 cam_alpha 0.0 cam_beta 1.0
```

Please see references [81828384858687888990919293] (not a complete list) for further details about the theory behind these functionals and applications.

Example illustrating the CAM-B3LYP functional:

```
start h2o-camb3lyp
geometry units angstrom
  O      0.00000000     0.00000000     0.11726921
  H      0.75698224     0.00000000    -0.46907685
  H     -0.75698224     0.00000000    -0.46907685
end
basis spherical
 * library aug-cc-pvdz
end
dft
 xc xcamb88 1.00 lyp 0.81 vwn_5 0.19 hfexch 1.00
 cam 0.33 cam_alpha 0.19 cam_beta 0.46
 direct
 iterations 100
end
task dft energy
```

Example illustrating the HSE03 functional:

```
echo
start h2o-hse
geometry units angstrom
O      0.00000000     0.00000000     0.11726921
H      0.75698224     0.00000000    -0.46907685
H     -0.75698224     0.00000000    -0.46907685
end
basis spherical
 * library aug-cc-pvdz
end
dft
 xc hse03
 iterations 100
 direct
 end
task dft energy
```

or alternatively

```
dft
 xc xpbe96 1.0 xcampbe96 -0.25 cpbe96 1.0 srhfexch 0.25
 cam 0.33 cam_alpha 0.0 cam_beta 1.0
 iterations 100
 direct
end
task dft energy
```

11.2.6.3 SSB-D functional

The SSB-D[9495] functional is a small correction to the non-empirical PBE functional and includes a portion of Grimme's dispersion correction (s6=0.847455). It is designed to reproduce the good results of OPBE for spin-state splittings and reaction barriers, and the good results of PBE for weak interactions. The SSB-D functional works well for these systems, including for difficult systems for DFT (dimerization of anthracene, branching of octane, water-hexamer isomers, $C_{12}H_{12}$ isomers, stacked adenine dimers), and for NMR chemical shieldings.

It can be specified as

```
xc ssb-d
```

11.2.6.4 Semi-empirical hybrid DFT combined with perturbative MP2

This theory combines hybrid density functional theory with MP2 semi-empirically. The B2PLYP functional, which is an example of this approximation, can be specified as:

```
mp2
 freeze atomic
end
dft
 xc HFexch 0.53 becke88 0.47 lyp 0.73 mp2 0.27
 dftmp2
end
```

For details of the theory, please see reference[96].

11.2.6.5 LB94 and CS00: Asymptotic correction

The keyword LB94 will correct the asymptotic region of the XC definition of exchange-correlation potential by the van-Leeuwen-Baerends exchange-correlation potential that has the correct \(-1/r\) asymptotic behavior. The total energy will be computed by the XC definition of exchange-correlation functional. This scheme is known to tend to overcorrect the deficiency of most uncorrected exchange-correlation potentials.

The keyword CS00, when supplied with a real value of shift (in atomic units), will perform Casida-Salahub '00 asymptotic correction. This is primarily intended for use with TDDFT. The shift is normally positive (which means that the original uncorrected exchange-correlation potential must be shifted down).

When the keyword CS00 is specified without the value of shift, the program will automatically supply it according to the semi-empirical formula of Zhan, Nichols, and Dixon (again, see TDDFT for more details and references). As the Zhan's formula is calibrated against B3LYP results, it is most meaningful to use this with the B3LYP functional, although the program does not prohibit (or even warn) the use of any other functional.

Sample input files of asymptotically corrected TDDFT calculations can be found in the corresponding section.

11.2.7 Sample input file

A simple example calculates the geometry of water, using the metaGGA functionals xtpss03 and ctpss03. This also highlights some of the print features in the DFT module. Note that you must use the line task dft freq numerical because analytic hessians are not available for the metaGGAs:

```
title "WATER 6-311G* meta-GGA XC geometry"
echo
geometry units angstroms
 O      0.0  0.0  0.0
 H      0.0  0.0  1.0
 H      0.0  1.0  0.0
end
basis
 H library 6-311G*
 O library 6-311G*
end
dft
 iterations 50
 print  kinetic_energy
 xc xtpss03 ctpss03
 decomp
end
task dft optimize
task dft freq numerical
```

11.2.8 ITERATIONS or MAXITER: Number of SCF iterations

```
ITERATIONS or MAXITER <integer iterations default 30>
```

The default optimization in the DFT module is to iterate on the Kohn-Sham (SCF) equations for a specified number of iterations (default 30). The keyword that controls this optimization is ITERATIONS, and has the following general form,

```
iterations <integer iterations default 30>
```

or

```
maxiter <integer iterations default 30>
```

The optimization procedure will stop when the specified number of iterations is reached or convergence is met. See an example that uses this directive in Sample input file.

11.2.9 CONVERGENCE: SCF Convergence Control

```
CONVERGENCE [energy <real energy default 1e-6>] \
        [density <real density default 1e-5>] \
        [gradient <real gradient default 5e-4>] \
        [hl_tol <real hl_tol default 0.01>]
        [dampon <real dampon default 0.0>] \
        [dampoff <real dampoff default 0.0>] \
        [ncydp <integer ncydp default 2>] \
        [ncyds <integer ncyds default 30>] \
        [ncysh <integer ncysh default 30>] \
        [damp <integer ndamp default 0>] [nodamping] \
        [diison <real diison default 0.0>] \
        [diisoff <real diisoff default 0.0>] \
        [(diis [nfock <integer nfock default 10>]) || nodiis] \
        [levlon <real levlon default 0.0>] \
        [levloff <real levloff default 0.0>] \
        [(lshift <real lshift default 0.5>) || nolevelshifting] \
        [rabuck [n_rabuck <integer n_rabuck default 25>] \
        [fast] ]
```

Convergence is satisfied by meeting any or all of three criteria;

- convergence of the total energy; this is defined to be when the total DFT energy at iteration N and at iteration N-1 differ by a value less than a threshold value (the default is 1e-6). This value can be modified using the key word,

  ```
  CONVERGENCE energy <real energy default 1e-6>
  ```

- convergence of the total density; this is defined to be when the total DFT density matrix at iteration N and at iteration N-1 have a RMS difference less than some value (the default is 1e-5). This value can be modified using the keyword,

  ```
  CONVERGENCE density <real density default 1e-5>
  ```

- convergence of the orbital gradient; this is defined to be when the DIIS error vector becomes less than some value (the default is 5e-4). This value can be modified using the keyword,

  ```
  CONVERGENCE gradient <real gradient default 5e-4>
  ```

The default optimization strategy is to immediately begin direct inversion of the iterative subspace. In addition, if the HOMO - LUMO gap is small and the Fock matrix diagonally dominant, then level-shifting is automatically initiated. There are a variety of ways to customize this procedure to whatever is desired.

An alternative optimization strategy is to specify, by using the change in total energy (between iterations *N* and *N-1*), when to turn damping, level-shifting, and/or DIIS on/off. Start and stop keywords for each of these is available as,

```
CONVERGENCE  [dampon <real dampon default 0.0>] \
        [dampoff <real dampoff default 0.0>] \
        [diison <real diison default 0.0>] \
        [diisoff <real diisoff default 0.0>] \
        [levlon <real levlon default 0.0>] \
        [levloff <real levloff default 0.0>]
```

So, for example, damping, DIIS, and/or level-shifting can be turned on/off as desired.

Another strategy can be to specify how many iterations (cycles) you wish each type of procedure to be used. The necessary keywords to control the number of damping cycles (ncydp), the number of DIIS cycles (ncyds), and the number of level-shifting cycles (ncysh) are input as,

```
CONVERGENCE  [ncydp <integer ncydp default 2>] \
        [ncyds <integer ncyds default 30>] \
        [ncysh <integer ncysh default 0>]
```

The amount of damping, level-shifting, time at which level-shifting is automatically imposed, and Fock matrices used in the DIIS extrapolation can be modified by the following keywords

```
CONVERGENCE  [damp <integer ndamp default 0>] \
        [diis [nfock <integer nfock default 10>]] \
        [lshift <real lshift default 0.5>] \
        [hl_tol <real hl_tol default 0.01>]]
```

11.2.9.1 CONVERGENCE DAMP Keyword

Damping is defined to be the percentage of the previous iterations density mixed with the current iterations density. So, for example

```
CONVERGENCE damp 70
```

would mix 30% of the current iteration density with 70% of the previous iteration density.

11.2.9.2 CONVERGENCE LSHIFT Keyword

Level-Shifting is defined as the amount of shift applied to the diagonal elements of the unoccupied block of the Fock matrix. The shift is specified by the keyword `lshift` . For example the directive,

```
CONVERGENCE lshift 0.5
```

causes the diagonal elements of the Fock matrix corresponding to the virtual orbitals to be shifted by 0.5 a.u. By default, this level-shifting procedure is switched on whenever the HOMO-LUMO gap is small. Small is defined by default to be 0.01 au but can be modified by the directive hl_tol. An example of changing the HOMO-LUMO gap tolerance to 0.005 would be,

```
CONVERGENCE hl_tol 0.005
```

11.2.9.3 CONVERGENCE DIIS Keyword

Direct inversion of the iterative subspace with extrapolation of up to 10 Fock matrices is a default optimization procedure. For large molecular systems the amount of available memory may preclude the ability to store this number of N^2 arrays in global memory. The user may then specify the number of Fock matrices to be used in the extrapolation (must be greater than three (3) to be effective). To set the number of Fock matrices stored and used in the extrapolation procedure to 3 would take the form,

```
CONVERGENCE diis 3
```

The user has the ability to simply turn off any optimization procedures deemed undesirable with the obvious keywords,

```
CONVERGENCE [nodamping] [nodiis] [nolevelshifting]
```

For systems where the initial guess is very poor, the user can try using fractional occupation of the orbital levels during the initial cycles of the SCF convergence[97]. The input has the following form

```
CONVERGENCE rabuck [n_rabuck <integer n_rabuck default 25>]]
```

where the optional value `n_rabuck` determines the number of SCF cycles during which the method will be active. For example, to set equal to 30 the number of cycles where the Rabuck method is active, you need to use the following line

```
CONVERGENCE rabuck 30
```

11.2.9.4 CONVERGENCE FAST Keyword

The `convergence fast` option turns on a series of parameters that most often speed-up convergence, but not in 100% of the cases.

```
CONVERGENCE fast
```

Here is an input snippet that would give you the same result as convergence fast

```
dft
 convergence lshift 0. ncydp 0 dampon 1d99 dampoff 1d-4 damp 40
end
set quickguess t
task dft
```

11.2.10 CDFT: Constrained DFT

This option enables the constrained DFT formalism by Wu and Van Voorhis[98]:

```
CDFT <integer fatom1 latom1> [<integer fatom2 latom2>] (charge||spin <real constaint_value>) \
   [pop (becke||mulliken||lowdin) default lowdin]
```

Variables fatom1 and latom1 define the first and last atom of the group of atoms to which the constraint will be applied. Therefore, the atoms in the same group should be placed continuously in the geometry input. If fatom2 and latom2 are specified, the difference between group 1 and 2 (i.e. 1-2) is constrained.

The constraint can be either on the charge or the spin density (number of alpha - beta electrons) with a user specified constraint_value . Note: No gradients have been implemented for the spin constraints case. Geometry optimizations can only be performed using the charge constraint.

To calculate the charge or spin density, the Becke, Mulliken, and Lowdin population schemes can be used. The Lowdin scheme is default while the Mulliken scheme is not recommended. If basis sets with many diffuse functions are used, the Becke population scheme is recommended.

Multiple constraints can be defined simultaniously by defining multiple cdft lines in the input. The same population scheme will be used for all constraints and only needs to be specified once. If multiple population options are defined, the last one will be used. When there are convergence problems with multiple constraints, the user is advised to do one constraint first and to use the resulting orbitals for the next step of the constrained calculations.

It is best to put convergence nolevelshifting in the dft directive to avoid issues with gradient calculations and convergence in CDFT. Use orbital swap to get a broken-symmetry solution.

An input example is given below.

```
geometry
symmetry
 C  0.0  0.0  0.0
 O  1.2  0.0  0.0
 C  0.0  0.0  2.0
 O  1.2  0.0  2.0
end
basis
 * library 6-31G*
end
dft
 xc b3lyp
 convergence nolevelshifting
 odft
 mult 1
 vectors swap beta 14 15
 cdft 1 2 charge 1.0
end
task dft
```

11.2.11 SMEAR: Fractional Occupation of the Molecular Orbitals

The SMEAR keyword is useful in cases with many degenerate states near the HOMO (eg metallic clusters)

```
SMEAR <real smear default 0.001>
```

This option allows fractional occupation of the molecular orbitals. A Gaussian broadening function of exponent smear is used as described in the paper by Warren and Dunlap[99]. The user must be aware that an additional energy term is added to the total energy in order to have energies and gradients consistent.

11.2.12 FON: Calculations with fractional numbers of electrons

11.2.12.1 Restricted

```
fon partial 3 electrons 1.8 filled 2
```

Here 1.8 electrons will be equally divided over 3 valence orbitals and 2 orbitals are fully filled. The total number of electrons here is 5.8

Example input:

```
echo
title "carbon atom"
start carbon_fon
geometry
symmetry c1
  C   0.0 0.0 0.0
end
basis
 * library 6-31G
end
dft
 direct
 grid xfine
 convergence energy 1d-8
 xc pbe0
 fon partial 3 electrons 1.8 filled 2
end
task dft energy
```

11.2.12.2 Unrestricted

```
fon alpha partial 3 electrons 0.9 filled 2
fon beta partial 3 electrons 0.9 filled 2
```

Here 0.9 electrons will be equally divided over 3 alpha valence orbitals and 2 alpha orbitals are fully filled. Similarly for beta. The total number of electrons here is 5.8

Example input:

```
echo
title "carbon atom"
start carbon_fon
geometry
 C   0.0 0.0 0.0
end
basis
 * library 6-31G
end
dft
 odft
 fon alpha partial 3 electrons 0.9 filled 2
 fon beta partial 3 electrons 0.9 filled 2
end
task dft energy
```

To set fractional numbers in the core orbitals, add the following directive in the input file:

```
set dft:core_fon .true.
```

Example input:

```
dft
 print "final vectors analysis"
 odft
 direct
 fon alpha partial 2 electrons 1.0 filled 2
 fon beta partial 2 electrons 1.0 filled 2
 xc pbe0
 convergence energy 1d-8
end
task dft
```

11.2.13 OCCUP: Controlling the occupations of molecular orbitals

Example:

```
echo
start h2o_core_hole
memory 1000 mb
geometry units au
 O 0       0        0
 H 0       1.430   -1.107
 H 0      -1.430   -1.107
end
basis
 O library 6-31g*
 H library 6-31g*
end
occup
 6 6      # occupation list for 6 alpha and 6 beta orbitals
 1.0 0.0  # core-hole in the first beta orbital
 1.0 1.0
 1.0 1.0
 1.0 1.0
 1.0 1.0
 0.0 0.0
end
dft
 odft
 mult 1
 xc beckehandh
end
task dft
```

11.2.14 GRID: Numerical Integration of the XC Potential

```
GRID [(xcoarse||coarse||medium||fine||xfine||huge) default medium] \
    [(gausleg||lebedev ) default lebedev ] \
    [(becke||erf1||erf2||ssf) default erf1] \
    [(euler||mura||treutler) default mura] \
    [rm <real rm default 2.0>] \
    [nodisk]
```

A numerical integration is necessary for the evaluation of the exchange-correlation contribution to the density functional. The default quadrature used for the numerical integration is an Euler-MacLaurin scheme for the radial components (with a modified Mura-Knowles transformation) and a Lebedev scheme for the angular components. Within this numerical integration procedure various levels of accuracy have been defined and are available to the user. The user can specify the level of accuracy with the keywords; xcoarse , coarse , medium , fine , xfine and huge . The default is medium .

```
GRID [xcoarse||coarse||medium||fine||xfine||huge]
```

Our intent is to have a numerical integration scheme which would give us approximately the accuracy defined below regardless of molecular composition.

Keyword	Total Energy Target Accuracy
xcoarse	$1 \cdot 10^{-4}$
coarse	$1 \cdot 10^{-5}$
medium	$1 \cdot 10^{-6}$
fine	$1 \cdot 10^{-7}$
xfine	$1 \cdot 10^{-8}$
huge	$1 \cdot 10^{-10}$

In order to determine the level of radial and angular quadrature needed to give us the target accuracy, we computed total DFT energies at the LDA level of theory for many homonuclear atomic, diatomic and triatomic systems in rows 1-4 of the periodic table. In each case all bond lengths were set to twice the Bragg-Slater radius. The total DFT energy of the system was computed using the converged SCF density with atoms having radial shells ranging from 35-235 (at fixed 48/96 angular quadratures) and angular quadratures of 12/24-48/96 (at fixed 235 radial shells). The error of the numerical integration was determined by comparison to a “best” or most accurate calculation in which a grid of 235 radial points 48 theta and 96 phi angular points on each atom was used. This corresponds to approximately 1 million points per atom. The following tables were empirically determined to give the desired target accuracy for DFT total energies. These tables below show the number of radial and angular shells which the DFT module will use for for a given atom depending on the row it is in (in the periodic table) and the desired accuracy. Note, differing atom types in a given molecular system will most likely have differing associated numerical grids. The intent is to generate the desired energy accuracy (at the expense of speed of the calculation).

Keyword	Radial	Angular
xcoarse	21	194
coarse	35	302
medium	49	434
fine	70	590
xfine	100	1202

Program default number of radial and angular shells empirically determined for Row 1 atoms (Li → F) to reach the desired accuracies.

Keyword	Radial	Angular
xcoarse	42	194
coarse	70	302
medium	88	434
fine	123	770
xfine	125	1454
huge	300	1454

Program default number of radial and angular shells empirically determined for Row 2 atoms (Na → Cl) to reach the desired accuracies.

Keyword	Radial	Angular
xcoarse	75	194
coarse	95	302
medium	112	590
fine	130	974
xfine	160	1454
huge	400	1454

Program default number of radial and angular shells empirically determined for Row 3 atoms (K → Br) to reach the desired accuracies.

Keyword	Radial	Angular
xcoarse	84	194
coarse	104	302
medium	123	590
fine	141	974
xfine	205	1454
huge	400	1454

Program default number of radial and angular shells empirically determined for Row 4 atoms (Rb → I) to reach the desired accuracies.

11.2.14.1 Angular grids

In addition to the simple keyword specifying the desired accuracy as described above, the user has the option of specifying

a custom quadrature of this type in which ALL atoms have the same grid specification. This is accomplished by using the `gausleg` keyword.

11.2.14.1.1 Gauss-Legendre angular grid

```
GRID gausleg <integer nradpts default 50> <integer nagrid default 10>
```

In this type of grid, the number of phi points is twice the number of theta points. So, for example, a specification of,

```
GRID gausleg 80 20
```

would be interpreted as 80 radial points, 20 theta points, and 40 phi points per center (or 64000 points per center before pruning).

11.2.14.1.2 Lebedev angular grid

A second quadrature is the Lebedev scheme for the angular components. Within this numerical integration procedure various levels of accuracy have also been defined and are available to the user. The input for this type of grid takes the form,

```
GRID lebedev <integer radpts > <integer iangquad >
```

In this context the variable `iangquad` specifies a certain number of angular points as indicated by the table below:

IANGQUAD	**$N_{angular}$**	**l**
1	38	9
2	50	11
3	74	13
4	86	15
5	110	17
6	146	19
7	170	21
8	194	23
9	230	25
10	266	27
11	302	29
12	350	31
13	434	35
14	590	41
15	770	47
16	974	53
17	1202	59

IANGQUAD	$N_{angular}$	P
18	1454	65
19	1730	71
20	2030	77
21	2354	83
22	2702	89
23	3074	95
24	3470	101
25	3890	107
26	4334	113
27	4802	119
28	5294	125
29	5810	131

Table 2: List of Lebedev quadratures

Therefore the user can specify any number of radial points along with the level of angular quadrature (1-29).

The user can also specify grid parameters specific for a given atom type: parameters that must be supplied are: atom tag and number of radial points. As an example, here is a grid input line for the water molecule

```
grid lebedev 80 11 H 70 8  O 90 11
```

11.2.14.2 Partitioning functions

```
GRID [(becke||erf1||erf2||ssf) default erf1]
```

- becke : see paper[100]
- ssf : see paper[101]
- erf1 : modified ssf
- erf2 : modified ssf

Erf*n* partitioning functions

$$\begin{array}{lcl} w_A(r) & = & \prod_{B\neq A}\frac{1}{2} \left[1 \ - \ erf(\mu^\prime_{AB})\right] \\ \mu^\prime_{AB} & = & \frac{1}{\alpha} \ \frac{\mu_{AB}}{(1-\mu_{AB}^2)^n} \\ \mu_{AB} & = & \frac{{\mathbf r}_A - {\mathbf r}_B} {\left|{\mathbf r}_A - {\mathbf r}_B \right|} \end{array}$$

11.2.14.3 Radial grids

```
GRID [[euler||mura||treutler]  default mura]
```

- euler : Euler-McLaurin quadrature with the transformation devised by Murray et al[102].
- mura : Modification of the Murray-Handy-Laming scheme (we are not using the same scaling factors proposed in the

paper by Mura and Knowles[103]).

- treutler : Gauss-Chebyshev using the transformation suggested by Treutler[104].

11.2.14.4 Disk usage for Grid

```
NODISK
```

This keyword turns off storage of grid points and weights on disk.

11.2.15 TOLERANCES: Screening tolerances

```
TOLERANCES [[tight] [tol_rho <real tol_rho default 1e-10>] \
        [accCoul <integer accCoul default 8>] \
        [radius <real radius default 25.0>]]
```

The user has the option of controlling screening for the tolerances in the integral evaluations for the DFT module. In most applications, the default values will be adequate for the calculation, but different values can be specified in the input for the DFT module using the keywords described below.

The input parameter accCoul is used to define the tolerance in Schwarz screening for the Coulomb integrals. Only integrals with estimated values greater than $10^{(-accCoul)}$ are evaluated.

```
TOLERANCES accCoul <integer accCoul default 8>
```

Screening away needless computation of the XC functional (on the grid) due to negligible density is also possible with the use of,

```
TOLERANCES tol_rho <real tol_rho default 1e-10>
```

XC functional computation is bypassed if the corresponding density elements are less than tol_rho .

A screening parameter, radius, used in the screening of the Becke or Delley spatial weights is also available as,

```
TOLERANCES radius <real radius default 25.0>
```

where radius is the cutoff value in bohr.

The tolerances as discussed previously are insured at convergence. More sleazy tolerances are invoked early in the iterative process which can speed things up a bit. This can also be problematic at times because it introduces a discontinuity in the convergence process. To avoid use of initial sleazy tolerances the user can invoke the tight option:

```
TOLERANCES tight
```

This option sets all tolerances to their default/user specified values at the very first iteration.

11.2.16 DIRECT, SEMIDIRECT and NOIO: Hardware Resource Control

```
DIRECT||INCORE
SEMIDIRECT [filesize <integer filesize default disksize>]
        [memsize  <integer memsize default available>]
        [filename <string filename default $file_prefix.aoints$]
NOIO
```

The inverted charge-density and exchange-correlation matrices for a DFT calculation are normally written to disk storage. The user can prevent this by specifying the keyword noio within the input for the DFT directive. The input to exercise this option is as follows,

```
noio
```

If this keyword is encountered, then the two matrices (inverted charge-density and exchange-correlation) are computed “on-the-fly” whenever needed.

The INCORE option is always assumed to be true but can be overridden with the option DIRECT in which case all integrals are computed “on-the-fly”.

The SEMIDIRECT option controls caching of integrals. A full description of this option is described in the Hartree-Fock section. Some functionality which is only compatible with the DIRECT option will not, at present, work when using SEMIDIRECT .

11.2.17 ODFT and MULT: Open shell systems

```
ODFT
MULT <integer mult default 1>
```

Both closed-shell and open-shell systems can be studied using the DFT module. Specifying the keyword MULT within the DFT directive allows the user to define the spin multiplicity of the system. The form of the input line is as follows;

```
MULT <integer mult default 1>
```

When the keyword MULT is specified, the user can define the integer variable mult , where mult is equal to the number of alpha electrons minus beta electrons, plus 1.

When MULT is set to a negative number. For example, if MULT = -3 , a triplet calculation will be performed with the beta electrons preferentially occupied. For MULT = 3 , the alpha electrons will be preferentially occupied.

The keyword ODFT is unnecessary except in the context of forcing a singlet system to be computed as an open shell system (i.e., using a spin-unrestricted wavefunction).

11.2.18 CGMIN: Quadratic convergence algorithm

The cgmin keyword will use the quadratic convergence algorithm. It is possible to turn the use of the quadratic convergence algorithm off with the nocgmin keyword.

11.2.19 RODFT: Restricted open-shell DFT

The rodft keyword will perform restricted open-shell calculations. This keyword can only be used with the CGMIN keyword.

11.2.20 SIC: Self-Interaction Correction

```
sic [perturbative || oep || oep-loc ]
<default perturbative>
```

The Perdew and Zunger[15] method to remove the self-interaction contained in many exchange-correlation functionals has been implemented with the Optimized Effective Potential method[105,106] within the Krieger-Li-Iafrate approximation[107,108,109]. Three variants of these methods are included in NWChem:

- sic perturbative This is the default option for the sic directive. After a self-consistent calculation, the Kohn-Sham orbitals are localized with the Foster-Boys algorithm (see section on orbital localization) and the self-interaction energy is added to the total energy. All exchange-correlation functionals implemented in the NWChem can be used with this option.
- sic oep With this option the optimized effective potential is built in each step of the self-consistent process. Because the electrostatic potential generated for each orbital involves a numerical integration, this method can be expensive.

- sic oep-loc This option is similar to the oep option with the addition of localization of the Kohn-Sham orbitals in each step of the self-consistent process.

With oep and oep-loc options a xfine grid (see section about numerical integration) must be used in order to avoid numerical noise, furthermore the hybrid functionals can not be used with these options. More details of the implementation of this method can be found in the paper by Garza[110]. The components of the sic energy can be printed out using:

```
print "SIC information"
```

11.2.21 MULLIKEN: Mulliken analysis

Mulliken analysis of the charge distribution is invoked by the keyword:

```
MULLIKEN
```

When this keyword is encountered, Mulliken analysis of both the input density as well as the output density will occur. For example, to perform a mulliken analysis and print the explicit population analysis of the basis functions, use the following

```
dft
 mulliken
 print "mulliken ao"
end
task dft
```

11.2.22 FUKUI: Fukui Indices

Fukui indices analysis is invoked by the keyword:

```
FUKUI
```

When this keyword is encounters, the condensed Fukui indices will be calculated and printed in the output. Detailed information about the analysis can be obtained using the following

```
dft
  fukui
  print "Fukui information"
end
task dft
```

The implementation of the Fukui analyis in NWChem was based on the papers by Galvan[111] and by Chamorro[112].
This implementation makes use of the generalized Fukui indices (\(f_{SN}, f_{NS}, f_{SS}\)).
The traditional, spin-restricted, Fukui indices are given by \(f_{NN}^+\), \(f_{NN}^-\) and their average:

\[\begin{array}{lcl} f_A^+ &=& f_{NN}^+ \\ f_A^- &=& f_{NN}^- \\ f_A^0 &=& \frac{1}{2}\left(f_{NN}^+ + f_{NN}^- \right) \end{array}\]

11.2.23 BSSE: Basis Set Superposition Error

Particular care is required to compute BSSE by the counter-poise method for the DFT module. In order to include terms deriving from the numerical grid used in the XC integration, the user must label the ghost atoms not just bq, but bq followed by the given atomic symbol. For example, the first component needed to compute the BSSE for the water dimer, should be written as follows

```
geometry h2o autosym units au
 O       0.00000000     0.00000000     0.22143139
 H       1.43042868     0.00000000    -0.88572555
 H      -1.43042868     0.00000000    -0.88572555
 bqH     0.71521434     0.00000000    -0.33214708
 bqH    -0.71521434     0.00000000    -0.33214708
 bqO     0.00000000     0.00000000    -0.88572555
end
basis
 H library aug-cc-pvdz
 O library aug-cc-pvdz
 bqH library H aug-cc-pvdz
 bqO library O aug-cc-pvdz
end
```

Please note that the *ghost* oxygen atom has been labeled `bqO`, and not just `bq`.

11.2.24 DISP: Empirical Long-range Contribution (vdW)

```
DISP \
    [ vdw <real vdw integer default 2]] \
    [[s6 <real s6 default depends on XC functional>] \
    [ alpha <real alpha default 20.0d0] \
    [ off ]
```

When systems with high dependence on van der Waals interactions are computed, the dispersion term may be added empirically through long-range contribution DFT-D, i.e. $E_{DFT-D}=E_{DFT-KS}+E_{disp}$, where:

$$E_{disp}=-s_6\sum_{i=1}^{N_{atom}-1}\sum_{j=i+1}^{N_{atom}} \frac{C_{6}^{ij}}{R_{ij}^{6}} \frac{1}{ 1+e^{-\alpha \frac{R_{ij}}{R_{vdw}}}}$$

In this equation, the s_6 term depends in the functional and basis set used, C_6^{ij} is the dispersion coefficient between pairs of atoms. R_{vdw} and R_{ij} are related with van der Waals atom radii and the nucleus distance respectively. The α value contributes to control the corrections at intermediate distances.

There are available three ways to compute C^6_{ij}:

1. `vdw 0` uses the formula $C_6^{ij}= \frac{2(C_6^{i}C_6^{j})^{2/3}(N_{eff\ i}N_{eff\ j})^{1/3}} {C_6^{i}(N_{eff\ i}^2)^{1/3}+(C_6^{i}N_{eff\ j}^2)^{1/3}}$
   where N_{eff} and C_6 are obtained from references [113] and [114]
2. `vdw 1` uses the formula $C_6^{ij}=2\ \frac{C_6^{i}C_6^{j}}{C_6^{i}+C_6^{j}}$.
   See details in reference[115].
3. `vdw 2` uses the formula $C_6^{ij}=\sqrt{C_6^{i}C_6^{j}}$.
   See details in reference[96].

Note that in each option there is a certain set ofC_6 and R_{vdw}. Also note that Grimme only defined parameters for elements up to Z=54 for the dispersion correction above. C_6 values for elements above Z=54 have been set to zero.

For options `vdw 1` and `vdw 2`, there are s_6 values by default for some functionals and triple-zeta plus double polarization basis set (TZV2P):

- `vdw 1` BLYP 1.40, PBE 0.70 and BP86 1.30.
- `vdw 2` BLYP 1.20, PBE 0.75, BP86 1.05, B3LYP 1.05, Becke97-D 1.25 and TPSS 1.00.

Grimme's DFT-D3 is also available. Here the dispersion term has the following form:

$$E_{disp} = \sum_{i,j} \sum_{n=6,8} s_n \ \frac{C^{ij}_n}{r_{ij}} \biggl\lbrace 1 + 6 \bigl[\frac{r_{ij}}{s_{r,n} \ R_0^{ij}} \bigr]^{-\alpha n} \biggr\rbrace^{-1}$$

This new dispersion correction covers elements through Z=94. C^{ij}_n *(n=6,8)* are coordination and geometry dependent. Details about the functional form can be found in reference [116].

To use the Grimme DFT-D3 dispersion correction, use the option

- `vdw 3` (`s6` and `alpha` cannot be set manually). Functionals for which DFT-D3 is available in NWChem are BLYP, B3LYP, BP86, Becke97-D, PBE96, TPSS, PBE0, B2PLYP, BHLYP, TPSSH, PWB6K, B1B95, SSB-D, MPW1B95, MPWB1K, M05, M05-2X, M06L, M06, M06-2X, and M06HF
- `vdw 4` triggers the DFT-D3BJ dispersion model. Currently only BLYP, B3LYP, BHLYP, TPSS, TPSSh, B2-PLYP, B97-D, BP86, PBE96, PW6B95, revPBE, B3PW91, pwb6k, b1b95, CAM-B3LYP, LC-wPBE, HCTH120, MPW1B95, BOP, OLYP, BPBE, OPBE and SSB are supported.

This capability is also supported for energy gradients and Hessian. Is possible to be deactivated with OFF.

11.2.25 NOSCF: Non Self-Consistent Calculations

The `noscf` keyword can be used to to calculate the non self-consistent energy for a set of input vectors. For example, the following input shows how a non self-consistent B3LYP energy can be calculated using a self-consistent set of vectors calculated at the Hartree-Fock level.

```
start h2o-noscf

geometry units angstrom
  O      0.00000000     0.00000000     0.11726921
  H      0.75698224     0.00000000    -0.46907685
  H     -0.75698224     0.00000000    -0.46907685
end

basis spherical
  * library aug-cc-pvdz
end
dft
  xc hfexch
  vectors output hf.movecs
end
task dft energy
dft
  xc b3lyp
  vectors input hf.movecs
  noscf
end
task dft energy
```

11.2.26 XDM: Exchange-hole dipole moment dispersion model

```
XDM  [ a1 <real a1> ]   [ a2 <real a2> ]
```

See details (including list of a1 and a2 parameters) in paper [117] and the website https://erin-r-johnson.github.io/software/

```
geometry
  O       -0.190010095135    -1.168397415155     0.925531922479
  H       -0.124425719598    -0.832776238160     1.818190662986
  H       -0.063897685990    -0.392575837594     0.364048725248
  O        0.174717244879     1.084630474836    -0.860510672419
  H       -0.566281023931     1.301941006866    -1.427261487135
  H        0.935093179777     1.047335209207    -1.441842151158
end

basis spherical
  * library aug-cc-pvdz
end

dft
   direct
   xc b3lyp
   xdm a1 0.6224 a2 1.7068
end

task dft optimize
```

11.2.27 Print Control

PRINT||NOPRINT

The PRINT||NOPRINT options control the level of output in the DFT. Please see some examples using this directive in Sample input file. Known controllable print options are:

Name	Print Level	Description
“all vector symmetries”	high	symmetries of all molecular orbitals
“alpha partner info”	high	unpaired alpha orbital analysis
“common”	debug	dump of common blocks
“convergence”	default	convergence of SCF procedure
“coulomb fit”	high	fitting electronic charge density
“dft timings”	high	
“final vectors”	high	
“final vectors analysis”	high	print all orbital energies and orbitals
“final vector symmetries”	default	symmetries of final molecular orbitals
“information”	low	general information
“initial vectors”	high	
“intermediate energy info”	high	
“intermediate evals”	high	intermediate orbital energies
“intermediate fock matrix”	high	
“intermediate overlap”	high	overlaps between the alpha and beta sets
“intermediate S2”	high	values of S2
“intermediate vectors”	high	intermediate molecular orbitals
“interm vector symm”	high	symmetries of intermediate orbitals
“io info”	debug	reading from and writing to disk
“kinetic_energy”	high	kinetic energy
“mulliken ao”	high	mulliken atomic orbital population
“multipole”	default	moments of alpha, beta, and nuclear charge densities
“parameters”	default	input parameters
“quadrature”	high	numerical quadrature
“schwarz”	high	integral screening info & stats at completion
“screening parameters”	high	integral accuracies
“semi-direct info”	default	semi direct algorithm

11.2.28 Spin-Orbit Density Functional Theory (SODFT)

The spin-orbit DFT module (SODFT) in the NWChem code allows for the variational treatment of the one-electron spin-orbit operator within the DFT framework. Calculations can be performed either with an electron relativistic approach (ZORA) or with an effective core potential (ECP) and a matching spin-orbit potential (SO). The current implementation does NOT use symmetry.

The actual SODFT calculation will be performed when the input module encounters the TASK directive (TASK).

```
TASK SODFT
```

Input parameters are the same as for the DFT. Some of the DFT options are not available in the SODFT. These are max_ovl and sic .

Besides using the standard ECP and basis sets, see Effective Core Potentials for details, one also has to specify a spin-orbit (SO) potential. The input specification for the SO potential can be found in Effective Core Potentials. At this time we have not included any spin-orbit potentials in the basis set library. However, one can get these from the Stuttgart/Köln web pages http://www.tc.uni-koeln.de/PP/clickpse.en.html.

Note: One should use a combination of ECP and SO potentials that were designed for the same size core, i.e., don't use a small core ECP potential with a large core SO potential (it will produce erroneous results).

The following is an example of a calculation of UO_2:

```
start uo2_sodft
echo

charge 2
geometry
 U     0.00000     0.00000     0.00000
 O     0.00000     0.00000     1.68000
 O     0.00000     0.00000    -1.68000
end
basis "ao basis"
 * library "stuttgart rlc ecp"
END
ECP
 * library "stuttgart rlc ecp"
END
SO
 U p
 2    3.986181     1.816350
 2    2.000160    11.543940
 2    0.960841     0.794644
 U d
 2    4.147972     0.353683
 2    2.234563     3.499282
 2    0.913695     0.514635
 U f
 2    3.998938     4.744214
 2    1.998840    -5.211731
 2    0.995641     1.867860
END
dft
 mult 1
 xc hfexch
end
task sodft
```

11.2.29 SYM and ADAPT

The options SYM and ADAPT works the same way as the analogous options for the SCF code. Therefore please use the following links for SYM and ADAPT, respectively.

11.2.30 References

1. Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. On First-Row Diatomic Molecules and Local Density Models. *The Journal of Chemical Physics* **1979**, *71* (12), 4993. https://doi.org/10.1063/1.438313
2. Eichkorn, K.; Treutler, O.; Öhm, H.; Häser, M.; Ahlrichs, R. Auxiliary Basis Sets to Approximate Coulomb Potentials. *Chemical Physics Letters* **1995**, *240* (4), 283–290. https://doi.org/10.1016/0009-2614(95)00621-a
3. Köster, A. M.; Reveles, J. U.; Campo, J. M. del. Calculation of Exchange-Correlation Potentials with Auxiliary Function Densities. *The Journal of Chemical Physics* **2004**, *121* (8), 3417–3424. https://doi.org/10.1063/1.1771638
4. Calaminici, P.; Alvarez-Ibarra, A.; Cruz-Olvera, D.; Domìnguez-Soria, V.-D.; Flores-Moreno, R.; Gamboa, G. U.; Geudtner, G.; Goursot, A.; Mejía-Rodríguez, D.; Salahub, D. R.; Zuniga-Gutierrez, B.; Köster, A. M. Auxiliary Density Functional Theory: From Molecules to Nanostructures. In *Handbook of computational chemistry*; Leszczynski, J., Kaczmarek-Kedziera, A., Puzyn, T., G. Papadopoulos, M., Reis, H., K. Shukla, M., Eds.; Springer International Publishing, 2017; pp 795–860. https://doi.org/10.1007/978-3-319-27282-5\16
5. Barca, G. M. J.; Gilbert, A. T. B.; Gill, P. M. W. Simple Models for Difficult Electronic Excitations. *Journal of Chemical Theory and Computation* **2018**, *14* (3), 1501–1509. https://doi.org/10.1021/acs.jctc.7b00994
6. Lykos, P. G.; Schmeising, H. N. Maximum Overlap Atomic and Molecular Orbitals. *The Journal of Chemical Physics* **1961**, *35* (1), 288–293. https://doi.org/10.1063/1.1731901
7. King, H. F.; Stanton, R. E.; Kim, H.; Wyatt, R. E.; Parr, R. G. Corresponding Orbitals and the Nonorthogonality Problem in Molecular Quantum Mechanics. *The Journal of Chemical Physics* **1967**, *47* (6), 1936–1941. https://doi.org/10.1063/1.1712221
8. Gilbert, A. T. B.; Besley, N. A.; Gill, P. M. W. Self-Consistent Field Calculations of Excited States Using the Maximum Overlap Method (MOM). *The Journal of Physical Chemistry A* **2008**, *112* (50), 13164–13171. https://doi.org/10.1021/jp801738f
9. Slater, J. C.; Johnson, K. H. Self-Consistent-Field *X**α* Cluster Method for Polyatomic Molecules and Solids. *Phys. Rev. B* **1972**, *5*, 844–853. https://doi.org/10.1103/PhysRevB.5.844
10. Slater, J. C. *Quantum Theory of Molecules and Solids: The Self-Consistent Field for Molecules and Solids* ; McGraw-Hill Education, 1974; p 640.
11. Vosko, S. H.; Wilk, L.; Nusair, M. Accurate Spin-Dependent Electron Liquid Correlation Energies for Local Spin Density Calculations: A Critical Analysis. *Canadian Journal of physics* **1980**, *58* (8), 1200–1211. https://doi.org/10.1139/p80-159
12. Becke, A. D. A New Mixing of Hartree-Fock and Local Density-Functional Theories. *The Journal of Chemical Physics* **1993**, *98* (2), 1372–1377. https://doi.org/https://aip.scitation.org/doi/abs/10.1063/1.464304
13. Becke, A. D. Density-Functional Thermochemistry. III. The Role of Exact Exchange. *The Journal of Chemical Physics* **1993**, *98* (7), 5648–5652. https://doi.org/10.1063/1.464913
14. Becke, A. D. Density-Functional Thermochemistry. V. Systematic Optimization of Exchange-Correlation Functionals. *The Journal of Chemical Physics* **1997**, *107* (20), 8554–8560. https://doi.org/10.1063/1.475007
15. Perdew, J. P.; Zunger, A. Self-Interaction Correction to Density-Functional Approximations for Many-Electron Systems. *Phys. Rev. B* **1981**, *23*, 5048–5079. https://doi.org/10.1103/PhysRevB.23.5048
16. Perdew, J. P.; Wang, Y. Accurate and Simple Analytic Representation of the Electron-Gas Correlation Energy. *Phys. Rev. B* **1992**, *45*, 13244–13249. https://doi.org/10.1103/PhysRevB.45.13244
17. Becke, A. D. On the Large-Gradient Behavior of the Density Functional Exchange Energy. *The Journal of Chemical Physics* **1986**, *85* (12), 7184–7187. https://doi.org/10.1063/1.451353
18. Becke, A. D. Density-Functional Exchange-Energy Approximation with Correct Asymptotic Behavior. *Physical Review A* **1988**, *38* (6), 3098–3100. https://doi.org/10.1103/PhysRevA.38.3098
19. Perdew, J. P. Density-Functional Approximation for the Correlation Energy of the Inhomogeneous Electron Gas. *Physical Review B* **1986**, *33* (12), 8822–8824. https://doi.org/10.1103/PhysRevB.33.8822
20. Perdew, J. P.; Burke, K.; Ernzerhof, M. Errata: Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.* **1996**, *77*, 3865–3868. https://doi.org/10.1103/PhysRevLett.77.3865
21. Perdew, J. P.; Burke, K.; Ernzerhof, M. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.* **1997**, *78*, 1396–1396. https://doi.org/10.1103/PhysRevLett.78.1396
22. Gill, P. M. W. A New Gradient-Corrected Exchange Functional. *Molecular Physics* **1996**, *89* (2), 433–445. https://doi.org/10.1080/002689796173813
23. Handy, N. C.; Cohen, A. J. Left-Right Correlation Energy. *Molecular Physics* **2001**, *99* (5), 403–412. https://doi.org/10.1080/00268970010018431
24. Adamo, C.; Barone, V. Exchange Functionals with Improved Long-Range Behavior and Adiabatic Connection Methods Without Adjustable Parameters: The mPW and mPW1PW Models. *The Journal of Chemical Physics* **1998**, *108* (2), 664–675. https://doi.org/10.1063/1.475428
25. Zhao, Y.; Truhlar, D. G. Design of Density Functionals That Are Broadly Accurate for Thermochemistry, Thermochemical Kinetics, and Nonbonded Interactions. *The Journal of Physical Chemistry A* **2005**, *109* (25), 5656–5667. https://doi.org/10.1021/jp050536c
26. Filatov, M.; Thiel, W. A New Gradient-Corrected Exchange-Correlation Density Functional. *Molecular Physics* **1997**, *91* (5), 847–860. https://doi.org/10.1080/002689797170950
27. Filatov, M.; Thiel, W. A Nonlocal Correlation Energy Density Functional from a Coulomb Hole Model. *International Journal of Quantum Chemistry* **1997**, *62* (6), 603–616. https://doi.org/10.1002/(SICI)1097-461X(1997)62:6<603::AID-QUA4>3.0.CO;2-\23
28. Hammer, B.; Hansen, L. B.; Nørskov, J. K. Improved Adsorption Energetics Within Density-Functional Theory Using Revised Perdew-Burke-Ernzerhof Functionals. *Physical Review B* **1999**, *59* (11), 7413–7421. https://doi.org/10.1103/physrevb.59.7413

29. Zhang, Y.; Yang, W. Comment on Generalized Gradient Approximation Made Simple. *Physical Review Letters* **1998**, *80* (4), 890–890. https://doi.org/10.1103/physrevlett.80.890
30. Zhao, Y.; Lynch, B. J.; Truhlar, D. G. Development and Assessment of a New Hybrid Density Functional Model for Thermochemical Kinetics. *The Journal of Physical Chemistry A* **2004**, *108* (14), 2715–2719. https://doi.org/10.1021/jp049908s
31. Lee, C.; Yang, W.; Parr, R. G. Development of the Colle-Salvetti Correlation-Energy Formula into a Functional of the Electron Density. *Physical Review B* **1988**, *37* (2), 785–789. https://doi.org/10.1103/PhysRevB.37.785
32. Perdew, J. P.; Chevary, J. A.; Vosko, S. H.; Jackson, K. A.; Pederson, M. R.; Singh, D. J.; Fiolhais, C. Atoms, Molecules, Solids, and Surfaces: Applications of the Generalized Gradient Approximation for Exchange and Correlation. *Physical Review B* **1992**, *46* (11), 6671–6687. https://doi.org/10.1103/PhysRevB.46.6671
33. Perdew, J. P.; Chevary, J. A.; Vosko, S. H.; Jackson, K. A.; Pederson, M. R.; Singh, D. J.; Fiolhais, C. Erratum: Atoms, Molecules, Solids, and Surfaces: Applications of the Generalized Gradient Approximation for Exchange and Correlation. *Physical Review B* **1993**, *48* (7), 4978–4978. https://doi.org/10.1103/PhysRevB.48.4978.2
34. Tsuneda, T.; Suzumura, T.; Hirao, K. A New One-Parameter Progressive Collesalvetti-Type Correlation Functional. *The Journal of Chemical Physics* **1999**, *110* (22), 10664–10678. https://doi.org/10.1063/1.479012
35. Hamprecht, F. A.; Cohen, A. J.; Tozer, D. J.; Handy, N. C. Development and Assessment of New Exchange-Correlation Functionals. *The Journal of Chemical Physics* **1998**, *109* (15), 6264–6271. https://doi.org/10.1063/1.477267
36. Boese, A. D.; Doltsinis, N. L.; Handy, N. C.; Sprik, M. New Generalized Gradient Approximation Functionals. *The Journal of Chemical Physics* **2000**, *112* (4), 1670–1678. https://doi.org/10.1063/1.480732
37. Boese, A. D.; Martin, J. M. L.; Handy, N. C. The Role of the Basis Set: Assessing Density Functional Theory. *The Journal of Chemical Physics* **2003**, *119* (6), 3005–3014. https://doi.org/10.1063/1.1589004
38. Boese, A. D.; Handy, N. C. A New Parametrization of Exchangecorrelation Generalized Gradient Approximation Functionals. *The Journal of Chemical Physics* **2001**, *114* (13), 5497–5503. https://doi.org/10.1063/1.1347371
39. Cohen, A. J.; Handy, N. C. Assessment of Exchange Correlation Functionals. *Chemical Physics Letters* **2000**, *316* (1-2), 160–166. https://doi.org/10.1016/s0009-2614(99)01273-7
40. Menconi, G.; Wilson, P. J.; Tozer, D. J. Emphasizing the Exchange-Correlation Potential in Functional Development. *The Journal of Chemical Physics* **2001**, *114* (9), 3958–3967. https://doi.org/10.1063/1.1342776
41. Boese, A. D.; Chandra, A.; Martin, J. M. L.; Marx, D. From Ab Initio Quantum Chemistry to Molecular Dynamics: The Delicate Case of Hydrogen Bonding in Ammonia. *The Journal of Chemical Physics* **2003**, *119* (12), 5965–5980. https://doi.org/10.1063/1.1599338
42. Tsuneda, T.; Suzumura, T.; Hirao, K. A Reexamination of Exchange Energy Functionals. *The Journal of Chemical Physics* **1999**, *111* (13), 5656–5667. https://doi.org/10.1063/1.479954
43. Perdew, J. P.; Kurth, S.; Zupan, A.; Blaha, P. Accurate Density Functional with Correct Formal Properties: A Step Beyond the Generalized Gradient Approximation. *Physical Review Letters* **1999**, *82* (12), 2544–2547. https://doi.org/10.1103/physrevlett.82.2544
44. Tao, J.; Perdew, J. P.; Staroverov, V. N.; Scuseria, G. E. Climbing the Density Functional Ladder: Nonempirical Meta-Generalized Gradient Approximation Designed for Molecules and Solids. *Physical Review Letters* **2003**, *91* (14), 146401. https://doi.org/10.1103/physrevlett.91.146401
45. Zhao, Y.; Truhlar, D. G. Hybrid Meta Density Functional Theory Methods for Thermochemistry, Thermochemical Kinetics, and Noncovalent Interactions: The MPW1B95 and MPWB1K Models and Comparative Assessments for Hydrogen Bonding and van Der Waals Interactions. *The Journal of Physical Chemistry A* **2004**, *108* (33), 6908–6918. https://doi.org/10.1021/jp048147q
46. Zhao, Y.; Schultz, N. E.; Truhlar, D. G. Exchange-Correlation Functional with Broad Accuracy for Metallic and Nonmetallic Compounds, Kinetics, and Noncovalent Interactions. *The Journal of Chemical Physics* **2005**, *123* (16), 161103. https://doi.org/10.1063/1.2126975
47. Zhao, Y.; Schultz, N. E.; Truhlar, D. G. Design of Density Functionals by Combining the Method of Constraint Satisfaction with Parametrization for Thermochemistry, Thermochemical Kinetics, and Noncovalent Interactions. *Journal of Chemical Theory and Computation* **2006**, *2* (2), 364–382. https://doi.org/10.1021/ct0502763
48. Staroverov, V. N.; Scuseria, G. E.; Tao, J.; Perdew, J. P. Comparative Assessment of a New Nonempirical Density Functional: Molecules and Hydrogen-Bonded Complexes. *The Journal of Chemical Physics* **2003**, *119* (23), 12129–12137. https://doi.org/10.1063/1.1626543
49. Becke, A. D. Density-Functional Thermochemistry. IV. A New Dynamical Correlation Functional and Implications for Exact-Exchange Mixing. *The Journal of Chemical Physics* **1996**, *104* (3), 1040–1046. https://doi.org/10.1063/1.470829
50. Voorhis, T. V.; Scuseria, G. E. A Novel Form for the Exchange-Correlation Energy Functional. *The Journal of Chemical Physics* **1998**, *109* (2), 400–410. https://doi.org/10.1063/1.476577
51. Zhao, Y.; Truhlar, D. G. A New Local Density Functional for Main-Group Thermochemistry, Transition Metal Bonding, Thermochemical Kinetics, and Noncovalent Interactions. *The Journal of Chemical Physics* **2006**, *125* (19), 194101. https://doi.org/10.1063/1.2370993
52. Zhao, Y.; Truhlar, D. G. Density Functional for Spectroscopy: No Long-Range Self-Interaction Error, Good Performance for Rydberg and Charge-Transfer States, and Better Performance on Average Than B3LYP for Ground States. *The Journal of Physical Chemistry A* **2006**, *110* (49), 13126–13130. https://doi.org/10.1021/jp066479k
53. Zhao, Y.; Truhlar, D. G. The M06 Suite of Density Functionals for Main Group Thermochemistry, Thermochemical Kinetics, Noncovalent Interactions, Excited States, and Transition Elements: Two New Functionals and Systematic Testing of Four M06-Class Functionals and 12 Other Functionals. *Theoretical Chemistry Accounts* **2008**, *120* (1-3), 215–241. https://doi.org/10.1007/s00214-007-0310-x
54. Zhao, Y.; Truhlar, D. G. Construction of a Generalized Gradient Approximation by Restoring the Density-Gradient Expansion and Enforcing a Tight

Lieboxford Bound. *The Journal of Chemical Physics* **2008**, *128* (18), 184109. https://doi.org/10.1063/1.2912068

55. Peverati, R.; Truhlar, D. G. Improving the Accuracy of Hybrid Meta-GGA Density Functionals by Range Separation. *The Journal of Physical Chemistry Letters* **2011**, *2* (21), 2810–2817. https://doi.org/10.1021/jz201170d
56. Peverati, R.; Truhlar, D. G. M11-l: A Local Density Functional That Provides Improved Accuracy for Electronic Structure Calculations in Chemistry and Physics. *The Journal of Physical Chemistry Letters* **2012**, *3* (1), 117–124. https://doi.org/10.1021/jz201525m
57. Peverati, R.; Zhao, Y.; Truhlar, D. G. Generalized Gradient Approximation That Recovers the Second-Order Density-Gradient Expansion with Optimized Across-the-Board Performance. *The Journal of Physical Chemistry Letters* **2011**, *2* (16), 1991–1997. https://doi.org/10.1021/jz200616w
58. Peverati, R.; Truhlar, D. G. Communication: A Global Hybrid Generalized Gradient Approximation to the Exchange-Correlation Functional That Satisfies the Second-Order Density-Gradient Constraint and Has Broad Applicability in Chemistry. *The Journal of Chemical Physics* **2011**, *135* (19), 191102. https://doi.org/10.1063/1.3663871
59. Pernal, K.; Podeszwa, R.; Patkowski, K.; Szalewicz, K. Dispersionless Density Functional Theory. *Physical Review Letters* **2009**, *103* (26), 263201. https://doi.org/10.1103/physrevlett.103.263201
60. Wilson, P. J.; Bradley, T. J.; Tozer, D. J. Hybrid Exchange-Correlation Functional Determined from Thermochemical Data and Ab Initio Potentials. *The Journal of Chemical Physics* **2001**, *115* (20), 9233–9242. https://doi.org/10.1063/1.1412605
61. Keal, T. W.; Tozer, D. J. Semiempirical Hybrid Functional with Improved Performance in an Extensive Chemical Assessment. *The Journal of Chemical Physics* **2005**, *123* (12), 121103. https://doi.org/10.1063/1.2061227
62. Grimme, S. Semiempirical GGA-Type Density Functional Constructed with a Long-Range Dispersion Correction. *Journal of Computational Chemistry* **2006**, *27* (15), 1787–1799. https://doi.org/10.1002/jcc.20495
63. Schmider, H. L.; Becke, A. D. Optimized Density Functionals from the Extended G2 Test Set. *The Journal of Chemical Physics* **1998**, *108* (23), 9624–9631. https://doi.org/10.1063/1.476438
64. Adamo, C.; Barone, V. Toward Reliable Density Functional Methods Without Adjustable Parameters: The PBE0 Model. *The Journal of Chemical Physics* **1999**, *110* (13), 6158–6170. https://doi.org/10.1063/1.478522
65. Lynch, B. J.; Fast, P. L.; Harris, M.; Truhlar, D. G. Adiabatic Connection for Kinetics. *The Journal of Physical Chemistry A* **2000**, *104* (21), 4811–4815. https://doi.org/10.1021/jp000497z
66. Sun, J.; Perdew, J. P.; Ruzsinszky, A. Semilocal Density Functional Obeying a Strongly Tightened Bound for Exchange. *Proceedings of the National Academy of Sciences* **2015**, *112* (3), 685–689. https://doi.org/10.1073/pnas.1423145112
67. Verma, P.; Truhlar, D. G. HLE16: A Local Kohn-Sham Gradient Approximation with Good Performance for Semiconductor Band Gaps and Molecular Excitation Energies. *The Journal of Physical Chemistry Letters* **2017**, *8* (2), 380–387. https://doi.org/10.1021/acs.jpclett.6b02757
68. Yang, Z.; Peng, H.; Sun, J.; Perdew, J. P. More Realistic Band Gaps from Meta-Generalized Gradient Approximations: Only in a Generalized Kohn-Sham Scheme. *Physical Review B* **2016**, *93* (20), 205205. https://doi.org/10.1103/physrevb.93.205205
69. Mejia-Rodriguez, D.; Trickey, S. B. Deorbitalization Strategies for Meta-Generalized-Gradient-Approximation Exchange-Correlation Functionals. *Physical Review A* **2017**, *96* (5), 052512. https://doi.org/10.1103/physreva.96.052512
70. Wang, Y.; Jin, X.; Yu, H. S.; Truhlar, D. G.; He, X. Revised M06-l Functional for Improved Accuracy on Chemical Reaction Barrier Heights, Noncovalent Interactions, and Solid-State Physics. *Proceedings of the National Academy of Sciences* **2017**, *114* (32), 8487–8492. https://doi.org/10.1073/pnas.1705670114
71. Wang, Y.; Verma, P.; Jin, X.; Truhlar, D. G.; He, X. Revised M06 Density Functional for Main-Group and Transition-Metal Chemistry. *Proceedings of the National Academy of Sciences* **2018**, *115* (41), 10257–10262. https://doi.org/10.1073/pnas.1810421115
72. Chai, J.-D.; Head-Gordon, M. Systematic Optimization of Long-Range Corrected Hybrid Density Functionals. *The Journal of Chemical Physics* **2008**, *128* (8), 084106. https://doi.org/10.1063/1.2834918
73. Lin, Y.-S.; Li, G.-D.; Mao, S.-P.; Chai, J.-D. Long-Range Corrected Hybrid Density Functionals with Improved Dispersion Corrections. *Journal of Chemical Theory and Computation* **2012**, *9* (1), 263–272. https://doi.org/10.1021/ct300715s
74. Bartók, A. P.; Yates, J. R. Regularized SCAN Functional. *The Journal of Chemical Physics* **2019**, *150* (16), 161101. https://doi.org/10.1063/1.5094646
75. Furness, J. W.; Kaplan, A. D.; Ning, J.; Perdew, J. P.; Sun, J. Accurate and Numerically Efficient r2SCAN Meta-Generalized Gradient Approximation. *The Journal of Physical Chemistry Letters* **2020**, *11* (19), 8208–8215. https://doi.org/10.1021/acs.jpclett.0c02405
76. Bursch, M.; Neugebauer, H.; Ehlert, S.; Grimme, S. Dispersion Corrected r2SCAN Based Global Hybrid Functionals: r2SCANh, r2SCAN0, and r2SCAN50. *The Journal of Chemical Physics* **2022**, *156* (13), 134105. https://doi.org/10.1063/5.0086040
77. Mejía-Rodríguez, D.; Trickey, S. B. Meta-GGA Performance in Solids at Almost GGA Cost. *Physical Review B* **2020**, *102* (12), 121109. https://doi.org/10.1103/physrevb.102.121109
78. Mejía-Rodríguez, D.; Trickey, S. B. Spin-Crossover from a Well-Behaved, Low-Cost Meta-GGA Density Functional. *The Journal of Physical Chemistry A* **2020**, *124* (47), 9889–9894. https://doi.org/10.1021/acs.jpca.0c08883
79. Carmona-Espíndola, J.; Gázquez, J. L.; Vela, A.; Trickey, S. B. Generalized Gradient Approximation Exchange Energy Functional with Near-Best Semilocal Performance. *Journal of Chemical Theory and Computation* **2018**, *15* (1), 303–310. https://doi.org/10.1021/acs.jctc.8b00998
80. Kurth, S.; Perdew, J. P.; Blaha, P. Molecular and Solid-State Tests of Density Functional Approximations: LSD, GGAs, and Meta-GGAs. *International Journal of Quantum Chemistry* **1999**, *75* (4-5), 889–909. https://doi.org/10.1002/(sici)1097-461x(1999)75:4/5<889::aid-qua54>3.0.co;2-8
81. Savin, A. Beyond the Kohn-Sham Determinant. In *Recent advances in density functional methods* ; WORLD SCIENTIFIC, 1995; pp 129–153. https://doi.org/10.1142/9789812830586\0004

82. Iikura, H.; Tsuneda, T.; Yanai, T.; Hirao, K. A Long-Range Correction Scheme for Generalized-Gradient-Approximation Exchange Functionals. *The Journal of Chemical Physics* **2001**, *115* (8), 3540–3544. https://doi.org/10.1063/1.1383587
83. Tawada, Y.; Tsuneda, T.; Yanagisawa, S.; Yanai, T.; Hirao, K. A Long-Range-Corrected Time-Dependent Density Functional Theory. *The Journal of Chemical Physics* **2004**, *120* (18), 8425–8433. https://doi.org/10.1063/1.1688752
84. Yanai, T.; Tew, D. P.; Handy, N. C. A New Hybrid Exchange-Correlation Functional Using the Coulomb-Attenuating Method (CAM-B3LYP). *Chemical Physics Letters* **2004**, *393* (1-3), 51–57. https://doi.org/10.1016/j.cplett.2004.06.011
85. Peach, M. J. G.; Cohen, A. J.; Tozer, D. J. Influence of Coulomb-Attenuation on Exchange-Correlation Functional Quality. *Phys. Chem. Chem. Phys.* **2006**, *8* (39), 4543–4549. https://doi.org/10.1039/b608553a
86. Song, J.-W.; Hirosawa, T.; Tsuneda, T.; Hirao, K. Long-Range Corrected Density Functional Calculations of Chemical Reactions: Redetermination of Parameter. *The Journal of Chemical Physics* **2007**, *126* (15), 154105. https://doi.org/10.1063/1.2721532
87. Livshits, E.; Baer, R. A Well-Tempered Density Functional Theory of Electrons in Molecules. *Physical Chemistry Chemical Physics* **2007**, *9* (23), 2932. https://doi.org/10.1039/b617919c
88. Cohen, A. J.; Mori-Sánchez, P.; Yang, W. Development of Exchange-Correlation Functionals with Minimal Many-Electron Self-Interaction Error. *The Journal of Chemical Physics* **2007**, *126* (19), 191109. https://doi.org/10.1063/1.2741248
89. Rohrdanz, M. A.; Herbert, J. M. Simultaneous Benchmarking of Ground- and Excited-State Properties with Long-Range-Corrected Density Functional Theory. *The Journal of Chemical Physics* **2008**, *129* (3), 034107. https://doi.org/10.1063/1.2954017
90. Govind, N.; Valiev, M.; Jensen, L.; Kowalski, K. Excitation Energies of Zinc Porphyrin in Aqueous Solution Using Long-Range Corrected Time-Dependent Density Functional Theory. *The Journal of Physical Chemistry A* **2009**, *113* (21), 6041–6043. https://doi.org/10.1021/jp902118k
91. Baer, R.; Livshits, E.; Salzner, U. Tuned Range-Separated Hybrids in Density Functional Theory. *Annual Review of Physical Chemistry* **2010**, *61* (1), 85–109. https://doi.org/10.1146/annurev.physchem.012809.103321
92. Autschbach, J.; Srebro, M. Delocalization Error and Functional Tuning in Kohn-Sham Calculations of Molecular Properties. *Accounts of Chemical Research* **2014**, *47* (8), 2592–2602. https://doi.org/10.1021/ar500171t
93. Verma, P.; Bartlett, R. J. Increasing the Applicability of Density Functional Theory. IV. Consequences of Ionization-Potential Improved Exchange-Correlation Potentials. *The Journal of Chemical Physics* **2014**, *140* (18), 18A534. https://doi.org/10.1063/1.4871409
94. Swart, M.; Solà, M.; Bickelhaupt, F. M. A New All-Round Density Functional Based on Spin States and SN2 Barriers. *The Journal of Chemical Physics* **2009**, *131* (9), 094103. https://doi.org/10.1063/1.3213193
95. Swart, M.; Solà, M.; Bickelhaupt, F. M. Switching Between OPTX and PBE Exchange Functionals. *Journal of Computational Methods in Sciences and Engineering* **2009**, *9* (1-2), 69–77. https://doi.org/10.3233/jcm-2009-0230
96. Grimme, S. Semiempirical Hybrid Density Functional with Perturbative Second-Order Correlation. *The Journal of Chemical Physics* **2006**, *124* (3), 034108. https://doi.org/10.1063/1.2148954
97. Rabuck, A. D.; Scuseria, G. E. Improving Self-Consistent Field Convergence by Varying Occupation Numbers. *The Journal of Chemical Physics* **1999**, *110* (2), 695–700. https://doi.org/10.1063/1.478177
98. Wu, Q.; Voorhis, T. V. Direct Optimization Method to Study Constrained Systems Within Density-Functional Theory. *Physical Review A* **2005**, *72* (2), 024502. https://doi.org/10.1103/physreva.72.024502
99. Warren, R. W.; Dunlap, B. I. Fractional Occupation Numbers and Density Functional Energy Gradients Within the Linear Combination of Gaussian-Type Orbitals Approach. *Chemical Physics Letters* **1996**, *262* (3-4), 384–392. https://doi.org/10.1016/0009-2614(96)01107-4
100. Becke, A. D. A Multicenter Numerical Integration Scheme for Polyatomic Molecules. *The Journal of Chemical Physics* **1988**, *88* (4), 2547–2553. https://doi.org/10.1063/1.454033
101. Stratmann, R. E.; Scuseria, G. E.; Frisch, M. J. Achieving Linear Scaling in Exchange-Correlation Density Functional Quadratures. *Chemical Physics Letters* **1996**, *257* (3-4), 213–223. https://doi.org/10.1016/0009-2614(96)00600-8
102. Murray, C. W.; Handy, N. C.; Laming, G. J. Quadrature Schemes for Integrals of Density Functional Theory. *Molecular Physics* **1993**, *78* (4), 997–1014. https://doi.org/10.1080/00268979300100651
103. Mura, M. E.; Knowles, P. J. Improved Radial Grids for Quadrature in Molecular Density-Functional Calculations. *The Journal of Chemical Physics* **1996**, *104* (24), 9848–9858. https://doi.org/10.1063/1.471749
104. Treutler, O.; Ahlrichs, R. Efficient Molecular Numerical Integration Schemes. *The Journal of Chemical Physics* **1995**, *102* (1), 346–354. https://doi.org/10.1063/1.469408
105. Sharp, R. T.; Horton, G. K. A Variational Approach to the Unipotential Many-Electron Problem. *Physical Review* **1953**, *90* (2), 317–317. https://doi.org/10.1103/physrev.90.317
106. Talman, J. D.; Shadwick, W. F. Optimized Effective Atomic Central Potential. *Physical Review A* **1976**, *14* (1), 36–40. https://doi.org/10.1103/physreva.14.36
107. Krieger, J. B.; Li, Y.; Iafrate, G. J. Construction and Application of an Accurate Local Spin-Polarized Kohn-Sham Potential with Integer Discontinuity: Exchange-Only Theory. *Physical Review A* **1992**, *45* (1), 101–126. https://doi.org/10.1103/physreva.45.101
108. Krieger, J. B.; Li, Y.; Iafrate, G. J. Systematic Approximations to the Optimized Effective Potential: Application to Orbital-Density-Functional Theory. *Physical Review A* **1992**, *46* (9), 5453–5458. https://doi.org/10.1103/physreva.46.5453
109. Li, Y.; Krieger, J. B.; Iafrate, G. J. Self-Consistent Calculations of Atomic Properties Using Self-Interaction-Free Exchange-Only Kohn-Sham Potentials. *Physical Review A* **1993**, *47* (1), 165–181. https://doi.org/10.1103/physreva.47.165
110. Garza, J.; Nichols, J. A.; Dixon, D. A. The Optimized Effective Potential and the Self-Interaction Correction in Density Functional Theory: Application

to Molecules. *The Journal of Chemical Physics* **2000**, *112* (18), 7880–7890. https://doi.org/10.1063/1.481421

111. Galvan, M.; Vela, A.; Gazquez, J. L. Chemical Reactivity in Spin-Polarized Density Functional Theory. *The Journal of Physical Chemistry* **1988**, *92* (22), 6470–6474. https://doi.org/10.1021/j100333a056
112. Chamorro, E.; Pérez, P. Condensed-to-Atoms Electronic Fukui Functions Within the Framework of Spin-Polarized Density-Functional Theory. *The Journal of Chemical Physics* **2005**, *123* (11), 114107. https://doi.org/10.1063/1.2033689
113. Wu, Q.; Yang, W. Empirical Correction to Density Functional Theory for van Der Waals Interactions. *The Journal of Chemical Physics* **2002**, *116* (2), 515–524. https://doi.org/10.1063/1.1424928
114. Zimmerli, U.; Parrinello, M.; Koumoutsakos, P. Dispersion Corrections to Density Functionals for Water Aromatic Interactions. *The Journal of Chemical Physics* **2004**, *120* (6), 2693–2699. https://doi.org/10.1063/1.1637034
115. Grimme, S. Accurate Description of van Der Waals Complexes by Density Functional Theory Including Empirical Corrections. *Journal of Computational Chemistry* **2004**, *25* (12), 1463–1473. https://doi.org/10.1002/jcc.20078
116. Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. A Consistent and Accurate Ab Initio Parametrization of Density Functional Dispersion Correction (DFT-d) for the 94 Elements h-Pu. *The Journal of Chemical Physics* **2010**, *132* (15), 154104. https://doi.org/10.1063/1.3382344
117. Otero-de-la-Roza, A.; Johnson, E. R. Non-Covalent Interactions and Thermochemistry Using XDM-Corrected Hybrid and Range-Separated Hybrid Density Functionals. *The Journal of Chemical Physics* **2013**, *138* (20), 204109. https://doi.org/10.1063/1.4807330

11.3 CIS, TDHF, TDDFT

11.3.1 Overview

NWChem supports a spectrum of single excitation theories for vertical excitation energy calculations, namely, configuration interaction singles (CIS)[1], time-dependent Hartree-Fock (TDHF or also known as random-phase approximation RPA), time-dependent density functional theory (TDDFT)[2], and Tamm-Dancoff approximation[3] to TDDFT. These methods are implemented in a single framework that invokes Davidson's trial vector algorithm (or its modification for a non-Hermitian eigenvalue problem). The capabilities of the module are summarized as follows:

- Vertical excitation energies (valence and core),
- Spin-restricted singlet and triplet excited states for closed-shell systems,
- Spin-unrestricted doublet, etc., excited states for open-shell systems,
- Tamm-Dancoff and full time-dependent linear response theories,
- Davidson's trial vector algorithm,
- Symmetry (irreducible representation) characterization and specification,
- Spin multiplicity characterization and specification,
- Transition moments and oscillator strengths,
- Analytical first derivatives of vertical excitation energies with a selected set of exchange-correlation functionals (see TDDFT gradients documentation for further information),
- Numerical second derivatives of vertical excitation energies,
- Disk-based and fully incore algorithms,
- Multiple and single trial-vector processing algorithms,
- Frozen core and virtual approximation,
- Asymptotically correct exchange-correlation potential by van Leeuwen and Baerends[4],
- Asymptotic correction by Casida and Salahub[5],
- Asymptotic correction by Hirata, Zhan, Aprà, Windus, and Dixon[6].

These are very effective way to rectify the shortcomings of TDDFT when applied to Rydberg excited states (see below).

11.3.2 Performance of CIS, TDHF, and TDDFT methods

The accuracy of CIS and TDHF for excitation energies of closed-shell systems are comparable to each other, and are normally considered a zeroth-order description of the excitation process. These methods are particularly well balanced in describing Rydberg excited states, in contrast to TDDFT. However, for open-shell systems, the errors in the CIS and TDHF excitation energies are often excessive, primarily due to the multi-determinantal character of the ground and excited state wave functions of open-shell systems in a HF reference. The scaling of the computational cost of a CIS or TDHF calculation per state with respect to the system size is the same as that for a HF calculation for the ground state, since the critical step of the both methods are the Fock build, namely, the contraction of two-electron integrals with density matrices. It is usually necessary to include two sets of diffuse exponents in the basis set to properly account for the diffuse Rydberg excited states of neutral species.

The accuracy of TDDFT may vary depending on the exchange-correlation functional. In general, the exchange-correlation functionals that are widely used today and are implemented in NWChem work well for low-lying valence excited states. However, for high-lying diffuse excited states and Rydberg excited states in particular, TDDFT employing these conventional functionals breaks down and the excitation energies are substantially underestimated. This is because of the fact that the exchange-correlation potentials generated from these functionals decay too rapidly (exponentially) as opposed to the slow *-1/r* asymptotic decay of the true potential. A rough but useful index is the negative of the highest occupied KS orbital energy; when the calculated excitation energies become close to this threshold, these numbers are most likely underestimated relative to experimental results. It appears that TDDFT provides a better-balanced description of radical excited states. This may be traced to the fact that, in DFT, the ground state wave function is represented well as a single KS determinant, with less multi-determinantal character and less spin contamination, and hence the excitation thereof is described well as a simple one electron transition. The computational cost per state of TDDFT calculations scales as the same as the ground state DFT calculations, although the prefactor of the scaling may be much greater in the former.

A very simple and effecive way to rectify the TDDFT's failure for Rydberg excited states has been proposed by Tozer and Handy[7] and by Casida and Salahub[5]. They proposed to splice a *-1/r* asymptotic tail to an exchange-correlation potential that does not have the correct asymptotic behavior. Because the approximate exchange-correlation potentials are too shallow everywhere, a negative constant must be added to them before they can be spliced to the *-1/r* tail seamlessly in a region that is not sensitive to chemical effects or to the long-range behavior. The negative constant or the shift is usually taken to be the difference of the HOMO energy from the true ionization potential, which can be obtained either from experiment or from a ΔSCF calculation. Recently, we proposed a new, expedient, and self-contained asymptotic correction that does not require an ionization potential (or shift) as an external parameter from a separate calculation. In this scheme, the shift is computed by a semi-empirical formula proposed by Zhan, Nichols, and Dixon[6]. Both Casida-Salahub scheme and this new asymptotic correction scheme give considerably improved (Koopmans type) ionization potentials and Rydberg excitation energies. The latter, however, supply the shift by itself unlike to former.

11.3.3 Input syntax

The module is called TDDFT as time-dependent density functional theory employing a hybrid HF-DFT functional encompasses all of the above-mentioned methods implemented. To use this module, one needs to specify TDDFT on the task directive, e.g.,

```
TASK TDDFT ENERGY
```

for a single-point excitation energy calculation, and

```
TASK TDDFT OPTIMIZE
```

for an excited-state geometry optimization (and perhaps an adiabatic excitation energy calculation), and

```
TASK TDDFT FREQUENCIES
```

for an excited-state vibrational frequency calculation. The TDDFT module first invokes DFT module for a ground-state calculation (regardless of whether the calculations uses a HF reference as in CIS or TDHF or a DFT functional), and hence there is no need to perform a separate ground-state DFT calculation prior to calling a TDDFT task. When no second

argument of the task directive is given, a single-point excitation energy calculation will be assumed. For geometry optimizations, it is usually necessary to specify the target excited state and its irreducible representation it belongs to. See the subsections TARGET and TARGETSYM for more detail.

Individual parameters and keywords may be supplied in the TDDFT input block. The syntax is:

```
TDDFT
 [(CIS||RPA) default RPA]
 [NROOTS <integer nroots default 1>]
 [MAXVECS <integer maxvecs default 1000>]
 [(SINGLET||NOSINGLET) default SINGLET]
 [(TRIPLET||NOTRIPLET) default TRIPLET]
 [THRESH <double thresh default 1e-4>]
 [MAXITER <integer maxiter default 100>]
 [TARGET <integer target default 1>]
 [TARGETSYM <character targetsym default 'none'>]
 [SYMMETRY]
 [ECUT] <-cutoff energy>
 [EWIN] <-lower cutoff energy>  <-higher cutoff energy>
 [ALPHA] <integer lower orbital>  <integer upper orbital>
 [BETA] <integer lower orbital>  <integer upper orbital>
 [CIVECS]
 [GRAD, END]
 [CDSPECTRUM]
 [GIAO]
 [VELOCITY]
 [SIMPLESO]
 [ALGORITHM <integer algorithm default 0>]
 [FREEZE [[core] (atomic || <integer nfzc default 0>)] \
       [virtual <integer nfzv default 0>]]
 [PRINT (none||low||medium||high||debug)
   <string list_of_names ...>]
END
```

The user can also specify the reference wave function in the DFT input block (even when CIS and TDHF calculations are requested). See the section of Sample input and output for more details.

Since each keyword has a default value, a minimal input file will be

```
GEOMETRY
 Be 0.0 0.0 0.0
END
BASIS
 Be library 6-31G**
END
TASK TDDFT ENERGY
```

Note that the keyword for the asymptotic correction must be given in the DFT input block, since all the effects of the correction (and also changes in the computer program) occur in the SCF calculation stage. See DFT (keywords CS00 and LB94) for details.

11.3.4 Keywords of TDDFT input block

11.3.4.1 CIS and RPA: the Tamm-Dancoff approximation

These keywords toggle the Tamm-Dancoff approximation. CIS means that the Tamm-Dancoff approximation is used and the CIS or Tamm-Dancoff TDDFT calculation is requested. RPA , which is the default, requests TDHF (RPA) or TDDFT calculation.

The performance of CIS (Tamm-Dancoff TDDFT) and RPA (TDDFT) are comparable in accuracy. However, the computational cost is slightly greater in the latter due to the fact that the latter involves a non-Hermitian eigenvalue problem and requires left and right eigenvectors while the former needs just one set of eigenvectors of a Hermitian eigenvalue problem. The latter has much greater chance of aborting the calculation due to triplet near instability or other instability problems.

11.3.4.2 NROOTS: the number of excited states

One can specify the number of excited state roots to be determined. The default value for NROOTS is 1. It is advised that the users request several more roots than actually needed, since owing to the nature of the trial vector algorithm, some low-lying roots can be missed when they do not have sufficient overlap with the initial guess vectors.

11.3.4.3 MAXVECS: the subspace size

The MAXVECS keyword limits the subspace size of Davidson's algorithm; in other words, it is the maximum number of trial vectors that the calculation is allowed to hold. Typically, 10 to 20 trial vectors are needed for each excited state root to be converged. However, it need not exceed the product of the number of occupied orbitals and the number of virtual orbitals. The default value is 1000.

11.3.4.4 SINGLET and NOSINGLET: singlet excited states

SINGLET || NOSINGLET requests (suppresses) the calculation of singlet excited states when the reference wave function is closed shell. The default is SINGLET .

11.3.4.5 TRIPLET and NOTRIPLET: triplet excited states

TRIPLET || NOTRIPLET requests (suppresses) the calculation of triplet excited states when the reference wave function is closed shell. The default is TRIPLET .

11.3.4.6 THRESH: the convergence threshold of Davidson iteration

The THRESH keyword specifies the convergence threshold of Davidson's iterative algorithm to solve a matrix eigenvalue problem. The threshold refers to the norm of residual, namely, the difference between the left-hand side and right-hand side of the matrix eigenvalue equation with the current solution vector. With the default value of 1e-4, the excitation energies are usually converged to 1e-5 hartree.

11.3.4.7 MAXITER: the maximum number of Davidson iteration

It typically takes 10-30 iterations for the Davidson algorithm to get converged results. The default value for MAXITER is 100.

11.3.4.8 TARGET and TARGETSYM: the target root and its symmetry

At the moment, excited-state first geometry derivatives can be calculated analytically for a set of functionals, while excited-state second geometry derivatives are obtained by numerical differentiation. These keywords may be used to specify which excited state root is being used for the geometrical derivative calculation. For instance, when TARGET 3 and TARGETSYM a1g are included in the input block, the total energy (ground state energy plus excitation energy) of the third lowest excited state root (excluding the ground state) transforming as the irreducible representation a1g will be passed to the module which performs the derivative calculations. The default values for TARGET and TARGETSYM are 1 and none , respectively.

The keyword TARGETSYM is essential in excited state geometry optimization, since it is very common that the order of excited states changes due to the geometry changes in the course of optimization. Without specifying the TARGETSYM , the optimizer could (and would likely) be optimizing the geometry of an excited state that is different from the one the user had intended to optimize at the starting geometry. On the other hand, in the frequency calculations, TARGETSYM must be none , since the finite displacements given in the course of frequency calculations will lift the spatial symmetry of the equilibrium geometry. When these finite displacements can alter the order of excited states including the target state, the frequency calculation is not be feasible.

11.3.4.9 SYMMETRY: restricting the excited state symmetry

By adding the SYMMETRY keyword to the input block, the user can request the module to generate the initial guess vectors

transforming as the same irreducible representation as TARGETSYM. This causes the final excited state roots be (exclusively) dominated by those with the specified irreducible representation. This may be useful, when the user is interested in just the optically allowed transitions, or in the geometry optimization of an excited state root with a particular irreducible representation. By default, this option is not set. TARGETSYM must be specified when SYMMETRY is invoked.

11.3.4.10 ECUT: energy cutoff

The ECUT keyword enables restricted excitation window TDDFT (REW-TDDFT)[8]. This is an approach best suited for core excitations. By specifying this keyword only excitations from occupied states below the energy cutoff will be considered.

11.3.4.11 EWIN: energy window

The EWIN keyword enables a restricted energy window between a lower energy cutoff and a higher energy cutoff. For example, ewin -20.0 -10.0 will only consider excitations from occupied orbitals within the specified energy window

11.3.4.12 Alpha, Beta: alpha, beta orbital windows

Orbital windows can be specified using the following keywords:

```
alpha 1 4
beta 2 5
```

Here alpha excitations will be considered from orbitals 1 through 4 depending on the number of roots requested and beta excitations will be considered from orbitals 2 through 5 depending on the number of roots requested.

11.3.4.13 CIVECS: CI vectors

The CIVECS keyword will result in the CI vectors being written out. By default this is off. Please note this can be a very large file, so avoid turning on this keyword if you are calculating a very large number of roots. CI vectors are needed for excited-state gradient and transition density calculations.

11.3.4.14 GRAD: TDDFT gradients

Analytical TDDFT gradients can be calculated by specifying a grad block within the main TDDFT block

For example, the following will perform a TDDFT optimization on the first singlet excited state (S1). Note that thecivecs keyword must be specified. To perform a single TDDFT gradient, replace the optimize keyword with gradient in the task line. A complete TDDFT optimization input example is given the Sample Inputs section. A TDDFT gradients calculation can be used to calculate the density of a specific excited state. The excited stated density is written to a file with the .dmat suffix.

```
tddft
 nroots 2
 algorithm 1
 notriplet
 target 1
 targetsym a
 civecs
 grad
   root 1
 end
end
task tddft optimize
```

At the moment the following exchange-correlation functionals are supported with TDDFT gradients

```
LDA, BP86, PBE, BLYP, B3LYP, PBE0, BHLYP, CAM-B3LYP, LC-PBE, LC-PBE0, BNL, LC-wPBE, LC-wPBEh, LC-BLYP
```

11.3.4.15 CDSpectrum: optical rotation calculations

Perform optical rotation calculations. We recommend to use the GIAO keyword

11.3.4.16 VELOCITY: velocity gauge

Perform CD spectrum calculations with the velocity gauge.

11.3.4.17 SIMPLESO: simplified Spin-Orbit coupling

Perform excited states calculations with a simplied Spin-Orbit coupling that uses eigenvalues from aspin-orbit calculation, instead of a standard dft calculation.
Here is a snippet of an input example (please notice the use of molecular orbitals).

```
start au2
 geometry
 au 0 0 1
 au 0 0 -1
 symmetry d2h
end
#basis sets, ecp and so-ecp skipped  for simplicity
...
dft
 odft
 vectors output au2_noso.mos
end
task dft
dft
 vectors input au2_noso.mos output au2_so.mos
end
task sodft
dft
 odft
 vectors input u2_noso.mos
end

tddft
  simpleso au2.evals
  nroots 1
  notriplet
end

task tddft
```

11.3.4.18 ALGORITHM: algorithms for tensor contractions

There are four distinct algorithms to choose from, and the default value of 0 (optimal) means that the program makes an optimal choice from the four algorithms on the basis of available memory. In the order of decreasing memory requirement, the four algorithms are:

- ALGORITHM 1 : Incore, multiple tensor contraction,
- ALGORITHM 2 : Incore, single tensor contraction,
- ALGORITHM 3 : Disk-based, multiple tensor contraction,
- ALGORITHM 4 : Disk-based, single tensor contraction.

The incore algorithm stores all the trial and product vectors in memory across different nodes with the GA, and often decreases the MAXITER value to accommodate them. The disk-based algorithm stores the vectors on disks across different nodes with the DRA, and retrieves each vector one at a time when it is needed. The multiple and single tensor contraction refers to whether just one or more than one trial vectors are contracted with integrals. The multiple tensor contraction algorithm is particularly effective (in terms of speed) for CIS and TDHF, since the number of the direct evaluations of two-electron integrals is diminished substantially.

11.3.4.19 FREEZE: the frozen core/virtual approximation

Some of the lowest-lying core orbitals and/or some of the highest-lying virtual orbitals may be excluded in the CIS, TDHF,

and TDDFT calculations by the FREEZE keyword (this does not affect the ground state HF or DFT calculation). No orbitals are frozen by default. To exclude the atom-like core regions altogether, one may request

```
FREEZE atomic
```

To specify the number of lowest-lying occupied orbitals be excluded, one may use

```
FREEZE 10
```

which causes 10 lowest-lying occupied orbitals excluded. This is equivalent to writing

```
FREEZE core 10
```

To freeze the highest virtual orbitals, use the virtual keyword. For instance, to freeze the top 5 virtuals

```
FREEZE virtual 5
```

11.3.4.20 TRIALS: restart

Setting the keyword trials restart the calculation from the trials vector of a previous run.

```
trials
```

11.3.4.21 PRINT: output verbosity

The PRINT keyword changes the level of output verbosity. One may also request some particular items in the table below.

Item	Print Level	Description
“timings”	high	CPU and wall times spent in each step
“trial vectors”	high	Trial CI vectors
“initial guess”	debug	Initial guess CI vectors
“general information”	default	General information
“xc information”	default	HF/DFT information
“memory information”	default	Memory information
“convergence”	debug	Convergence
“subspace”	debug	Subspace representation of CI matrices
“transform”	debug	MO to AO and AO to MO transformation of CI vectors
“diagonalization”	debug	Diagonalization of CI matrices
“iteration”	default	Davidson iteration update
“contract”	debug	Integral transition density contraction
“ground state”	default	Final result for ground state
“excited state”	low	Final result for target excited state

Printable items in the TDDFT modules and their default print levels.

11.3.5 Sample input

The following is a sample input for a spin-restricted TDDFT calculation of singlet excitation energies for the water molecule at the B3LYP/6-31G*.

```
START h2o
TITLE "B3LYP/6-31G* H2O"
GEOMETRY
 O     0.00000000     0.00000000     0.12982363
 H     0.75933475     0.00000000    -0.46621158
 H    -0.75933475     0.00000000    -0.46621158
END
BASIS
 * library 6-31G*
END
DFT
 XC B3LYP
END
TDDFT
 RPA
 NROOTS 20
END
TASK TDDFT ENERGY
```

To perform a spin-unrestricted TDHF/aug-cc-pVDZ calculation for the CO+ radical,

```
START co
title "TDHF/aug-cc-pVDZ CO+"
charge 1
geometry
 c  0.0  0.0  0.0
 o  0.0  0.0  1.5
 symmetry c2v # enforcing abelian symmetry
end
basis
 * library aug-cc-pvdz
end
dft
 xc hfexch
 mult 2
end
task dft optimize
tddft
 rpa
 nroots 5
end
task tddft energy
```

A geometry optimization followed by a frequency calculation for an excited state is carried out for BF at the CIS/6-31G* level in the following sample input.

```
start bf
title "CIS/6-31G* BF optimization frequencies"
geometry
 b 0.0 0.0 0.0
 f 0.0 0.0 1.2
 symmetry c2v # enforcing abelian symmetry
end
basis
 * library 6-31g*
end
dft
 xc hfexch
end
tddft
 cis
 nroots 3
 notriplet
 target 1
  civecs
  grad
    root 1
  end
end
task tddft optimize
task tddft frequencies
```

TDDFT with an asymptotically corrected SVWN exchange-correlation potential. Casida-Salahub scheme has been used with the shift value of 0.1837 a.u. supplied as an input parameter.

```
START tddft_ac_co
GEOMETRY
 O 0.0 0.0  0.0000
 C 0.0 0.0  1.1283
 symmetry c2v # enforcing abelian symmetry
END
BASIS SPHERICAL
 C library aug-cc-pVDZ
 O library aug-cc-pVDZ
END
DFT
 XC Slater VWN_5
 CS00 0.1837
END
TDDFT
 NROOTS 12
END
TASK TDDFT ENERGY
```

TDDFT with an asymptotically corrected B3LYP exchange-correlation potential. Hirata-Zhan-Apra-Windus-Dixon scheme has been used (this is only meaningful with B3LYP functional).

```
START tddft_ac_co
GEOMETRY
 O 0.0 0.0  0.0000
 C 0.0 0.0  1.1283
 symmetry c2v # enforcing abelian symmetry
END
BASIS SPHERICAL
 C library aug-cc-pVDZ
 O library aug-cc-pVDZ
END
DFT
 XC B3LYP
 CS00
END
TDDFT
 NROOTS 12
END
TASK TDDFT ENERGY
```

TDDFT for core states. The following example illustrates the usage of an energy cutoff and energy and orbital windows[8]:

```
echo
start h2o_core
memory 1000 mb
geometry units au noautosym noautoz
  O 0.00000000     0.00000000     0.22170860
  H 0.00000000     1.43758081    -0.88575430
  H 0.00000000    -1.43758081    -0.88575430
end
basis
 O library 6-31g*
 H library 6-31g*
end
dft
 xc beckehandh
 print "final vector analysis"
end
task dft
tddft
 ecut -10
 nroots 5
 notriplet
 thresh 1d-03
end
task tddft
tddft
 ewin -20.0 -10.0
 cis
 nroots 5
 notriplet
 thresh 1d-03
end
task tddft
dft
 odft
 mult 1
 xc beckehandh
 print "final vector analysis"
end
task dft
tddft
 alpha 1 1
 beta 1 1
 cis
 nroots 10
 notriplet
 thresh 1d-03
end
task tddft
```

TDDFT optimization with LDA of Pyridine with the 6-31G basis

```
echo
start tddftgrad_pyridine_opt
title "TDDFT/LDA geometry optimization of Pyridine with 6-31G"
geometry nocenter
 N     0.00000000    0.00000000    1.41599295
 C     0.00000000   -1.15372936    0.72067272
 C     0.00000000    1.15372936    0.72067272
 C     0.00000000   -1.20168790   -0.67391011
 C     0.00000000    1.20168790   -0.67391011
 C     0.00000000    0.00000000   -1.38406147
 H     0.00000000   -2.07614628    1.31521089
 H     0.00000000    2.07614628    1.31521089
 H     0.00000000    2.16719803   -1.19243296
 H     0.00000000   -2.16719803   -1.19243296
 H     0.00000000    0.00000000   -2.48042299
 symmetry c1
end
basis spherical
* library "6-31G"
end
driver
  clear
  maxiter 100
end
dft
  iterations 500
  grid xfine
end
tddft
  nroots 2
  algorithm 1
  notriplet
  target 1
  targetsym a
  civecs
  grad
    root 1
  end
end
task tddft optimize
```

TDDFT calculation followed by a calculation of the transition density for a specific excited state using the DPLOT block

```
echo
start h2o-td
title h2o-td

charge 0
geometry units au
symmetry group c1
 O    0.00000000000000     0.00000000000000     0.00000000000000
 H    0.47043554760291     1.35028113274600     1.06035416576826
 H   -1.74335410533480    -0.23369304784300     0.27360785442967
end
basis "ao basis"
* library "Ahlrichs pVDZ"
end
dft
 xc bhlyp
 grid fine
 direct
 convergence energy 1d-5
end
tddft
 rpa
 nroots 5
 thresh 1d-5
 singlet
 notriplet
 civecs
end
task tddft energy
dplot
 civecs h2o-td.civecs_singlet
 root 2
 LimitXYZ
  -3.74335 2.47044 50
  -2.23369 3.35028 50
  -2 3.06035 50
   gaussian
   output root-2.cube
end
task dplot
```

TDDFT protocol for calculating the valence-to-core (1s) X-ray emission spectrum[10]

1. Calculate the neutral ground state.
2. Calculate a full core hole (FCH) ionized state self-consistently, where the 1s core orbital of the absorbing center is swapped with a virtual orbital. Use the maximum overlap constraint to prevent core hole collapse during the FCH calculation.
3. Perform a LR-TDDFT calculation within the TDA is performed with the FCH ionized state as reference.
4. Final spectra is produced by taking the absolute value of the negative eigenvalues.

11.3.6 Spectrum parser

A Python script is available for parsing NWChem output for TDDFT/vspec excitation energies, and optionally Lorentzian broadenening the spectra . The nw_spectrum.py file can be found at https://raw.githubusercontent.com/nwchemgit/nwchem/master/contrib/parsers/nw_spectrum.py

```
Usage: nw_spectrum.py [options]

Reads NWChem output from stdin, parses for the linear response TDDFT or DFT
vspec excitations, and prints the absorption spectrum to stdout.  It will
optionally broaden peaks using a Lorentzian with FWHM of at least two
energy/wavelength spacings.  By default, it will automatically determine data
format (tddft or vspec) and generate a broadened spectrum in eV.

Example:

   nw_spectrum -b0.3 -p5000 -wnm < water.nwo > spectrum.dat

Create absorption spectrum in nm named "spectrum.dat" from the NWChem output
file "water.nwo" named spectrum.dat with peaks broadened by 0.3 eV and 5000
points in the spectrum.


Options:
  -h, --help            show this help message and exit
  -f FMT, --format=FMT  data file format: auto (default), tddft, vspec, dos
  -b WID, --broad=WID   broaden peaks (FWHM) by WID eV (default 0.1 eV)
  -n NUM, --nbin=NUM    number of eigenvalue bins for DOS calc (default 20)
  -p NUM, --points=NUM  create a spectrum with NUM points (default 2000)
  -w UNT, --units=UNT   units for frequency:  eV (default), au, nm
  -d STR, --delim=STR   use STR as output separator (four spaces default)
  -x, --extract         extract unbroadened roots; do not make spectrum
  -C, --clean           clean output; data only, no header or comments
  -c CHA, --comment=CHA
                        comment character for output ('#' default)
  -v, --verbose         echo warnings and progress to stderr
```

11.3.7 References

1. J. B. Foreman, M. Head-Gordon, J. A. Pople, and M. J. Frisch, *J. Phys. Chem.* **96**, 135 (1992), DOI: 10.1021/j100180a030
2. C. Jamorski, M. E. Casida, and D. R. Salahub, *J. Chem. Phys.* **104**, 5134 (1996), DOI: 10.1063/1.471140; R. Bauernschmitt and R. Ahlrichs, *Chem. Phys. Lett.* **256**, 454 (1996), DOI: 10.1016/0009-2614(96)00440-X; R. Bauernschmitt, M. Häser, O. Treutler, and R. Ahlrichs, *Chem. Phys. Lett.* **264**, 573 (1997), DOI:10.1016/S0009-2614(96)01343-7.
3. S. Hirata and M. Head-Gordon, *Chem. Phys. Lett.* **314**, 291 (1999). DOI:10.1016/S0009-2614(99)01149-5
4. R. van Leeuwen and E. J. Baerends, *Phys. Rev. A* **49**, 2421 (1994), DOI:10.1103/PhysRevA.49.2421
5. M. E. Casida, C. Jamorski, K. C. Casida, and D. R. Salahub, *J. Chem. Phys.* **108**, 4439 (1998), DOI: 10.1063/1.475855
6. S. Hirata, C.-G. Zhan, E. Aprà, T. L. Windus, and D. A. Dixon, *J. Phys. Chem. A* **107**, 10154 (2003). DOI:10.1021/jp035667x
7. D. J. Tozer and N. C. Handy, *J. Chem. Phys.* **109**, 10180 (1998), DOI: 10.1063/1.477711
8. K. Lopata, B. E. Van Kuiken, M. Khalil, N. Govind, “Linear-Response and Real-Time Time-Dependent Density Functional Theory Studies of Core-Level Near-Edge X-Ray Absorption”, *J. Chem. Theory Comput.*, 2012, **8** (9), pp 3284–3292, DOI:10.1021/ct3005613
9. D. W. Silverstein, N. Govind, H. J. J. van Dam, L. Jensen, “Simulating One-Photon Absorption and Resonance Raman Scattering Spectra Using Analytical Excited State Energy Gradients within Time-Dependent Density Functional Theory” *J. Chem. Theory Comput.*, 2013, **9** (12), pp 5490–

5503, DOI:10.1021/ct4007772

10. Y. Zhang, S. Mukamel, M. Khalil, N. Govind, “Simulating Valence-to-Core X-ray Emission Spectroscopy of Transition Metal”, *J. Chem. Theory Comput.*, 2015, **11** (12), pp 5804–5809, DOI:10.1021/acs.jctc.5b00763

11.4 Real-time TDDFT

11.4.1 Overview

Real-time time-dependent density functional theory (RT-TDDFT) is a DFT-based approach to electronic excited states based on integrating the time-dependent Kohn-Sham (TDKS) equations in time. The theoretical underpinnings, strengths, and limitations are similar to traditional linear-response (LR) TDDFT methods, but instead of a frequency domain solution to the TDKS equations, RT-TDDFT yields a full time-resolved, potentially non-linear solution. Real-time simulations can be used to compute not only spectroscopic properties (e.g., absorption spectra, polarizabilites, etc), but also the time and space-resolved electronic response to arbitrary external stimuli (e.g., electron charge dynamics after laser excitation). For theoretical and computational details, please refer to the following paper:

1. K. Lopata, N. Govind,”Modeling Fast Electron Dynamics with Real-Time Time-Dependent Density Functional Theory: Application to Small Molecules and Chromophores”, J. Chem. Theory Comput., 7, 1344 (2011) DOI:10.1021/ct200137z

This functionality is built on the Gaussian basis set DFT module, and will work for closed-shell (spin-restricted) and open-shell (spin unrestricted) calculations with essentially any combination of basis set and exchange-correlation functional in NWChem. The current implementation assumes frozen nuclei (Born-Oppenheimer approximation).

In a nutshell, running a RT-TDDFT calculation takes the following form:

1. Compute ground state density matrix with DFT module
2. Propagate density matrix using RT-TDDFT
3. Post-process resulting time-dependent observables (e.g., dipole moment)

11.4.2 Units

Unless specified otherwise, all inputs and outputs are in **atomic units**. Some useful conversions are:

Quantity	Conversion
Time	1 au = 0.02419 fs
Length	1 au = 0.5292 A
Energy	1 au = 27.2114 eV
Electric field	1 au = 514.2 V/nm
Dipole moment	1 au = 2.542 D

11.4.3 Syntax

The charge, geometry, basis set, and DFT options are all specified as normal, using their respective syntax. Real-time TDDFT parameters are supplied in the `RT_TDDFT` block (note, nothing is case-sensitive), with all possible options summarized below, and each discussed in detail afterwards.

```
RT_TDDFT
  [TMAX <double default 1000>]
  [DT <double default 0.1>]
  [TAG <string default "<rt_tddft>: "]
  [LOAD (scf || vectors <string>)]
  [NCHECKS <integer default 10>]
  [NPRINTS (* || <integer>)]
  [NRESTARTS (* || <integer>)]
  [TOLERANCES (zero <double default 1e-8> || series <double default 1e-10> || interpol <double default 1e-7>)]
  [PROPAGATOR (euler || rk4 || magnus) default magnus]
  [EXP (diag || pseries)]
  [PROF]
  [NOPROP]
  [STATIC]
  [PRINT (*, dipole, quadrupole, field, moocc, field, energy, cputime, charge, convergence, s2)]
  [EXCITE <string geomname>` with <string fieldname>]
  [FIELD]
    ...
  [END]
  [VISUALIZATION]
    ...
  [END]
  [LOAD RESTART]
END
```

11.4.3.1 TMAX: Simulation time

This option specifies the maximum time (in au) to run the simulation before stopping, which must be a positive real number. In practice, you can just stop the simulation early, so in most cases it is simplest to just set this to a large value to ensure you capture all the important dynamics (see the example of resonant ultraviolet excitation of water). For most valence excitations, for example, 1000 au is overkill so you might want to automatically stop at 500 au:

```
rt_tddft
  ...
  tmax 500.0
  ...
end
```

11.4.3.2 DT: Time step

This specifies the electronic time step for time integration. A larger time step results in a faster simulation, but there are two issues which limit the size of the time step. First, the integration algorithm can become unstable and/or inaccurate for larger time steps, which depends on the integrator used. Very roughly speaking, the second order Magnus integrator should be reliable for time steps up to 0.2 au. Second, you must choose a time step small enough to capture the oscillations of interest, i.e., to resolve an excitation of energy ω, your time step needs to be smaller than π/ω, and typically a tenth of that for accuracy. For example, to capture high energy oscillations such as core-level excitations (e.g., ω = 50au) you might want a relative small time step:

```
rt_tddft
  ...
  dt 0.01
  ...
end
```

It is always good practice to check that your results are independent of time step.

11.4.3.3 TAG: Output label

This option sets a label for the output for convenient parsing (e.g., with “grep”). Every output line with time-dependent data will begin with this string (set to `<rt_tddft>:` by default). For example setting:

```
rt_tddft
  ...
  tag "nameofrun"
  ...
end
```

will result in outputs that look like:

```
...
nameofrun      2.20000  -7.589146713114E+001     # Etot
...
```

11.4.3.4 NCHECKS: Number of run-time check points

This option takes an integer number (default 10), or `*` which means every step, which sets the total of number of run-time checkpoints, where various sanity checks are made such as symmetries, idempotency, traces, etc. These checks are not terribly efficient (e.g., involve re-building the TD Fock matrix) so excessive checking will slow down execution.

```
rt_tddft
  ...
  nchecks 10
  ...
end
```

11.4.3.5 NPRINTS: Number of print points

This sets the number of print points, i.e., the total number of time-dependent observables (e.g., dipole, charge, energy) that are computed and printed. It either takes an integer number or `*` which means every time step (this is the default). Since there is no appreciable cost to computing and printing these quantities, there is usually no need to change this from `*`.

```
rt_tddft
  ...
  nprints *
  ...
end
```

11.4.3.6 NRESTARTS: Number of restart checkpoints

This sets the number of run-time check points where the time-dependent complex density matrix is saved to file, allowing the simulation to be restarted) from that point. By default this is set to 0. There is no significant computational cost to restart checkpointing, but of course there is some disk I/O cost (which may become somewhat significant for larger systems). For example, in the following example there will be 100 restart points, which corresponds to 1 backup every 100 time steps.

```
rt_tddft
  ...
  tmax 1000.0
  dt 0.1
  nrestarts 100
  ...
end
```

11.4.3.7 TOLERANCES: Controlling numerical tolerances

This option controls various numerical tolerances:

- `zero` : threshold for checks that quantities are zero, e.g., in symmetry checks (default 1e-8)
- `series` : numerical convergence for series, e.g., matrix exponentiation (default 1e-10)
- `interpol` : numerical convergence for interpolation, e.g., in Magnus propagator (default 1e-7)

Occasionally it is useful to loosen the interpolation tolerances if the Magnus interpolator requires an excessive amount of steps; usually this will not impact accuracy. For example, this sets the tolerances to their defaults with a looser interpolation tolerance:

```
rt_tddft
  ...
  tolerances zero 1d-8 series 1d-10 interpol 1e-5
end
```

11.4.3.8 PROPAGATOR: Selecting the integrator method

This selects the propagator (i.e., time integrator) method. Possible choices include `euler` for Euler integration (terrible, you should never use this), `rk4` for 4th order Runge-Kutta, and `magnus` for 2nd order Magnus with self-consistent interpolation. In virtually all cases Magnus is superior in terms of stability. Euler or rk4 are perhaps only useful for debugging or simplicity (e.g., for code development).

```
rt_tddft
  ...
  propagator magnus
  ...
end
```

11.4.3.9 EXP: Selecting the matrix exponentiation method

This selects the method for exponentiation matrices. For now this can either be `pseries` for a contractive power series or `diag` for diagonalization. In general the power series (default) is faster.

```
rt_tddft
  ...
  exp diag
  ...
end
```

11.4.3.10 PROF: Run-time profiling

This turns on run-time profiling, which will display time spent in each component of the code (e.g., building components of the TD Fock matrix, properites, etc). This slows code down slightly and results in very verbose output.

```
rt_tddft
  ...
  prof
  ...
end
```

11.4.3.11 NOPROP: Skipping propagation

This options causes the RT-TDDFT module skip propagation, i.e., to initialize and finalize. For now this is largely useful for skipping to visualization post-processing without having to re-run a simulation.

```
rt_tddft
  ...
  noprop
  ...
end
```

11.4.3.12 STATIC: Force static Fock matrix

This option sets the static Fock matrix flag, meaning the time-dependent Fock matrix will not be recalculated at each time, but instead use the *t=0* value. This will drastically increase the simulation speed since the bulk of the work is spent rebuilding the TD Fock matrix, but will give non-physical results. For example, using `static` to compute an absorption spectrum will result in excitations corresponding to the raw eigenvalue differences without electron-hole relaxation. This option has few uses besides dry-runs and debugging.

```
rt_tddft
  ...
  static
  ...
end
```

11.4.3.13 PRINT: Selecting time-dependent quantities to be printed

This sets the various time-dependent properties that are to be computed and printed at each print point. Note that for many of these options, the values are computed and printed for each geometry specified in the input deck, not only the active one (i.e., the one set using `set geometry ...` in the input deck). Possible choices are:

- `dipole` : Dipole moment
- `quadrupole` : Quadrupole moment
- `field` : External (applied) electric field
- `moocc` : Molecular orbital occupations
- `energy` : Components of system energy (e.g., core, XC, total, etc)
- `cputime` : CPU time taken in simulation so far (useful for checking scaling)
- `charge` : Electronic charge (computed from density matrix, not from the XC grid)
- `convergence` : Convergence information (e.g., from Magnus)
- `s2` : value (openshell only)
- `*` : Print all quantities

The defaults correspond to:

```
rt_tddft
  ...
  print dipole field energy convergence
  ...
end
```

11.4.3.14 FIELD: Sub-block for specifying external electric fields

This sub-block is used to specify external electric fields, each of which must be given a unique name. Numerous fields can be specified, but each will applied to the system only if an appropriate excitation rule is set. There are a few preset field types; others would have to be manually coded. Note the names are arbitrary, but chosen here to be descriptive:

```
field "kick"
  type delta      # E(t=0) = max; E(t>0) = 0
  polarization x  # = x, y, z
  max 0.0001      # maximum value of electric field
  spin total      # = alpha, beta, total (only valid for open-shell)
end


field "gpulse"
  type gaussian   # Gaussian enveloped quasi-monochromatic pulse: E(t) = max * exp( -(t-t0)^2 / 2s^2) * sin(w0*t + phi0)
  polarization x  # = x, y, z
  frequency 0.12  # frequency of laser pulse in au (e.g., 0.12 au = 3.27 eV)
  phase 0.0       # phase shift of laser pulse (in rad)
  center 200.0    # center of Gaussian envelope (in au time)
  width 50.0      # width of Gaussian pulse (in au time)
  max 0.0001      # maximum value of electric field
  spin total      # = alpha, beta, total (only valid for open-shell)
end
```

```
field "hpulse"
  type hann       # sin^2 (Hann) enveloped quasi-monochromatic pulse
  polarization x  # = x, y, z
  frequency 0.12  # frequency of laser pulse in au (e.g., 0.12 au = 3.27 eV)
  center 200.0    # center of Hann envelope (in au time)
  width 50.0      # width of Hann pulse (in au time)
  max 0.0001      # maximum value of electric field
  spin total      # = alpha, beta, total (only valid for open-shell)
end


field "resonant"
  type cw         # monochromatic continuous wave
  frequency 0.12  # frequency of laser pulse in au (e.g., 0.12 au = 3.27 eV)
  polarization x  # = x, y, z
  max 0.0001      # maximum value of electric field
  spin total      # = alpha, beta, total (only valid for open-shell)
end
```

11.4.3.15 EXCITE: Excitation rules

This sets the rules for applying external fields to the system. It takes the form `excite <geom> with <field>` , where `<geom>` is the name of a geometry fragment (defaults to “geometry” which is the default geometry name), and `<field>` is the name of a field structure. Assuming, for example, you have defined a field name `kick` this option takes the form (note that quotes are optional and shown for clarity):

```
rt_tddft
  ...
  excite "geometry" with "kick"
  ...
end
```

11.4.3.16 VISUALIZATION: Sub-block for controlling 3D visualization

This block is used to control visualization of the electronic charge density, which is typically used during a resonant excitation. This is a two stage process. During the propagation, a series of density matrices will be dumped to file (see options below). After propagation, if the `dplot` option is set, the code will read in options from a separate `DPLOT` block and convert the density matrix snapshots to a corresponding series of real-space charge density `cube` files.

```
visualization
  tstart 0.0       # start visualization at this time
  tend 100.0       # stop visualization at this time
  treference 0.0   # subtract density matrix at this time (useful for difference densities)
  dplot            # post-process density matrices into cube files after propagation
end
```

11.4.3.17 LOAD RESTART

This keyword needs to be added to restart a calculation. In the following example, the calculation will restart from the previous calculation and extend the run to the new `tmax`

```
rt_tddft
  ...
  tmax 10.0       # new end time
  load restart
end
```

11.4.3.18 MOCAP: Sub-block for molecular orbital complex absorbing potential

The K. Lopata and N. Govind, “Near and Above Ionization Electronic Excitations with Non-Hermitian Real-Time Time-Dependent Density Functional Theory”, Journal of Chemical Theory and Computation 9 (11), 4939-4946 (2013) DOI:10.1021/ct400569s

```
mocap
  maxval 100.0     # clamp CAP at this value (in Ha)
  emin 0.5         # any MO with eigenvalue >= 0.5 Ha will have CAP applied to it
  prefac 1.0       # prefactor for exponential
  expconst 1.0     # exponential constant for CAP
  on               # turn on/off CAP
  nochecks         # disable checks for speed
  noprint          # don't print CAP value
end
```

11.4.3.18.1 MAXVAL: Exponential Maximum Value

11.4.3.18.2 EMIN: Vacuum Energy Level

11.4.3.18.3 ON/OFF: Turn on/off CAP

The default value is `off`, i.e. no complex absorbing potential is applied.

11.4.4 Worked Examples

11.4.4.1 Absorption spectrum of water

Here, we compute the 6-31G/TD-PBE0 absorption spectrum of gas-phase water using RT-TDDFT. In the weak-field limit, these results are essentially identical to those obtained via traditional linear-response TDDFT. Although it involves significantly more work do use RT-TDDFT in this case, for very large systems with many roots a real-time approach becomes advantageous computationally and also does not suffer from algorithm stability issues.

To compute the absorption, we find the ground state of the system and then subject it to three delta-function excitations (x,y,z), which simultaneously excites all electronic modes of that polarization. The three resulting dipole moments are then Fourier transformed to give the frequency-dependent linear polarizability, and thus the absorption spectrum. The full input deck is RT_TDDFT_h2o_abs.nw and the corresponding output is RT_TDDFT_h2o_abs.nwo.gz.

```
title "Water TD-PBE0 absorption spectrum"
echo


start water

##
## aug-cc-pvtz / pbe0 optimized
##
## Note: you are required to explicitly name the geometry
##
geometry "system" units angstroms nocenter noautoz noautosym
  O     0.00000000    -0.00001441    -0.34824012
  H    -0.00000000     0.76001092    -0.93285191
  H     0.00000000    -0.75999650    -0.93290797
end

## Note: We need to explicitly set the "active" geometry even though there is only one geom.
set geometry "system"

## All DFT and basis parameters are inherited by the RT-TDDFT code
basis
 * library 6-31G
end

dft
  xc pbe0
end

## Compute ground state of the system
task dft energy

##
## Now, we compute an x, y, and z kick simulation, which we give separate "tags" for post-processing.
##

unset rt_tddft:*
rt_tddft
  tmax 200.0
  dt 0.2

  tag "kick_x"

  field "kick"
    type delta
    polarization x
    max 0.0001
  end

  excite "system" with "kick"
 end
task dft rt_tddft

unset rt_tddft:*
rt_tddft
  tmax 200.0
  dt 0.2

  tag "kick_y"

  field "kick"
    type delta
    polarization y
    max 0.0001
  end

  excite "system" with "kick"
 end
task dft rt_tddft

unset rt_tddft:*
rt_tddft
  tmax 200.0
  dt 0.2

  tag "kick_z"

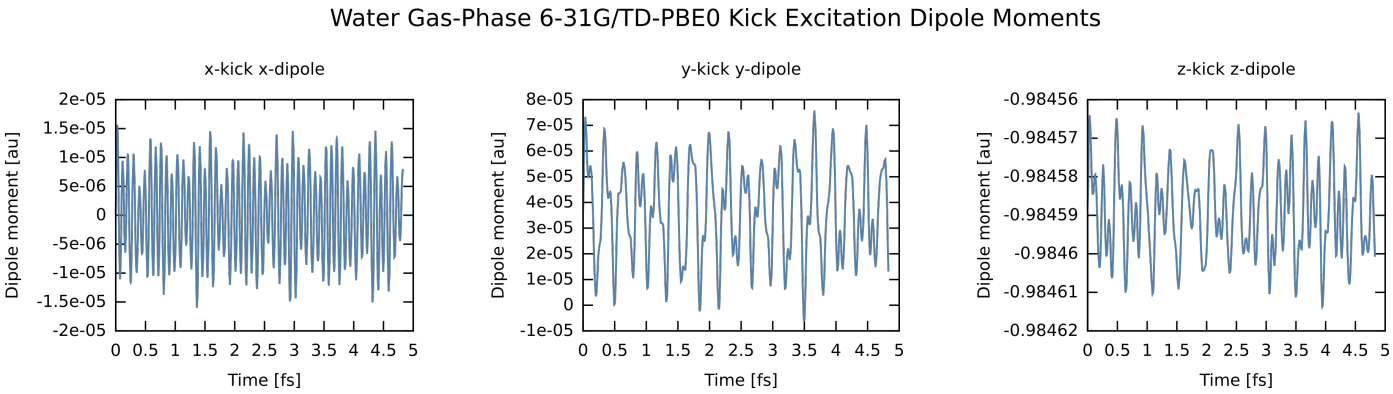
  field "kick"
    type delta
    polarization z
    max 0.0001
  end

  excite "system" with "kick"
 end
task dft rt_tddft
```

After running the simulation, we extract the x-dipole moment for the x-kick and similarly for the y and z-kicks (see “contrib/parsers” directory for this script or download here:RT_TDDFT_scripts.tgz).

```
nw_rtparse.py -xdipole -px -tkick_x h2o_abs.nwo > x.dat
nw_rtparse.py -xdipole -py -tkick_y h2o_abs.nwo > y.dat
nw_rtparse.py -xdipole -pz -tkick_z h2o_abs.nwo > z.dat
```

Note, the syntax for extracting the x polarization for the x-kick, etc. Alternatively, we could grep and cut, or whatnot. This will give use the resulting time-dependent dipole moments:



Now, we need to take the Fourier transforms of these dipole moments to yield the the x,x element of the 3x3 linear polarizability tensor, and similarly for the y,y and z,z elements. Here I am using an FFT utility, although any discrete Fourier transform will do. To accelerate convergence of the FFT, I have damped the time signals by $exp(-t/\tau)$ which results in Lorentzians with FWHM of $2/\tau$ and have also “zero padded” the data with 50000 points. This is not critical for extracting frequencies, but creates “cleaner” spectra, although care must be taken to damp sufficiently if padding to avoid artifacts (see small ripples around 23 eV in plot below). After Fourier transform, I “paste” the three files together to easily plot the absorption, which is constructed from the trace of the polarizability matrix, i.e., the sum of the imaginary parts of the FFTs of the dipole moments.

$S(\omega) = (4\pi\omega)/(3c\kappa)$Tr[Im* α(ω)*]

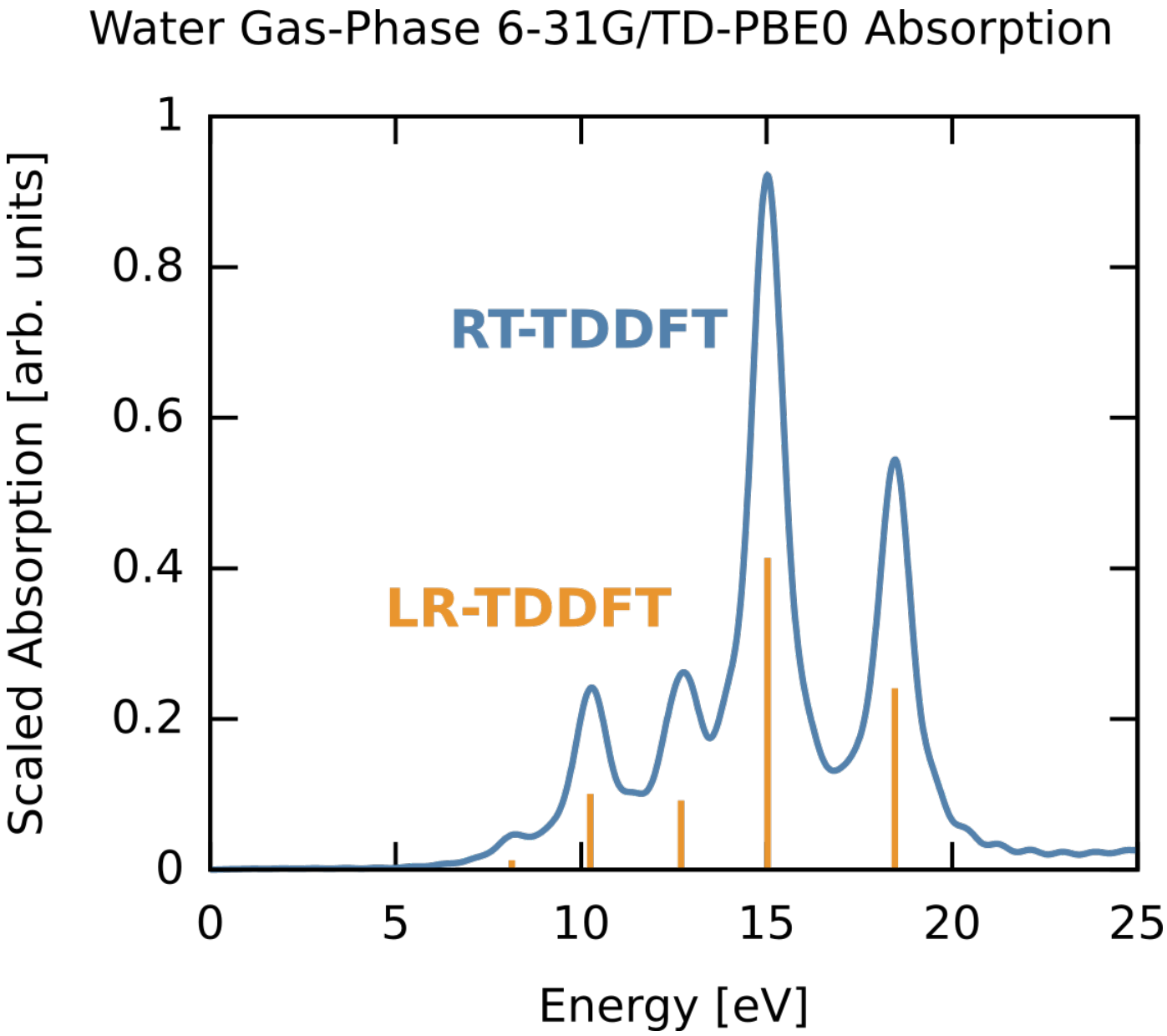
where c is the speed of light (137 in atomic units),κ is the kick electric field strength, and $\alpha(\omega)$ is the linear polarizabilty tensor computed from the Fourier transforms of the time-dependent dipole moments. For example,

```
fft1d -d50 -z -p50000 < x.dat | rotate_fft > xw.dat
fft1d -d50 -z -p50000 < y.dat | rotate_fft > yw.dat
fft1d -d50 -z -p50000 < z.dat | rotate_fft > zw.dat
paste xw.dat yw.dat zw.day > s.dat
```

Here, you can just use your favorite Fourier transform utility or analysis software, but for convenience there is also a simple GNU Octave fft1d.m utility in the “contrib/parsers” directory of the trunk or download here: RT_TDDFT_scripts.tgz Note, options are hardcoded at the moment, so the switches above are not correct instead edit the file and run (also it reads file rather than redirect from stdin). Assuming the FFT output takes the form (w, Re, Im, Abs), to plot using gnuplot we would do:

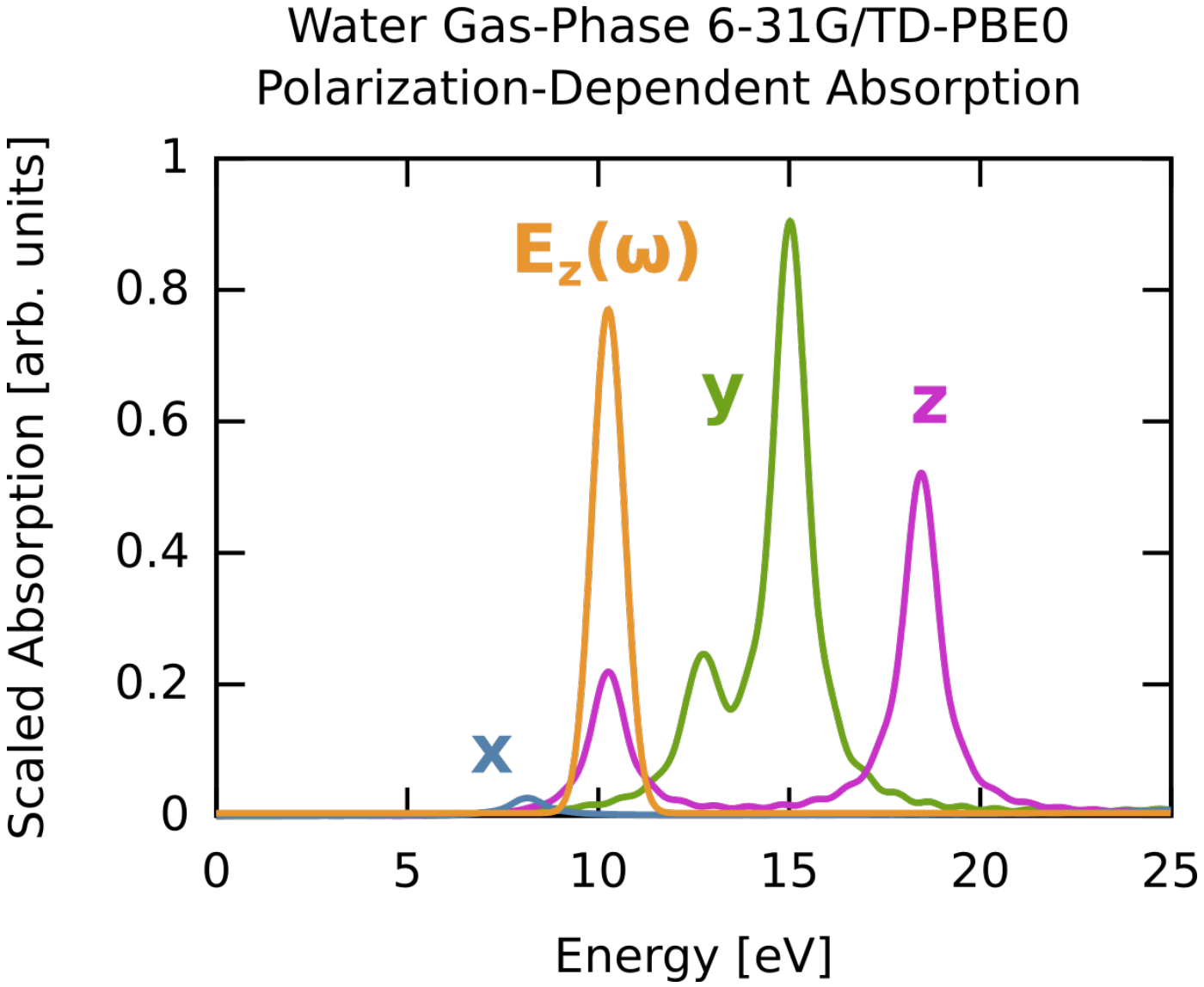
```
gnuplot> plot "s.dat" u ($1*27.2114):($1*($3+$7+$11))
```

where we have scaled by 27.2114 to output in eV instead of atomic units, and we have not properly scaled to get the absolute oscillator strengths (thus our magnitudes are in “arbitrary units”). The real-time spectrum is shown below, along with the corresponding linear-response TDDFT excitations are shown in orange for comparison. Since we are in the weak field regime, the two are identical. Note the oscillator strengths are arbitrary and scaled, if not scaled the area under each RT-TDDFT curve should integrate to the linear response oscillator strength.



11.4.4.2 Resonant ultraviolet excitation of water

In this example we compute the time-dependent electron response to a quasi-monochromatic laser pulse tuned to a particular transition. We will use the results of the previous example (6-31G/PBE0 gas-phase water). First, we consider the absorption spectrum (computed previously) but plotted for the three polarizations (x,y,z) rather then as a sum. Say we are interested in the excitation near 10 eV. We can clearly see this is a z-polarized transition (green on curve). To selectively excite this we could use a continuous wave E-field, which has a delta-function, i.e., single frequency, bandwidth but since we are doing finite simulations we need a suitable envelope. The broader the envelope in time the narrower the excitation in frequency domain, but of course long simulations become costly so we need to put some thought into the choice of our envelope. In this case the peak of interest is spectrally isolated from other z-polarized peaks, so this is quite straightforward. The procedure is outlined below, and the corresponding frequency extent of the pulse is shown on the absorption figure in orange. Note that it only covers one excitation, i.e., the field selectively excites one mode. The full input deck is RT_TDDFT_h2o_resonant.nw and the output is RT_TDDFT_h2o_resonant.nwo.gz.


Water Gas-Phase 6-31G/TD-PBE0
Polarization-Dependent Absorption
Scaled Absorption [arb. units]
Energy [eV]
0
0.2
0.4
0.6
0.8
1
0
5
10
15
20
25
E_z(ω)
y
z
x

```
title "Water TD-PBE0 resonant excitation"
echo
scratch_dir ./scratch
permanent_dir ./perm

start water

##
## aug-cc-pvtz / pbe0 optimized
##
## Note: you are required to explicitly name the geometry
##
geometry "system" units angstroms nocenter noautoz noautosym
  O     0.00000000    -0.00001441    -0.34824012
  H    -0.00000000     0.76001092    -0.93285191
  H     0.00000000    -0.75999650    -0.93290797
end

## Note: We need to explicitly set the "active" geometry even though there is only one geom.
set geometry "system"

## All DFT and basis parameters are inherited by the RT-TDDFT code
basis
 * library 6-31G
end

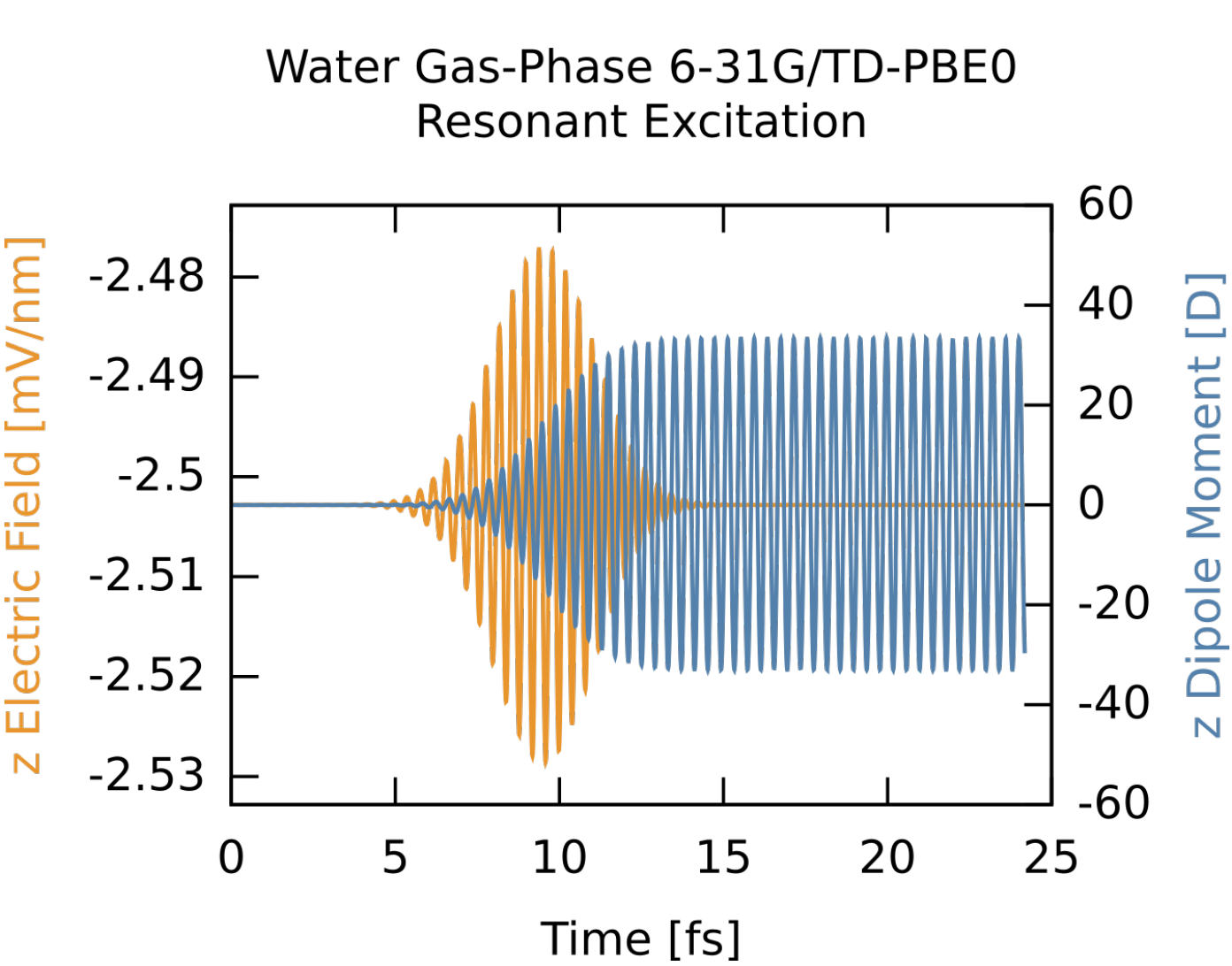
dft
 xc pbe0
end

## Compute ground state of the system
task dft energy

##
## We excite the system with a quasi-monochromatic
## (Gaussian-enveloped) z-polarized E-field tuned to a transition at
## 10.25 eV.  The envelope takes the form:
##
## G(t) = exp(-(t-t0)^2 / 2s^2)
##
## The target excitation has an energy (frequency) of w = 0.3768 au
## and thus an oscillation period of T = 2 pi / w = 16.68 au
##
## Since we are doing a Gaussian envelope in time, we will get a
## Gaussian envelope in frequency (Gaussians are eigenfunctions of a
## Fourier transform), with width equal to the inverse of the width in
## time.  Say, we want a Gaussian in frequency with FWHM = 1 eV
## (recall FWHM = 2 sqrt (2ln2) s_freq) we want an s_freq = 0.42 eV =
## 0.0154 au, thus in time we need s_time = 1 / s_time = 64.8 au.
##
## Now we want the envelope to be effectively zero at t=0, say 1e-8
## (otherwise we get "windowing" effects).  Reordering G(t):
##
## t0 = t - sqrt(-2 s^2 ln G(t))
##
## That means our Gaussian needs to be centered at t0 = 393.3 au.
##
## The total simulation time will be 1000 au to leave lots of time to
## see oscillations after the field has passed.
##
rt_tddft
  tmax 1000.0
  dt 0.2

  field "driver"
    type gaussian
    polarization z
    frequency 0.3768  # = 10.25 eV
    center 393.3
    width 64.8
    max 0.0001
  end

  excite "system" with "driver"
 end
task dft rt_tddft
```



From the time-dependent dipole moment you can see the field driving the system into a superposition of the ground state and the one excited state, which manifests as monochromatic oscillations. After the field has passed the dipole oscillations continue forever as there is no damping in the system.

11.4.4.3 Charge transfer between a TCNE dimer

Here we compute the time-dependent charge oscillations between a TCNE (tetracyanoethylene) dimer separated by 3 Angstroms, where the top molecule starts neutral and the bottom one starts with a -1 charge. This somewhat non-physical starting point will lead to far-from-equilibrium dynamics as the charge violently sloshes between the two molecules, with the oscillation period a function of the molecular separation. The trick here is to use fragments by have multiple geometries in the input deck, where each fragment is converged separately, then assembled together without SCF to use as a starting point. We use a small but diffuse basis and a range-separated functional (CAM-B3LYP). The input deck is RT_TDDFT_tcne_dimer.nw and the full output is RT_TDDFT_tcne_dimer.nwo.gz.

```
title "Tetracyanoethylene dimer charge transfer"

echo
scratch_dir ./scratch
permanent_dir ./perm

start tcne
echo

##
## Each fragment optimized with cc-pvdz/B3LYP
##
geometry "bottom" units angstroms noautosym nocenter noautoz
 C    -1.77576486    0.66496556    0.00004199
 N    -2.94676621    0.71379797    0.00004388
 C    -0.36046718    0.62491168    0.00003506
 C     0.36049301   -0.62492429   -0.00004895
 C     1.77579907   -0.66504145   -0.00006082
 N     2.94680364   -0.71382258   -0.00006592
 C    -0.31262746   -1.87038951   -0.00011201
```

```
 N    -0.85519492    -2.90926164    -0.00016331
 C     0.31276207     1.87031662     0.00010870
 N     0.85498782     2.90938919     0.00016857
end

geometry "top" units angstroms noautosym nocenter noautoz
 C    -1.77576486     0.66496556     3.00004199
 N    -2.94676621     0.71379797     3.00004388
 C    -0.36046718     0.62491168     3.00003506
 C     0.36049301    -0.62492429     2.99995105
 C     1.77579907    -0.66504145     2.99993918
 N     2.94680364    -0.71382258     2.99993408
 C    -0.31262746    -1.87038951     2.99988799
 N    -0.85519492    -2.90926164     2.99983669
 C     0.31276207     1.87031662     3.00010870
 N     0.85498782     2.90938919     3.00016857
end


## dimer geometry is the union of bottom and top geometry
geometry "dimer" units angstroms noautosym nocenter noautoz
 C    -1.77576486     0.66496556     0.00004199
 N    -2.94676621     0.71379797     0.00004388
 C    -0.36046718     0.62491168     0.00003506
 C     0.36049301    -0.62492429    -0.00004895
 C     1.77579907    -0.66504145    -0.00006082
 N     2.94680364    -0.71382258    -0.00006592
 C    -0.31262746    -1.87038951    -0.00011201
 N    -0.85519492    -2.90926164    -0.00016331
 C     0.31276207     1.87031662     0.00010870
 N     0.85498782     2.90938919     0.00016857
#---
 C    -1.77576486     0.66496556     3.00004199
 N    -2.94676621     0.71379797     3.00004388
 C    -0.36046718     0.62491168     3.00003506
 C     0.36049301    -0.62492429     2.99995105
 C     1.77579907    -0.66504145     2.99993918
 N     2.94680364    -0.71382258     2.99993408
 C    -0.31262746    -1.87038951     2.99988799
 N    -0.85519492    -2.90926164     2.99983669
 C     0.31276207     1.87031662     3.00010870
 N     0.85498782     2.90938919     3.00016857
end


##
## C, N: 3-21++G
##
basis spherical
C    S
   172.2560000             0.0617669
    25.9109000             0.3587940
     5.5333500             0.7007130
C    SP
     3.6649800            -0.3958970             0.2364600
     0.7705450             1.2158400             0.8606190
C    SP
     0.1958570             1.0000000             1.0000000
C    SP
     0.0438000             1.0000000             1.0000000
N    S
   242.7660000             0.0598657
    36.4851000             0.3529550
     7.8144900             0.7065130
N    SP
     5.4252200            -0.4133010             0.2379720
     1.1491500             1.2244200             0.8589530
N    SP
     0.2832050             1.0000000             1.0000000
N    SP
     0.0639000             1.0000000             1.0000000
end


##
## Charge density fitting basis.
##
basis "cd basis"
C    S
     5.91553927E+02        0.31582020
     1.72117940E+02        0.87503863
     5.47992590E+01        2.30760524
C    S
     1.89590940E+01        1.0000000
C    S
     7.05993000E+00        1.0000000
C    S
     2.79484900E+00        1.0000000
```

```
C    S
     1.15863400E+00         1.0000000
C    S
     4.94324000E-01         1.0000000
C    S
     2.12969000E-01         1.0000000
C    P
     3.27847358E-01         1.0000000
C    P
     7.86833659E-01         1.0000000
C    P
     1.97101832E+00         1.0000000
C    D
     4.01330100E+00         1.0000000
C    D
     1.24750500E+00         1.0000000
C    D
     4.08148000E-01         1.0000000
C    F
     9.00000000E-01         1.0000000
N    S
     7.91076935E+02         0.41567506
     2.29450184E+02         1.14750694
     7.28869600E+01         3.01935767
N    S
     2.51815960E+01         1.0000000
N    S
     9.37169700E+00         1.0000000
N    S
     3.71065500E+00         1.0000000
N    S
     1.53946300E+00         1.0000000
N    S
     6.57553000E-01         1.0000000
N    S
     2.83654000E-01         1.0000000
N    P
     4.70739194E-01         1.0000000
N    P
     1.12977407E+00         1.0000000
N    P
     2.83008403E+00         1.0000000
N    D
     5.83298650E+00         1.0000000
N    D
     1.73268650E+00         1.0000000
N    D
     5.45242500E-01         1.0000000
N    F
     1.82648000E+00         1.0000000
end


##
## Universal DFT parameters. Note, we are doing open-shell even for
## the neutral fragment so the movecs have the correct size.
##
## We are using the CAM-B3LYP functional (no need to use "direct"
## since we are doing CD fitting).
##
dft
  xc xcamb88 1.00 lyp 0.81 vwn_5 0.19 hfexch 1.00
  cam 0.33 cam_alpha 0.19 cam_beta 0.46
  odft
  convergence density 1d-9
  grid fine
  maxiter 1000
end


##
## Converge bottom fragment with extra electron and top fragment as
## neutral.
##
charge -1
set geometry "bottom"
dft
  mult 2
  vectors input atomic output "bottom.movecs"
end
task dft energy

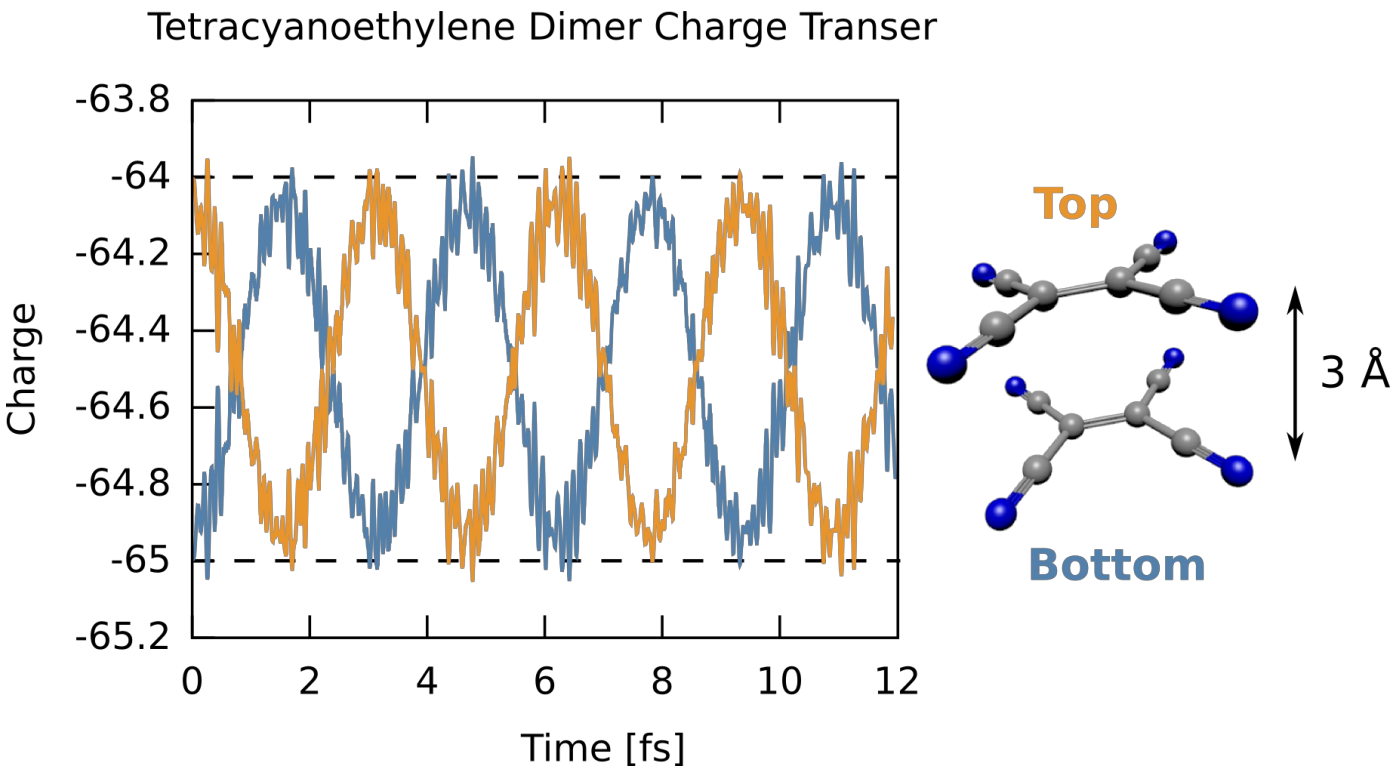
charge 0
set geometry "top"
dft
  mult 1
  vectors input atomic output "top.movecs"
end
```

```
task dft energy


##
## Assemble the two fragments but don't do SCF--this keeps the system
## in a far-from-equilibrium state from which we will watch the
## dynamics.
##
charge -1
set geometry "dimer"
dft
  mult 2
  vectors input fragment "bottom.movecs" "top.movecs" output "dimer.movecs"
  noscf
end
task dft energy


##
## Now do RT-TDDFT from this crazy state without any electric fields.
##
rt_tddft
  tmax 500.0
  dt 0.2
  load vectors "dimer.movecs"

  print dipole field energy s2 charge
end
task dft rt_tddft
```



The time-dependent charge shows that the excess electron starts on the “bottom” molecule (i.e., a total electronic charge of -65), then swings to completely occupy the “top” molecule then oscillates back and forth. The frequency of this oscillation is dependent on the separation, with larger separations leading to lower frequencies. It is important to note, however, this starting point is highly non-physical, specifically converging the two fragments together and “gluing” them together introduces an indeterminate amount of energy to the system, but this simulation shows how charge dynamics simulations can be done.

11.4.4.4 MO CAP example

```
## aug-cc-pvtz/PBE0 optimized
geometry "system" units angstroms noautosym noautoz nocenter
  O     0.00000043     0.11188833     0.00000000
  H     0.76000350    -0.47275229     0.00000000
  H    -0.76000393    -0.47275063     0.00000000
end

set geometry "system"

basis spherical
  * library 6-31G*
end

dft
  xc pbe0
  convergence density 1d-9
end
task dft

rt_tddft
  dt 0.2
  tmax 250.0

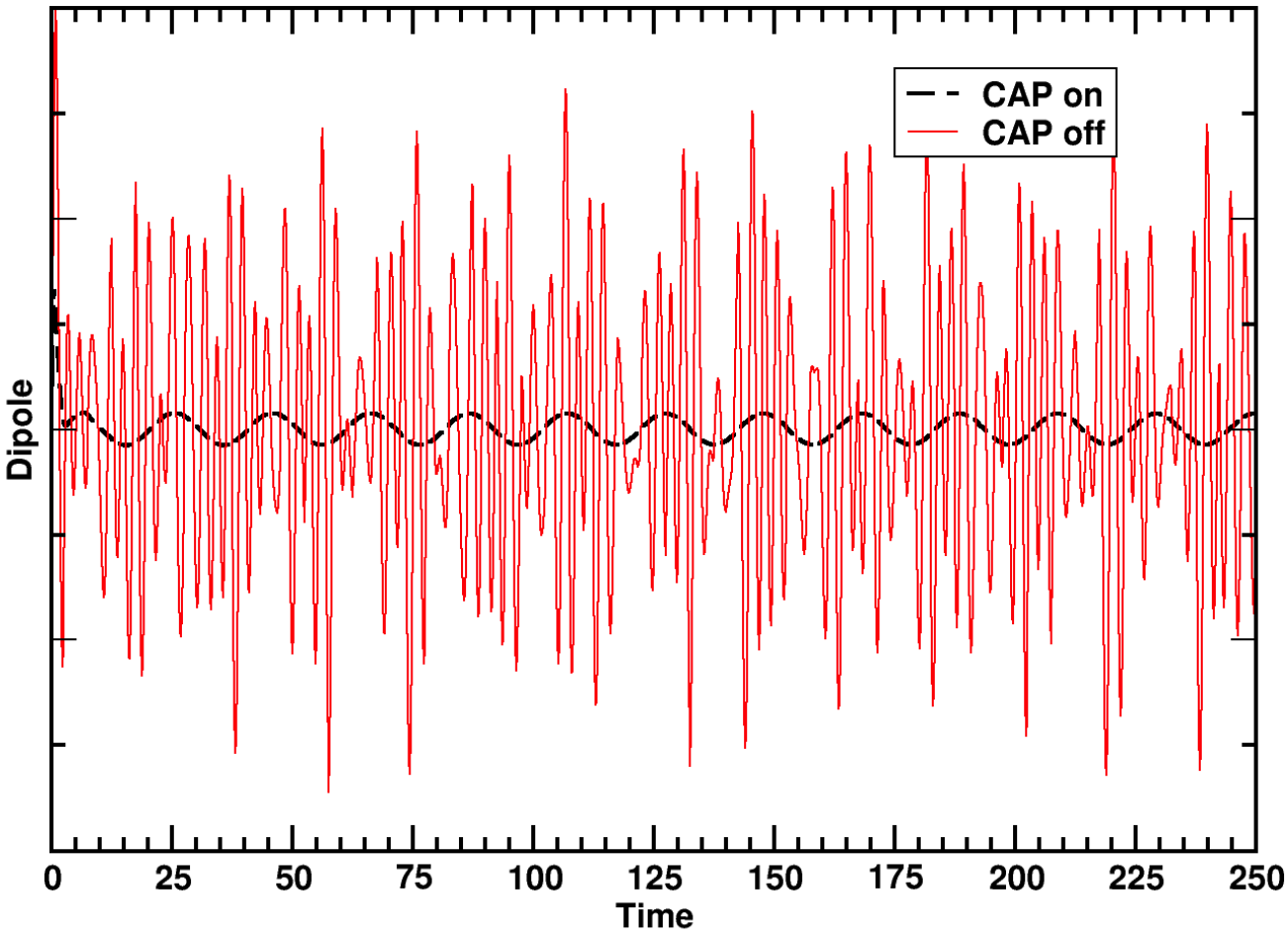
  print dipole field energy charge

  mocap
    expconst 1.0     # exponential constant for CAP
    emin 0.5         # any MO with eigenvalue >= 0.5 Ha will have CAP applied to it
    prefac 1.0       # prefactor for exponential
    maxval 100.0     # clamp CAP at this value (in Ha)
    on               # turn on CAP
    nochecks         # disable checks for speed
    noprint          # don't print CAP value
  end

  field "kick"
    type delta
    max 0.0001
    polarization z
  end

  excite "system" with "kick"
end
task dft rt_tddft
```

After running the parser script on the output files from the input above using either with the `on` or `off` keywords, the following plot can be produced from the data file obtained with the command `nw_rtparse.py -xdipole -pz -t"<rt_tddft>"`



12.1 Pseudopotential plane-wave density functional theory (NWPW)

- Pseudopotential plane-wave density functional theory (NWPW)
  - Overview
  - PSPW Tasks: Gamma Point Calculations
    - PAW Potentials
      - PAW Implementation Notes
    - Exchange-Correlation Potentials
      - DFT + U Corrections
      - Langreth style vdw and vdw van der Wall functionals
      - Grimme Dispersion Corrections
      - Using Exchange-Correlation Potentials Available in the DFT Module
      - Exact Exchange
      - Self-Interaction Corrections
    - Wannier
    - Mulliken Analysis
    - Density of States
    - Projected Density of States

- Point Charge Analysis
- PSPW_DPLOT: Generate Gaussian Cube Files

- Band Tasks: Multiple k-point Calculations
  - Brillouin Zone
    - Band Structure Paths
    - Special Points of Different Space Groups (Conventional Cells)
  - Screened Exchange
  - Density of States and Projected Density of States
  - Two-Component Wavefunctions (Spin-Orbit ZORA)
  - BAND_DPLOT: Generate Gaussian Cube Files

- Car-Parrinello
  - Adding Geometry Constraints to a Car-Parrinello Simulation
  - Car-Parrinello Output Datafiles
    - XYZ motion file
    - ION_MOTION motion file
    - EMOTION motion file
    - HMOTION motion file
    - EIGMOTION motion file
    - OMOTION motion file

- Born-Oppenheimer Molecular Dynamics
- i-PI Socket Communication
- Metropolis Monte-Carlo
- Free Energy Simulations
  - MetaDynamics
    - Input
  - TAMD - Temperature Accelerated Molecular Dynamics
    - Input
  - Collective Variables
    - Bond Distance Collective Variable
    - Angle Collective Variable
    - Coordination Collective Variable
    - N-Plane Collective Variable
    - User defined Collective Variable
- Extended X-Ray Absorption Fine Structure (EXAFS) - Integration with FEFF6L
- Frozen Phonon Calculations
- Steepest Descent
- Simulation Cell
- Unit Cell Optimization
- SMEAR - Fractional Occupation of the Molecular Orbitals
- Spin Penalty Functions

- AIMD/MM (QM/MM)
- PSP_GENERATOR
  - ATOMIC_FILLING Block
  - CUTOFF
  - SEMICORE_RADIUS
- PAW Tasks: Legacy Implementation
- Pseudopotential and PAW basis Libraries
- NWPW RTDB Entries and Miscellaneous DataFiles
  - Ion Positions
  - Ion Velocities
  - Wavefunction Datafile
  - Velocity Wavefunction Datafile
  - Formatted Pseudopotential Datafile
  - One-Dimensional Pseudopotential Datafile
- Car-Parrinello Scheme for Ab Initio Molecular Dynamics
  - Verlet Algorithm for Integration
  - Constant Temperature Simulations: Nose-Hoover Thermostats
- NWPW Tutorial 1: S2 dimer examples with PSPW
  - Total energy of S2 dimer with LDA approximation
  - Structural optimization of S2 dimer with LDA approximation
  - Frequency calculation of S2 dimer with LDA approximation
  - Ab initio molecular dynamics simulation (Car-Parrinello) of S2 dimer using the LDA approximation
  - Ab initio molecular dynamics simulation (Born-Oppenheimer) of S<sub2 dimer using the LDA approximation
- NWPW Tutorial 2: Using PSPW Car-Parrinello Simulated Annealing Simulations to Optimize Structures
  - Simulated Annealing Using Constant Energy Simulation
  - Simulated Annealing Using Constant Temperature Simulation
- NWPW Tutorial 3: using isodesmic reaction energies to estimate gas-phase thermodynamics
- NWPW Tutorial 4: AIMD/MM simulation of CCl\(_4\) + 64 H\(_2\)O
- NWPW Tutorial 5: Optimizing the Unit Cell and Geometry of Diamond
  - Optimizing the Unit Cell and Geometry for an 8 Atom Supercell of Diamond with PSPW
  - Optimizing the Unit Cell for an 8 Atom Supercell of Diamond with BAND
  - Using BAND to Optimize the Unit Cell for a 2 Atom Primitive Cell of Diamond
  - Using BAND to Calculate the Band Structures of Diamond
  - Using BAND to Calculate the Density of States of Diamond
  - Calculate the Phonon Spectrum of Diamond
- NWPW Tutorial 6: optimizing the unit cell of nickel with fractional occupation
- NWPW Tutorial 7: Optimizing the unit cells with symmetry: Diamond with Fd-3m symmetry and Brucite with P-3m1 symmetry
- NWPW Tutorial 8: NVT Metropolis Monte-Carlo Simulations
- NWPW Tutorial 9: NPT Metropolis Monte-Carlo Simulations

- NWPW Tutorial 9: Free Energy Simulations
- PAW Tutorial
  - Optimizing a water molecule
  - Optimizing a unit cell and geometry for Silicon-Carbide
  - Running a Car-Parrinello Simulation
- NWPW Capabilities and Limitations
- Questions and Difficulties

12.1.1 Overview

The NWChem plane-wave (NWPW) module uses pseudopotentials and plane-wave basis sets to perform Density Functional Theory calculations (simple introduction pw-lecture.pdf). This module complements the capabilities of the more traditional Gaussian function based approaches by having an accuracy at least as good for many applications, yet is still fast enough to treat systems containing hundreds of atoms. Another significant advantage is its ability to simulate dynamics on a ground state potential surface directly at run-time using the Car-Parrinello algorithm. This method's efficiency and accuracy make it a desirable first principles method of simulation in the study of complex molecular, liquid, and solid state systems. Applications for this first principles method include the calculation of free energies, search for global minima, explicit simulation of solvated molecules, and simulations of complex vibrational modes that cannot be described within the harmonic approximation.

The NWPW module is a collection of three modules.

- PSPW - (PSeudopotential Plane-Wave) A gamma point code for calculating molecules, liquids, crystals, and surfaces.
- Band - A band structure code for calculating crystals and surfaces with small band gaps (e.g. semi-conductors and metals).
- PAW - a (gamma point) projector augmented plane-wave code for calculating molecules, crystals, and surfaces (*This module will be deprecated in the future releases since PAW potentials have been added to PSPW*)

The PSPW, Band, and PAW modules can be used to compute the energy and optimize the geometry. Both the PSPW and Band modules can also be used to find saddle points, and compute numerical second derivatives. In addition the PSPW module can also be used to perform Car-Parrinello molecular dynamics. Section PSPW Tasks describes the tasks contained within the PSPW module, section Band Tasks describes the tasks contained within the Band module, section PAW Tasks describes the tasks contained within the PAW module, and sectionPseudopotential and PAW basis Libraries describes the pseudopotential library included with NWChem. The datafiles used by the PSPW module are described in section NWPW RTDB Entries and DataFiles. Car-Parrinello output data files are described in sectionCar-Parrinello Output Datafiles, and the minimization and Car-Parrinello algorithms are described in sectionCar-Parrinello Scheme for Ab Initio Molecular Dynamics. Examples of how to setup and run a PSPW geometry optimization, a Car-Parrinello simulation, a band structure minimization, and a PAW geometry optimization are presented at the end. Finally in section NWPW Capabilities and Limitations the capabilities and limitations of the NWPW module are discussed.

As of NWChem 6.6 to use PAW potentials the user is recommended to use the implementation contained in the PSPW module (see Sections). PAW potentials are also being integrated into the BAND module. Unfortunately, the porting to BAND was not completed for the NWChem 6.6 release.

If you are a first time user of this module it is recommended that you skip the next five sections and proceed directly to the tutorials.

12.1.2 PSPW Tasks: Gamma Point Calculations

All input to the PSPW Tasks is contained within the compound PSPW block,

```
PSPW
  ...
END
```

To perform an actual calculation a TASK PSPW directive is used (Section Task).

```
TASK PSPW
```

In addition to the directives listed inTask, i.e.

```
TASK PSPW energy
TASK PSPW gradient
TASK PSPW optimize
TASK PSPW saddle
TASK PSPW freqencies
TASK PSPW vib
```

there are additional directives that are specific to the PSPW module, which are:

```
TASK PSPW [Car-Parrinello            ||
        Born-Oppenheimer          ||
        Metropolis                ||
        pspw_et                   ||
        noit_energy               ||
        stress                    ||
        pspw_dplot                ||
        wannier                   ||
        expand_cell               ||
        exafs                     ||
        ionize                    ||
        lcao                      ||
        rdf                       ||
        aimd_properties           ||
        translate                 ||
        psp_generator             ||
        steepest_descent          ||
        psp_formatter             ||
        wavefunction_initializer  ||
        v_wavefunction_initializer ||
        wavefunction_expander      ]
```

Once a user has specified a geometry, the PSPW module can be invoked with no input directives (defaults invoked throughout). However, the user will probably always specify the simulation cell used in the computation, since the default simulation cell is not well suited for most systems. There are sub-directives which allow for customized application; those currently provided as options for the PSPW module are:

```
NWPW
 SIMULATION_CELL          ... (see section [Simulation Cell](#simulation-cell)) END
 CELL_NAME <string cell_name default 'cell_default'>
 VECTORS [[input (<string input_wavefunctions default file_prefix.movecs>) ||
       [output(<string output_wavefunctions default file_prefix.movecs>)]]
 XC (Vosko     || LDA         || PBE96        || revPBE       || PBEsol         ||
    LDA-SIC    || LDA-SIC/2   || LDA-0.4SIC   || LDA-SIC/4    || LDA-0.2SIC    ||
    PBE96-SIC  || PBE96-SIC/2  || PBE96-0.4SIC  || PBE96-SIC/4  || PBE96-0.2SIC  ||
    revPBE-SIC || revPBE-SIC/2 || revPBE-0.4SIC || revPBE-SIC/4 || revPBE-0.2SIC ||
    PBE96-Grimme2  || PBE96-Grimme3  || PBE96-Grimme4  || BLYP-Grimme2   || BLYP-Grimme3    || BLYP-Grimme4 ||
    revPBE-Grimme2 || revPBE-Grimme3 || revPBE-Grimme4 || PBEsol-Grimme2 || PBEsol-Grimme3 || PBEsol-Grimme4 ||
    PBE0-Grimme2   || PBE0-Grimme3   || PBE0-Grimme4   || B3LYP-Grimme2  || B3LYP-Grimme3  || B3LYP-Grimme4 ||
    revPBE0-Grimme2 || revPBE0-Grimme3 || revPBE0-Grimme4 ||
    PBE0      || revPBE0      || HSE  || HF || default Vosko)
 XC new ...(see section [Using Exchange-Correlation Potentials Available in the DFT Module](#Using_Exchange-Correlation_Potentials_Available_in_the_DFT_Module))
 DFT||ODFT||RESTRICTED||UNRESTRICTED
 MULT <integer mult default 1>
 CG
 LMBFGS
 SCF [Anderson|| simple || Broyden]
    [CG || RMM-DIIS]
    [density || potential]
    [ALPHA real alpha default 0.25]
    [Kerker real ekerk nodefault]
    [ITERATIONS integer inner_iterations default 5]
    [OUTER_ITERATIONS integer outer_iterations default 0]
 LOOP <integer inner_iteration outer_iteration default 10 100>
 TOLERANCES <real tole tolc default 1.0e-7 1.0e-7>
 FAKE_MASS <real fake_mass default 400000.0>
 TIME_STEP <real time_step default 5.8>
 EWALD_NCUT <integer ncut default 1>
 EWALD_RCUT <real rcut default (see input description)>
 CUTOFF  <real cutoff>
 ENERGY_CUTOFF <real ecut default (see input description)>
 WAVEFUNCTION_CUTOFF <real wcut default (see input description)>
 ALLOW_TRANSLATION
 TRANSLATION (ON || OFF)
 ROTATION (ON || OFF)
 MULLIKEN [OFF]
 EFIELD

 BO_STEPS <integer bo_inner_iteration bo_outer_iteration default 10 100>
 MC_STEPS <integer mc_inner_iteration mc_outer_iteration default 10 100>
 BO_TIME_STEP  <real bo_time_step default 5.0>
 BO_ALGORITHM [verlet|| velocity-verlet || leap-frog]
 BO_FAKE_MASS <real bo_fake_mass default 500.0>

 SOCKET (UNIX || IPI_CLIENT) <string socketname default (see input description)>

 MAPPING <integer mapping default 1>
 NP_DIMENSIONS <integer npi npj default -1 -1>
 CAR-PARRINELLO           ... (see section [Car-Parrinello](#car-parrinello-scheme-for-ab-initio-molecular-dynamics)) END
 STEEPEST_DESCENT         ... (see section [Steepest Descent](#STEEPEST_DESCENT)) END
 DPLOT                    ... (see section [DPLOT](#DPLOT)) END
 WANNIER                  ... (see section [Wannier](#Wannier)) END
 PSP_GENERATOR            ... (see section [PSP Generator](#PSP_GENERATOR))) END

 WAVEFUNCTION_INITIALIZER   ... (see section [Wavefunction Initializer](NWPW_RETIRED.md#WAVEFUNCTION_INITIALIZER) - retired) END
 V_WAVEFUNCTION_INITIATIZER ... (see section [Wavefunction Velocity Initializer (NWPW_RETIRED#V_WAVEFUNCTION_INITIALIZER) - retired) END
 WAVEFUNCTION_EXPANDER      ... (see section [Wavefunction Expander](NWPW_RETIRED.md#WAVEFUNCTION_EXPANDER) - retired) END
 INPUT_WAVEFUNCTION_FILENAME <string input_wavefunctions default file_prefix.movecs>
OUTPUT_WAVEFUNCTION_FILENAME <string output_wavefunctions default file_prefix.movecs>
END
```

The following list describes the keywords contained in the PSPW input block.

- cell_name - name of the simulation_cell named cell_name . See section Simulation Cell.
- input_wavefunctions - name of the file containing one-electron orbitals
- output_wavefunctions - name of the file that will contain the one-electron orbitals at the end of the run.
- fake_mass - value for the electronic fake mass\(\mu\) This parameter is not presently used in a conjugate gradient simulation.
- time_step - value for the time step\(\Delta t\). This parameter is not presently used in a conjugate gradient simulation.
- inner_iteration - number of iterations between the printing out of energies and tolerances
- outer_iteration - number of outer iterations
- tole - value for the energy tolerance.

- tolc - value for the one-electron orbital tolerance.
- cutoff - value for the cutoff energy used to define the wavefunction. In addition using the CUTOFF keyword automatically sets the cutoff energy for the density to be twice the wavefunction cutoff.
- ecut - value for the cutoff energy used to define the density. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- wcut - value for the cutoff energy used to define the one-electron orbitals. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- ncut - value for the number of unit cells to sum over (in each direction) for the real space part of the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.
- rcut - value for the cutoff radius used in the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.

Default set to be $\frac{MIN(\left| \vec{a_i} \right|)}{\pi}, i=1,2,3$.

- (Vosko || PBE96 || revPBE || …) - Choose between Vosko et al's LDA parameterization or the orginal and revised Perdew, Burke, and Ernzerhof GGA functional. In addition, several hybrid options.
- MULT - optional keyword which if specified allows the user to define the spin multiplicity of the system
- MULLIKEN - optional keyword which if specified causes a Mulliken analysis to be performed at the end of the simulation.
- EFIELD - optional keyword which if specified causes an atomic electric field analysis to be performed at the end of the simulation.
- ALLOW_TRANSLATION - By default the the center of mass forces are projected out of the computed forces. This optional keyword if specified allows the center of mass forces to not be zero.
- TRANSLATION - By default the the center of mass forces are projected out of the computed forces. TRANSLATION ON allows the center of mass forces to not be zero.
- ROTATION - By default the overall rotation is not projected out of the computed forces. ROTATION OFF projects out the overal rotation of the molecule.
- CG - optional keyword which sets the minimizer to 1
- LMBFGS - optional keyword which sets the minimizer to 2
- SCF - optional keyword which sets the minimizer to be a band by band minimizer. Several options are available for setting the density or potential mixing, and the type of Kohn-Sham minimizer.
- mapping - for a value of 1 slab FFT is used, for a value of 2 a 2d-hilbert FFT is used.

A variety of prototype minimizers can be used to minimize the energy. To use these other optimizers the following SET directive needs to be specified:

```
set nwpw:mimimizer 1 # Default - Grassman conjugate gradient minimizer is used to minimize the energy.
set nwpw:mimimizer 2 # Grassman LMBFGS minimimzer is used to minimize the energy.
set nwpw:minimizer 4 # Stiefel conjugate gradient minimizer is used to minimize the energy.
set nwpw:minimizer 5 # Band-by-band (potential) minimizer is used to minimize the energy.
set nwpw:minimizer 6 # Projected Grassman LMBFGS minimizer is used to minimize the energy.
set nwpw:minimizer 7 # Stiefel LMBFGS minimizer is used to minimize the energy.
set nwpw:minimizer 8 # Band-by-band (density) minimizer is used to minimize the energy.
```

Limited testing suggests that the Grassman LMBFGS minimizer is about twice as fast as the conjugate gradient minimizer. However, there are several known cases where this optimizer fails, so it is currently not a default option, and should be used with caution.

In addition the following SET directives can be specified:

```
set nwpw:lcao_skip .false. # Initial wavefunctions generated using an LCAO guess.
set nwpw:lcao_skip .true. # Default - Initial wavefunctions generated using a random plane-wave guess.
set nwpw:lcao_print .false. # Default - Output not produced during the generation of the LCAO guess.
set nwpw:lcao_print .true. # Output produced during the generation of the LCAO guess.
set nwpw:lcao_iterations 2 #specifies the number of LCAO iterations.
```

12.1.2.1 PAW Potentials

The PSPW code can now handle PAW potentials. To use them the pseudopotentials input block is used to redirect the code to use the paw potentials located in the default paw potential library ($NWCHEM_TOP/src/nwpw/libraryp/paw_default). For example, to redirect the code to use PAW potentials for carbon and hydrogen, the following input would be used.

```
nwpw
  pseudopotentials
    C library paw_default
    H library paw_default
  end
end
```

Most of the capabilities of PSPW will work with PAW potentials including geometry optimization, Car-Parrinello ab initio molecular dynamics, Born-Oppenheimer ab initio molecular dynamics, Metropolis Monte-Carlo, and AIMD/MM. Unfortunately, some of the functionality is missing at this point in time such as Mulliken analysis, and analytic stresses. However these small number of missing capabilities should become available over the next couple of months in the development tree of NWChem.

Even though analytic stresses are not currently available with PAW potentials unit cell optimization can still be carried out using numerical stresses. The following SET directives can be used to tell the code to calculate stresses numerically.

```
set includestress  .true.              #this option tells driver to optimize the unit cell
set includelattice .true.              #this option tells driver to optimize cell using a,b,c,alpha,beta,gamma
set nwpw:frozen_lattice:thresh 999.0   #large number guarentees the lattice gridding does not adjust during optimization
set nwpw:cif_filename  pspw_corundum
set nwpw:stress_numerical .true.
set nwpw:lstress_numerical .true.
```

12.1.2.1.1 PAW Implementation Notes

The main idea in the PAW method(Blochl 1994) is to project out the high-frequency components of the wavefunction in the atomic sphere region. Effectively this splits the original wavefunction into two parts:

$$\psi_n(\mathbf{r}) = \tilde{\psi}_n(\mathbf{r}) + \sum_I \psi_n^I(\mathbf{r})$$

The first part $\tilde{\psi}_n(\mathbf{r})$ is smooth and can be represented using a plane wave basis set of practical size. The second term is localized with the atomic spheres and is represented on radial grids centered on the atoms as

$$\psi_n^I(\mathbf{r}) = \sum_{\alpha} (\varphi_{\alpha}^I(\mathbf{r})-\tilde{\varphi}_{\alpha}^I(\mathbf{r}))c_{n\alpha}^I$$

where the coefficients $c_{n\alpha}^I$ are given by

$$ c_{n\alpha}^I = <\tilde{p}_{\alpha}^I | \tilde{\psi}_n>$$

This decomposition can be expressed using an invertible linear transformation,T, is defined which relates the stiff one-electron wavefunctions ψ_n to a set of smooth one-electron wavefunctions$\tilde{\psi}_n$

$$\tilde{\psi}_n = T \psi_n$$
$$\psi_n = T^{-1} \tilde{\psi}_n$$

which can be represented by fairly small plane-wave basis. The transformation$T$ is defined using a local PAW basis, which consists of atomic orbitals, $\varphi{\alpha}^I(\mathbf{r})$, smooth atomic orbitals, $\tilde{\varphi}_\alpha^I(\mathbf{r})$ which coincide with the atomic orbitals outside a defined atomic sphere, and projector functions, $p_{\alpha}^I(\mathbf{r})$. Where I is the atomic index and α is the orbital index. The projector functions are constructed such that they are localized within the defined atomic sphere and in addition are orthonormal to the atomic orbitals. Blöchl defined the invertible linear

transformations by

$$T = 1 + \sum_I \sum_\alpha (|\tilde{\varphi}_\alpha^I> - |\varphi_\alpha^I >) < p_\alpha^I|$$
$$\tilde{T} = 1 + \sum_I \sum_{\alpha} (|\varphi_{\alpha}^I> - |\tilde{\varphi}_{\alpha}^I >)<\tilde{\varphi}_{\alpha}^I|$$
$$|\tilde{p}_{\alpha}^I> = \sum_{\beta} [<\tilde{p}^I | \varphi^I >]_{\alpha\beta}^{-1} | p_{\beta}^I>$$

The main effect of the PAW transformation is that the fast variations of the valence wave function in the atomic sphere region are projected out using local basis set, thereby producing a smoothly varying wavefunction that may be expanded in a plane wave basis set of a manageable size.

The expression for the total energy in PAW method can be separated into the following 15 terms.

$$E_{PAW} = \tilde{E}_{kinetic-pw} + \tilde{E}_{vlocal-pw} + \tilde{E}_{Coulomb-pw} + \tilde{E}_{xc-pw} + E_{ion-ion}$$
$$ + E_{cmp-cmp} + E_{cmp-pw} + E_{valence-core} + E_{kinetic-core} + E_{ion-core}$$

The first five terms are essentially the same as for a standard pseudopotential plane-wave program, minus the non-local pseudopotential, where

$$\tilde{E}_{kinetic-pw} = \sum_i \sum_{\mathbf{G}} \frac{|\mathbf{G}|^2}{2} \tilde{\psi}_i^{*}(\mathbf{G}) \tilde{\psi}_i(\mathbf{G})$$
$$\tilde{E}_{vlocal-pw} = \sum_{\mathbf{G}} \tilde{\rho}(\mathbf{G}) V_{local}(\mathbf{G})$$
$$\tilde{E}_{Coulomb-pw} = \frac{\Omega}{2} \sum_{\mathbf{G}} \frac{4\pi}{|\mathbf{G}|} \tilde{\rho}^{*}(\mathbf{G}) \tilde{\rho}(\mathbf{G})$$
$$\tilde{E}_{xc-pw} = \frac{\Omega}{N_1 N_2 N_3} \sum_{\mathbf{r}} \tilde{\rho}(\mathbf{r}) \epsilon_{xc}(\tilde{\rho}(\mathbf{r}))$$
$$E_{ion-ion} = \frac{1}{2\Omega} \sum_{\mathbf{G}} \frac{4\pi}{|\mathbf{G}|^2} \exp(\frac{|\mathbf{G}|^2}{4\epsilon}) \sum_{I,J} Z_I \exp (-i \mathbf{G} \cdot \mathbf{R}_I) Z_J \exp (-i \mathbf{G} \cdot \mathbf{R}_J) +$$
$$\frac{1}{2}\sum_{\mathbf{a}} \sum_{I,J \in |\mathbf{R}_I-\mathbf{R}_J+\mathbf{a}|} Z_I Z_J \frac{erf(\epsilon |\mathbf{R}_I-\mathbf{R}_J+\mathbf{a}|)}{|\mathbf{R}_I-\mathbf{R}_J+\mathbf{a}|} - \frac{\epsilon}{\pi}\sum_I Z_I^2 +$$
$$ - \frac{\pi}{2\epsilon^2\Omega} \left(\sum_{I} Z_{I} \right)^2 $$

The local potential in the $\tilde{E}_{vlocal-pw}$ term is the Fourier transform of

$$V_{local}(\mathbf{r}) = -\sum_I Z_I \frac{\frac{|\mathbf{r}-\mathbf{R}_I|}{\sigma_I}}{|\mathbf{r}-\mathbf{R}_I|} + v_{ps}^I(|\mathbf{|\mathbf{r}-\mathbf{R}_I|})$$

It turns out that for many atoms σ_I needs to be fairly small. This results in $V_{local} (\mathbf{r})$ being stiff. However, since in the integral above this function is multiplied by a smooth density $\tilde{\rho}(\mathbf{G})$ the expansion of $V_{local}(\mathbf{G})$ only needs to be the same as the smooth density. The auxiliary pseudoptential $v_{ps}^I (|\mathbf{r}-\mathbf{R}_I |)$ is defined to be localized within the atomic sphere and is introduced to remove ghost states due to local basis set incompleteness.

The next four terms are atomic based and they essentially take into account the difference between the true valence wavefunctions and the pseudowavefunctions.

$$ E_{kinetic-atom}= \sum_I \sum_i \sum_{\alpha\beta} <\tilde{\psi}_i|\tilde{p}_{\alpha}^I < (t_{atom}^I)_{\alpha\beta} <\tilde{p}_{\beta}^I|\tilde{\psi}_i $$
$$ E_{local-atom}=\sum_I \sum_i \sum_{\alpha\beta} <\tilde{\psi}_i|\tilde{p}_{\alpha}^I> (u_{atom}^I)_{\alpha\beta} <\tilde{p}_{\beta}^I|\tilde{\psi}_i>$$
$$ E_{xc-atom}=\sum_I \sum_{\theta\phi} w_{\theta\phi} \int_0^{r_{cut}^I} r^2 (\rho^I(r,\theta,\phi) \epsilon_{xc}(\rho^I(r,\theta,\phi)) -\tilde{\rho}^I(r,\theta,\phi) \epsilon_{xc}(\tilde{\rho}^I(r,\theta,\phi))) dr $$
$$ E_{hartree-atom}= \sum_I W_{atom}^I =\frac{1}{2} \sum_I \sum_i \sum_{\alpha\beta} <\tilde{\psi}_i|\tilde{p}_{\alpha}> <\tilde{p}_{\beta}^I|\tilde{\psi}_i> $$
$$ \sum_j \sum_{\mu\nu} <\tilde{\psi}_j|\tilde{p}_{\mu}^I> <\tilde{p}_{\nu}^I|\tilde{\psi}_j> \sum_{lm} \tau_{l_\alpha m_\alpha, l_\beta m_\beta}^{lm} \tau_{l_\mu m_\mu, l_\nu m_\nu}^{lm} (V_{Heff}^I)_{\alpha\beta\mu\nu}^{l} $$

The next three terms are the terms containing the compensation charge densities.

$$E_{cmp\text{-}vloc}= \sum_{\mathbf{G}} [\rho_{cmp} (\mathbf{G}) \tilde{V}_{local}(\mathbf{G})+\tilde{\rho}_{cmp}(\mathbf{G}) (V_{local}(\mathbf{G})-\tilde{V}_{local}(\mathbf{G}))] + \\ \int (\rho_{cmp}(\mathbf{r})-\tilde{\rho}_{cmp}(\mathbf{r}))(V_{local}(r)-\tilde{V}_{local}(\mathbf{r}))d\mathbf{r}$$

$$E_{cmp\text{-}cmp}= \Omega \sum_{\mathbf{G} \neq 0} \frac{4\pi}{|\mathbf{G}|^2} [\rho_{cmp}(\mathbf{G}) \tilde{\rho}_{cmp}(\mathbf{G})- \frac{1}{2} \tilde{\rho}_{cmp}(\mathbf{G})\tilde{\rho}_{cmp}(\mathbf{G})] + \\ \frac{1}{2} \iint \frac{(\rho_{cmp}(\mathbf{r})-\tilde{\rho}_{cmp}(\mathbf{r}))(\rho_{cmp}(\mathbf{r}')-\tilde{\rho}_{cmp}(\mathbf{r}'))} {|\mathbf{r}-\mathbf{r}'|} d\mathbf{r} d\mathbf{r}'$$

$$E_{cmp\text{-}pw}= \Omega \sum_{\mathbf{G} \neq 0} \frac{4\pi}{|\mathbf{G}|^2} \rho_{cmp}(\mathbf{G}) \tilde{\rho}(\mathbf{G})$$

In the first two formulas the first terms are computed using plane-waves and the second terms are computed using Gaussian two center integrals. The smooth local potential in the $E_{cmp\text{-}vloc}$ term is the Fourier transform of

$$\tilde{V}_{local}(\mathbf{r})=- \sum_I Z_I \frac{ erf(\frac{ |\mathbf{r}-\mathbf{R}_I | }{\tilde{\sigma}_I })} {|r-R_I |}$$

The stiff and smooth compensation charge densities in the above formula are

$$\rho_{cmp}(r)=\sum_I\sum_{lm} Q_{lm}^I g_{lm}^(\sigma_I) (r-R_I)$$
$$\tilde{\rho}_{cmp}(r)=\sum_I \sum_{lm} Q_{lm}^I g_{lm}^(\tilde{\sigma}_I) (r-R_I)$$

where

$$Q_{lm}^I = \sum_i \sum_{\alpha\beta} <\tilde{\psi}_i|\tilde{p}_\alpha^I > <\tilde{p}_\beta^I|\tilde{\psi}_i > \tau_{l_\alpha m_\alpha, l_\beta m_\beta}^{lm} (q_{comp}^I)_{\alpha\beta}^I$$

The decay parameter σ_I is defined the same as above, and$\tilde{\sigma}\)_I$ is defined to be smooth enough in order that $\tilde{\rho}_{cmp}(r)$ and $\tilde{V}\)_{local}(r)$ can readily be expanded in terms of plane-waves.

The final three terms are the energies that contain the core densities

$$E_{valence\text{-}core}=\sum_{i}\sum_{I}\sum_{\alpha \beta}<\tilde{\psi}_{i} | \tilde{p}_{\alpha}^{I} > (V_{valence\text{-}core})_{\alpha \beta}^{I} <\tilde{p}_\beta^{I} | \tilde{\psi}_{i} >$$
$$E_{kinetic\text{-}core}=\sum_c \int_0^\infty \left[(\varphi_{n_c l_c}^I (r))^{'} (\varphi_{n_c l_c}^{I} (r))^{'} + l_c (l_c+1) \frac{\varphi_{n_c l_c}^{I} (r) \varphi_{n_c l_c}^{I} (r)}{r^{2}} \right] dr$$
$$E_{ion\text{-}core}= \sum_I \frac{1}{2} \iint \frac{\rho_c^{I} (r) \rho_c^{I} (r')}{|r-r'|} drdr^{'} - \int \frac{\rho_c^{I} (r)}{|r|} (Z_I+Z_I^{core}) dr$$

The matrix elements contained in the above formula are

$$(t_{atom}^{I})_{\alpha \beta}= {\delta_{m_\alpha m_\beta} \delta_{l_\alpha l_\beta }}2 \int_0^{r_{cut}^I} \left[(\varphi_{n_\alpha l_\alpha}^{I} (r))' (\varphi_{n_\beta l_\beta}^{I} (r))' - (\tilde{\varphi}_{n_\alpha l_\alpha}^{I} (r))' (\tilde{\varphi}_{n_\beta l_\beta}^{I} (r))' + l_\alpha (l_\alpha+1) \frac{ \varphi_{n_\alpha l_\alpha}^{I}(r) \varphi_{n_\beta l_\beta}^{I}(r) - \tilde{\varphi}_{n_\alpha l_\alpha}^{I}(r) \tilde{\varphi}_{n_\beta l_\beta}^I }{r^2 } \right] dr$$
$$(u_{atom}^I)_{\alpha \beta} =\frac{Z_I}{4\pi} (V_{comp}^I)_{\alpha \beta}^{l=0} + \frac{2Z_I}{(\sqrt{2\pi} \sigma_I } (q_comp^I)_{\alpha \beta}^(l=0) +\delta_{m_\alpha m_\beta } \delta_{l_\alpha l_\beta } \int_0^{r_{cut}^I} [\varphi_{n_\alpha l_\alpha}^I (r) \varphi_(n_\beta l_\beta)^I (r) \frac{-Z_I}{r} + \tilde{\varphi}_{n_\alpha l_\alpha}^I (r) \tilde{\varphi}_{n_\beta l_\beta}^I (r) (-v_{ps}^I (r))] dr$$

$$(V_{Heff}^I)_{\alpha \beta \mu \nu}^I = (V_H^I)_{\alpha\beta\mu\nu}^I - 2(V_{comp}^I)_{\alpha\beta}^I (q_{comp}^I)_{\mu\nu}^I -(v_g^I)^I (q_{comp}^I)_{\alpha\beta}^I (q_{comp}^I)_{\mu\nu}^I$$
$$(V_H^I)_{\alpha\beta\mu\nu}^I = \frac{4\pi}{2l+1} \int_0^{r_{cut}^I} \int_0^{r_{cut}^I} \left(\frac{r_{<}^l}{r_{>}^{l+1} } \right)(\varphi_{n_\alpha l_\alpha } (r) \varphi_{n_\beta l_\beta } (r) \varphi_{n_\mu l_\mu } (r) \varphi_{n_\nu l_\nu } (r) - \tilde{\varphi}_{n_\alpha l_\alpha } (r) \tilde{\varphi}_(n_\beta l_\beta) (r) \tilde{\varphi}_{n_\mu l_\mu } (r) \tilde{\varphi}_{n_\nu l_\nu } (r)) drdr'$$
$$(V_{comp}^I)_{\alpha\beta}^I = \frac{4\pi}{2l+1} \int_0^{r_{cut}^I} \int_0^{r_{cut}^I} \tilde{\varphi}_{n_\alpha l_\alpha } (r) \tilde{\varphi}_{n_\beta l_\beta } (r) \left(\frac{r_{<}^l}{r_{>}^{l+1} } \right) g_l^I (r') r'^2 drdr'$$
$$(q_{comp}^I)_{\alpha\beta}^I = \int_0^\infty r^I \left(\varphi_{n_\alpha l_\alpha } (r) \varphi_{n_\beta l_\beta } (r)- \tilde{\varphi}_{n_\alpha l_\alpha } (r) \tilde{\varphi}_{n_\beta l_\beta } (r) \right) dr$$

\[(v_g^l)^l =\frac{4\sqrt{2\pi}}{ (2l+1)(2l+1)!! \sigma_l^{2l+1} }\]
\[\tau_{l_\alpha m_\alpha,l_\beta m_\beta}^{lm}= \int_0^{2\pi} \int_0^{\pi} T_{lm} (\theta,\phi) T_{l_\alpha m_\alpha } (\theta,\phi) T_{l_\beta m_\beta } (\theta,\phi) \sin(\theta) d\theta d\phi\]

12.1.2.2 Exchange-Correlation Potentials

12.1.2.2.1 DFT + U Corrections

TO DO

```
nwpw
  uterm d 0.13634 0.0036749 1
end
```

12.1.2.2.2 Langreth style vdw and vdw van der Wall functionals

These potenials that are used to augment standard exchange-correlation potentials area calculated from a double integral over a nonlocal interaction kernel, \(\phi(\mathbf{r},\mathbf{r}^{'})\)

\[E_{vdw} = \int \rho(\mathbf{r}) \phi(\mathbf{r},\mathbf{r}^{'}) \rho(\mathbf{r}^{'}) d\mathbf{r} d\mathbf{r}^{'} \]

that is evaluated using the fast Fourier transformation method of Roman-Perez and Soler.

G. Roman-Perez and J. M. Soler, Phys. Rev. Lett. 103, 096102 (2009).

Langreth vdw and vdw2 van der Wall functionals are currently available for the BEEF, PBE96, revPBE, PBEsol, BLYP, PBE0, revPBE0, HSE, and B3LYP exchange-correlation functionals. To use them the following keywords BEEF-vdw, BEEF-vdw2, PBE96-vdw, PBE96-vdw2, BLYP-vdw, BLYP-vdw2, revPBE-vdw, revPBE-vdw, PBEsol-vdw PBEsol-vdw2, PBE0-vdw, PBE0-vdw2, revPBE0-vdw, revPBE0-vdw2, HSE-vdw, HSE-vdw2, B3LYP-vdw, and B3LYP-vdw2 can be used in the XC input directive, e.g.

```
nwpw
  xc beef-vdw
end
```

```
nwpw
  xc beef-vdw2
end
```

the vdw and vdw2 functionals are defined in

(vdw) Dion M, Rydberg H, Schröder E, Langreth DC, Lundqvist BI. Van der Waals density functional for general geometries. Physical review letters. 2004 Jun 16;92(24):246401.

(vdw2) K. Lee, E. D. Murray, L. Kong, B. I. Lundqvist, and D. C. Langreth, Phys. Rev. B 82, 081101 (2010).

12.1.2.2.3 Grimme Dispersion Corrections

Grimme dispersion corrections are currently available for the PBE96, revPBE, PBEsol, BLYP, PBE0, revPBE0, HSE, and B3LYP exchange-correlation functionals. To use them the following keywords PBE96-Grimme2, PBE96-Grimme3, PBE96-Grimme4, BLYP-Grimme2, BLYP-Grimme3, BLYP-Grimme4, revPBE-Grimme2, revPBE-Grimme3, revPBE-Grimme4, PBEsol-Grimme2, PBEsol-Grimme3, PBEsol-Grimme4, PBE0-Grimme2, PBE0-Grimme3, PBE0-Grimme4, revPBE0-Grimme2, revPBE0-Grimme3, revPBE0-Grimme4, HSE-Grimme2, HSE-Grimme3, HSE-Grimme4, B3LYP-Grimme2, B3LYP-Grimme3, and B3LYP-Grimme4 can be used in the XC input directive, e.g.

```
nwpw
  xc pbe96-grimme2
end
```

In these functionals Grimme2, Grimme3 and Grimme4 are defined in the following papers by S. Grimme.

Grimme2 - Commonly known as DFT-D2, S. Grimme, J. Comput. Chem., 27 (2006), 1787-1799.

Grimme3 - Commonly known as DFT-D3, S. Grimme, J. Antony, S. Ehrlich and H. Krieg A consistent and accurate ab initio parameterization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu, J. Chem. Phys, 132 (2010), 154104

Grimme4 - Commonly known as DFT-D3 with BJ damping. This correction augments the Grimme3 correction by including BJ-damping, S. Grimme, J. Antony, S. Ehrlich and H. Krieg A consistent and accurate ab initio parameterization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu, J. Chem. Phys, 132 (2010), 154104. S. Grimme, S. Ehrlich and L. Goerigk, J. Comput. Chem, 32 (2011), 1456-1465. This correction augments the Grimme3 correction by including BJ-damping.

If these functionals are used in a publication please include in your citations the references to Grimme's work.

12.1.2.2.4 Using Exchange-Correlation Potentials Available in the DFT Module

(*Warning - To use this capability in NWChem 6.6 the user must explicitly include the nwxc module in the NWCHEM_MODULES list when compiling. Unfortunately, there was too much uncertainty in how the nwxc computed higher-order derivatives used by some of the functionality in nwdft module to include it in a release for all the functionality in NWChem. We are planning to have a debug release in winter 2016 to take fix this problem. This capability is still included by default in NWChem 6.5*)

The user has the option of using many of the exchange-correlation potentials available in DFT Module (see Section XC and DECOMP – Exchange-Correlation Potentials).

```
XC [[acm] [b3lyp] [beckehandh] [pbe0] [bhlyp]\
   [becke97]  [becke97-1] [becke97-2] [becke97-3] [becke98] [hcth] [hcth120] [hcth147] \
   [hcth407] [becke97gga1] [hcth407p] \
   [optx] [hcthp14] [mpw91] [mpw1k] [xft97] [cft97] [ft97] [op] [bop] [pbeop]\
   [HFexch <real prefactor default 1.0>] \
   [becke88 [nonlocal] <real prefactor default 1.0>] \
   [xperdew91 [nonlocal] <real prefactor default 1.0>] \
   [xpbe96 [nonlocal] <real prefactor default 1.0>] \
   [gill96 [nonlocal] <real prefactor default 1.0>] \
   [lyp <real prefactor default 1.0>] \
   [perdew81 <real prefactor default 1.0>] \
   [perdew86 [nonlocal] <real prefactor default 1.0>] \
   [perdew91 [nonlocal] <real prefactor default 1.0>] \
   [cpbe96 [nonlocal] <real prefactor default 1.0>] \
   [pw91lda <real prefactor default 1.0>] \
   [slater <real prefactor default 1.0>] \
   [vwn_1 <real prefactor default 1.0>] \
   [vwn_2 <real prefactor default 1.0>] \
   [vwn_3 <real prefactor default 1.0>] \
   [vwn_4 <real prefactor default 1.0>] \
   [vwn_5 <real prefactor default 1.0>] \
   [vwn_1_rpa <real prefactor default 1.0>]]
```

These functional can be invoked by prepending the "new" directive before the exchange correlation potetntials in the input directive, XC new slater vwn_5.

That is, this statement in the input file

```
nwpw
 XC new slater vwn_5
end
task pspw energy
```

Using this input the user has ability to include only the local or nonlocal contributions of a given functional. The user can also specify a multiplicative prefactor (the variable `prefactor` in the input) for the local/nonlocal component or total (for more details see Section XC and DECOMP – Exchange-Correlation Potentials). An example of this might be,

```
XC new becke88 nonlocal 0.72
```

The user should be aware that the Becke88 local component is simply the Slater exchange and should be input as such.

Any combination of the supported exchange functional options can be used. For example the popular Gaussian B3

exchange could be specified as:

```
XC new slater 0.8 becke88 nonlocal 0.72 HFexch 0.2
```

Any combination of the supported correlation functional options can be used. For example B3LYP could be specified as:

```
XC new vwn_1_rpa 0.19 lyp 0.81 HFexch 0.20 slater 0.80 becke88 nonlocal 0.72
```

and X3LYP as:

```
xc new vwn_1_rpa 0.129 lyp 0.871 hfexch 0.218 slater 0.782 \
becke88 nonlocal 0.542  xperdew91 nonlocal 0.167
```

12.1.2.2.5 Exact Exchange

12.1.2.2.6 Self-Interaction Corrections

The SET directive is used to specify the molecular orbitals contribute to the self-interaction-correction (SIC) term.

```
set pspw:SIC_orbitals <integer list_of_molecular_orbital_numbers>
```

This defines only the molecular orbitals in the list as SIC active. All other molecular orbitals will not contribute to the SIC term. For example the following directive specifies that the molecular orbitals numbered 1,5,6,7,8, and 15 are SIC active.

```
set pspw:SIC_orbitals 1 5:8 15
```

or equivalently

```
set pspw:SIC_orbitals 1 5 6 7 8 15
```

The following directive turns on self-consistent SIC.

```
set pspw:SIC_relax .false. # Default - Perturbative SIC calculation
set pspw:SIC_relax .true. # Self-consistent SIC calculation
```

Two types of solvers can be used and they are specified using the following SET directive

```
set pspw:SIC_solver_type 1 # Default - cutoff coulomb kernel
set pspw:SIC_solver_type 2 # Free-space boundary condition kernel
```

The parameters for the cutoff coulomb kernel are defined by the following SET directives:

```
set pspw:SIC_screening_radius <real rcut>
set pspw:SIC_screening_power <real rpower>
```

12.1.2.3 Wannier

The pspw wannier task is generate maximally localized (Wannier) molecular orbitals. The algorithm proposed by Silvestrelli et al is use to generate the Wannier orbitals.

Input to the Wannier task is contained within the Wannier sub-block.

```
NWPW
...
 Wannier
   ...
 END
...
END
```

To run a Wannier calculation the following directive is used:

```
TASK PSPW Wannier
```

Listed below is the format of a Wannier sub-block.

```
NWPW
...
 Wannier
  OLD_WAVEFUNCTION_FILENAME <string input_wavefunctions default input_movecs>
  NEW_WAVEFUNCTION_FILENAME <string output_wavefunctions default input_movecs>
 END
...
END
```

The following list describes the input for the Wannier sub-block.

- input_wavefunctions - name of pspw wavefunction file.
- output_wavefunctions - name of pspw wavefunction file that will contain the Wannier orbitals.

12.1.2.4 Mulliken Analysis

12.1.2.5 Density of States

The “dos” option is used to turn on a density of states analysis. This option can be specified without additional parameters, i.e.

```
nwpw
  dos
end
```

in which case default values are used, or it can be specified with additional parameters, e.g.

```
nwpw
   dos 0.002 700 -0.80000 0.8000
end
```

The parameters are

```
nwpw
 dos  [<alpha> <npoints> <emin> <emax>]
end
```

where

- alpha value for the broadening the eigenvalues, default 0.05/27.2116 au
- npoints number of plotting points in dos files, default 500
- emin minimum energy in dos plots, default min(eigenvalues)-0.1 au
- emax maximimum energy in dos plots, default max(eigenvalues)+0.1 au

The units for dos parameters are in atomic units. Note that if virtual states are specified in the pspw calculation then the virtual density of states will also be generated in addition to the filled density of states.

The following files are generated and written to the permanent_dir for restricted calculations

- file_prefix.smear_dos_both - total density of states
- file_prefix.smear_fdos_both - density of states of filled states
- file_prefix.smear_vdos_both - density of states of virtual states

For unrestricted calculations

- file_prefix.smear_dos_alpha - total density of states for up electrons
- file_prefix.smear_dos_beta - total density of states for down electrons

- file_prefix.smear_fdos_alpha - density of states for filled up electrons
- file_prefix.smear_fdos_beta - density of states for filled down electrons
- file_prefix.smear_vdos_alpha - density of states for virtual up electrons
- file_prefix.smear_vdos_beta - density of states for virtual down electrons

The nwpw:dos:actlist variable is used to specify the atoms used to determine weights for dos generation. If the variable is not set then all the atoms are used, e.g.

```
set nwpw:dos:actlist 1 2 3 4
```

12.1.2.6 Projected Density of States

For projected density of states the “Mulliken” keyword needs to be set, e.g.

```
nwpw
  Mulliken
  dos
end
```

The following additional files are generated and written to the permanent_dir for restricted calculations

- file_prefix.mulliken_dos_both_s - total s projected density of restricted states
- file_prefix.mulliken_fdos_both_s - s projected density of states of filled restricted states
- file_prefix.mulliken_vdos_both_s - s projected density of states of virtual restricted states
- file_prefix.mulliken_dos_both_p - total p projected density of states
- file_prefix.mulliken_fdos_both_p - p projected density of states of filled states
- file_prefix.mulliken_vdos_both_p - p projected density of states of virtual states

…

- file_prefix.mulliken_dos_both_all - total of projected density of filled and virtual restricted states
- file_prefix.mulliken_fdos_both_all - total of projected density of filled restricted states
- file_prefix.mulliken_vdos_both_all - total of projected density of states of virtual restricted states

Similarly for unrestricted calculations

- file_prefix.mulliken_dos_alpha_s - total s projected density of up states
- file_prefix.mulliken_fdos_alpha_s - s projected density of states of filled up states
- file_prefix.mulliken_vdos_alpha_s - s projected density of states of virtual up states
- file_prefix.mulliken_dos_alpha_p - total p projected density of up states
- file_prefix.mulliken_fdos_alpha_p - p projected density of states of filled up states
- file_prefix.mulliken_vdos_alpha_p - p projected density of states of virtual up states

…

- file_prefix.mulliken_dos_alpha_all - total of projected density of filled up states
- file_prefix.mulliken_fdos_alpha_all - total of projected density of filled up states
- file_prefix.mulliken_vdos_alpha_all - total of projected density of states of virtual up states

…

- file_prefix.mulliken_dos_beta_s - total s projected density of down states

- file_prefix.mulliken_fdos_beta_s - s projected density of states of filled down states
- file_prefix.mulliken_vdos_beta_s - s projected density of states of virtual down states
- file_prefix.mulliken_dos_beta_p - total p projected density of down states
- file_prefix.mulliken_fdos_beta_p - p projected density of states of filled down states
- file_prefix.mulliken_vdos_beta_p - p projected density of states of virtual down states

…

- file_prefix.mulliken_dos_beta_all - total of projected density of filled down states
- file_prefix.mulliken_fdos_beta_all - total of projected density of filled down states
- file_prefix.mulliken_vdos_beta_all - total of projected density of states of virtual down states

12.1.2.7 Point Charge Analysis

The MULLIKEN option can be used to generate derived atomic point charges from a plane-wave density. This analysis is based on a strategy suggested in the work of P.E. Blochl, J. Chem. Phys. vol. 103, page 7422 (1995). In this strategy the low-frequency components a plane-wave density are fit to a linear combination of atom centered Gaussian functions.

The following SET directives are used to define the fitting.

```
set nwpw_APC:Gc <real Gc_cutoff> # specifies the maximum frequency component of the density to be used in the fitting in units of au.
set nwpw_APC:nga <integer number_gauss> # specifies the the number of Gaussian functions per atom.
set nwpw_APC:gamma <real gamma_list> # specifies the decay lengths of each atom centered Gaussian.
```

We suggest using the following parameters.

```
set nwpw_APC:Gc 2.5
set nwpw_APC:nga 3
set nwpw_APC:gamma 0.6 0.9 1.35
```

12.1.2.8 PSPW_DPLOT: Generate Gaussian Cube Files

The pspw dplot task is used to generate plots of various types of electron densities (or orbitals) of a molecule. The electron density is calculated on the specified set of grid points from a PSPW calculation. The output file generated is in the Gaussian Cube format. Input to the DPLOT task is contained within the DPLOT sub-block.

```
NWPW
...
 DPLOT
   ...
 END
...
END
```

To run a DPLOT calculation the following directive is used:

```
TASK PSPW PSPW_DPLOT
```

Listed below is the format of a DPLOT sub-block.

```
NWPW
...
 DPLOT
  VECTORS <string input_wavefunctions default input_movecs>
  DENSITY [total||diff||alpha||beta||laplacian||potential default total]
        <string density_name no default>
  ELF [restricted|alpha|beta] <string elf_name no default>
  ORBITAL <integer orbital_number no default> <string orbital_name no default>
  [LIMITXYZ [units <string Units default au>]
  <real X_From> <real X_To> <integer No_Of_Spacings_X>
  <real Y_From> <real Y_To> <integer No_Of_Spacings_Y>
  <real Z_From> <real Z_To> <integer No_Of_Spacings_Z>]
  NCELL <integer nx default 0> <integer ny default 0> <integer nz default 0>
  POSITION_TOLERANCE <real rtol default 0.001>
 END
...
END
```

The following list describes the input for the DPLOT sub-block.

```
VECTORS <string input_wavefunctions default input_movecs>
```

This sub-directive specifies the name of the molecular orbital file. If the second file is optionally given the density is computed as the difference between the corresponding electron densities. The vector files have to match.

```
DENSITY [total||difference||alpha||beta||laplacian||potential default total]
      <string density_name no default>
```

This sub-directive specifies, what kind of density is to be plotted. The known names for total, difference, alpha, beta, laplacian, and potential.

```
ELF [restricted|alpha|beta] <string elf_name no default>
```

This sub-directive specifies that an electron localization function (ELF) is to be plotted.

```
ORBITAL <integer orbital_number no default> <string orbital_name no default>
```

This sub-directive specifies the molecular orbital number that is to be plotted.

```
LIMITXYZ [units <string Units default angstroms>]
<real X_From> <real X_To> <integer No_Of_Spacings_X>
<real Y_From> <real Y_To> <integer No_Of_Spacings_Y>
<real Z_From> <real Z_To> <integer No_Of_Spacings_Z>
```

By default the grid spacing and the limits of the cell to be plotted are defined by the input wavefunctions. Alternatively the user can use the LIMITXYZ sub-directive to specify other limits. The grid is generated using No_Of_Spacings + 1 points along each direction. The known names for Units are angstroms, au and bohr.

12.1.3 Band Tasks: Multiple k-point Calculations

All input to the Band Tasks is contained within the compound NWPW block,

```
NWPW
 ...
END
```

To perform an actual calculation a Task Band directive is used (Section Task).

```
Task Band
```

Once a user has specified a geometry, the Band module can be invoked with no input directives (defaults invoked throughout). There are sub-directives which allow for customized application; those currently provided as options for the Band module are:

```
NWPW
 CELL_NAME <string cell_name default cell_default>
 ZONE_NAME <string zone_name default zone_default>
 INPUT_WAVEFUNCTION_FILENAME <string input_wavefunctions default input_movecs>
 OUTPUT_WAVEFUNCTION_FILENAME <string output_wavefunctions default input_movecs>
 FAKE_MASS <real fake_mass default 400000.0>
 TIME_STEP <real time_step default 5.8>
 LOOP <integer inner_iteration outer_iteration default 10 100>
 TOLERANCES <real tole tolc default 1.0e-7 1.0e-7>
 CUTOFF <real cutoff>
 ENERGY_CUTOFF <real ecut default (see input description)>
 WAVEFUNCTION_CUTOFF <real wcut default (see input description)>
 EWALD_NCUT <integer ncut default 1>]
 EWALD_RCUT <real rcut default (see input description)>

 XC (Vosko     || LDA         || PBE96       || revPBE      || PBEsol         ||  `
            || HSE  || default Vosko)                                              `
  #Note that HSE is the only hybrid functional implemented in BAND

 DFT||ODFT||RESTRICTED||UNRESTRICTED
 MULT <integer mult default 1>
 CG
 LMBFGS
 SCF [Anderson|| simple || Broyden]
 [CG || RMM-DIIS] [density || potential]
 [ALPHA real alpha default 0.25]
 [ITERATIONS integer inner_iterations default 5]
 [OUTER_ITERATIONS integer outer_iterations default 0]

 SIMULATION_CELL  [units <string units default bohrs>]
   ... (see input description)
 END
 BRILLOUIN_ZONE
   ... (see input description)
 END
 MONKHORST-PACK <real n1 n2 n3 default 1 1 1>
 BAND_DPLOT
   ... (see input description)
 END
 MAPPING <integer mapping default 1>
 SMEAR <sigma default 0.001>
 [TEMPERATURE <temperature>]
 [FERMI || GAUSSIAN || MARZARI-VANDERBILT default FERMI]
 [ORBITALS <integer orbitals default 4>]
END
```

The following list describes these keywords.

- cell_name - name of the simulation_cell named cell_name . See Simulation Cell.
- input_wavefunctions - name of the file containing one-electron orbitals
- output_wavefunctions - name that will point to file containing the one-electron orbitals at the end of the run.
- fake_mass - value for the electronic fake mass\(\mu\). This parameter is not presently used in a conjugate gradient simulation
- time_step - value for the time step\(\Delta t\). This parameter is not presently used in a conjugate gradient simulation.
- inner_iteration - number of iterations between the printing out of energies and tolerances
- outer_iteration - number of outer iterations
- tole - value for the energy tolerance.
- tolc - value for the one-electron orbital tolerance.
- cutoff - value for the cutoff energy used to define the wavefunction. In addition using the CUTOFF keyword automatically sets the cutoff energy for the density to be twice the wavefunction cutoff.
- ecut - value for the cutoff energy used to define the density. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- wcut - value for the cutoff energy used to define the one-electron orbitals. Default is set to be the maximum value that will fix within the simulation_cell cell_name .
- ncut - value for the number of unit cells to sum over (in each direction) for the real space part of the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.

- rcut - value for the cutoff radius used in the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.

Default set to be $\frac{MIN(\left| \vec{a_i} \right|)}{\pi}, i=1,2,3$.

- (Vosko || PBE96 || revPBE) - Choose between Vosko et al's LDA parameterization or the orginal and revised Perdew, Burke, and Ernzerhof GGA functional.
- SIMULATION_CELL (see section -sec:pspw_cell-)
- BRILLOUIN_ZONE (see section -sec:band_brillouin_zone-)
- MONKHORST-PACK - Alternatively, the MONKHORST-PACK keyword can be used to enter a MONKHORST-PACK sampling of the Brillouin zone.
- smear - value for smearing broadending
- temperature - same as smear but in units of K.
- CG - optional keyword which sets the minimizer to 1
- LMBFGS - optional keyword which sets the minimizer to 2
- SCF - optional keyword which sets the minimizer to be a band by band minimizer. Several options are available for setting the density or potential mixing, and the type of Kohn-Sham minimizer.

12.1.3.1 Brillouin Zone

To supply the special points of the Brillouin zone, the user defines a brillouin_zone sub-block within the NWPW block. Listed below is the format of a brillouin_zone sub-block.

```
NWPW
...
 BRILLOUIN_ZONE
   ZONE_NAME <string name default zone_default>
   (KVECTOR <real k1 k2 k3 no default> <real weight default (see input description)>
    ...)
 END
...
END
```

The user enters the special points and weights of the Brillouin zone. The following list describes the input in detail.

- name - user-supplied name for the simulation block.
- k1 k2 k3 - user-supplied values for a special point in the Brillouin zone.
- weight - user-supplied weight. Default is to set the weight so that the sum of all the weights for the entered special points adds up to unity.

12.1.3.1.1 Band Structure Paths

SC: gamma, m, r, x

FCC: gamma, k, l, u, w, x

BCC: gamma, h, n, p

Rhombohedral: not currently implemented

Hexagonal: gamma, a, h, k, l, m

Simple Tetragonal: gamma, a, m, r, x, z

Simple Orthorhombic: gamma, r, s, t, u, x, y, z

Body-Centered Tetragonal: gamma, m, n, p, x

12.1.3.1.2 Special Points of Different Space Groups (Conventional Cells)

(1) P1

(2) P-1

(3)

12.1.3.2 Screened Exchange

12.1.3.3 Density of States and Projected Density of States

The “dos” option is used to calculate density of states using broadening of the eigenvalues . This option can be specified without additional parameters, i.e.

```
nwpw
  dos
end
```

in which case default values are used, or it can be specified with additional parameters, e.g.

```
nwpw
  dos 0.002 700 -0.80000 0.8000
end
```

The parameters are

```
nwpw
 dos  [<alpha> <npoints> <emin> <emax>]
end
```

where

- alpha - value for the broadening the eigenvalues, default 0.05/27.2116 au
- npoints - number of plotting points in dos files, default 500
- emin - minimum energy in dos plots, default min(eigenvalues)-0.1 au
- emax - maximimum energy in dos plots, default max(eigenvalues)+0.1 au

The units for dos parameters are in atomic units. Note that if virtual states are specified in the pspw calculation then the virtual density of states will also be generated in addition to the filled density of states.

The following files are generated and written to the permanent_dir for restricted calculations

- file_prefix.smear_dos_both - total density of states
- file_prefix.smear_fdos_both - density of states of filled states
- file_prefix.smear_vdos_both - density of states of virtual states

For unrestricted calculations

- file_prefix.smear_dos_alpha - total density of states for up electrons
- file_prefix.smear_dos_beta - total density of states for down electrons
- file_prefix.smear_fdos_alpha - density of states for filled up electrons
- file_prefix.smear_fdos_beta - density of states for filled down electrons
- file_prefix.smear_vdos_alpha - density of states for virtual up electrons
- file_prefix.smear_vdos_beta - density of states for virtual down electrons

The nwpw:dos:actlist variable is used to specify the atoms used to determine weights for dos generation. If the variable is not set then all the atoms are used, e.g.

```
set nwpw:dos:actlist 1 2 3 4
```

For projected density of states the “Mulliken” keyword needs to be set, e.g.

```
nwpw
  Mulliken
  dos
end
```

The following additional files are generated and written to the permanent_dir for restricted calculations

- file_prefix.mulliken_dos_both_s - total s projected density of restricted states
- file_prefix.mulliken_fdos_both_s - s projected density of states of filled restricted states
- file_prefix.mulliken_vdos_both_s - s projected density of states of virtual restricted states
- file_prefix.mulliken_dos_both_p - total p projected density of states
- file_prefix.mulliken_fdos_both_p - p projected density of states of filled states
- file_prefix.mulliken_vdos_both_p - p projected density of states of virtual states

…

- file_prefix.mulliken_dos_both_all - total of projected density of filled and virtual restricted states
- file_prefix.mulliken_fdos_both_all - total of projected density of filled restricted states
- file_prefix.mulliken_vdos_both_all - total of projected density of states of virtual restricted states

Similarly for unrestricted calculations

- file_prefix.mulliken_dos_alpha_s - total s projected density of up states
- file_prefix.mulliken_fdos_alpha_s - s projected density of states of filled up states
- file_prefix.mulliken_vdos_alpha_s - s projected density of states of virtual up states
- file_prefix.mulliken_dos_alpha_p - total p projected density of up states
- file_prefix.mulliken_fdos_alpha_p - p projected density of states of filled up states
- file_prefix.mulliken_vdos_alpha_p - p projected density of states of virtual up states

…

- file_prefix.mulliken_dos_alpha_all - total of projected density of filled up states
- file_prefix.mulliken_fdos_alpha_all - total of projected density of filled up states
- file_prefix.mulliken_vdos_alpha_all - total of projected density of states of virtual up states

…

- file_prefix.mulliken_dos_beta_s - total s projected density of down states
- file_prefix.mulliken_fdos_beta_s - s projected density of states of filled down states
- file_prefix.mulliken_vdos_beta_s - s projected density of states of virtual down states
- file_prefix.mulliken_dos_beta_p - total p projected density of down states
- file_prefix.mulliken_fdos_beta_p - p projected density of states of filled down states
- file_prefix.mulliken_vdos_beta_p - p projected density of states of virtual down states

…

- file_prefix.mulliken_dos_beta_all - total of projected density of filled down states
- file_prefix.mulliken_fdos_beta_all - total of projected density of filled down states
- file_prefix.mulliken_vdos_beta_all - total of projected density of states of virtual down states

12.1.3.4 Two-Component Wavefunctions (Spin-Orbit ZORA)

12.1.3.5 BAND_DPLOT: Generate Gaussian Cube Files

The BAND BAND_DPLOT task is used to generate plots of various types of electron densities (or orbitals) of a crystal. The electron density is calculated on the specified set of grid points from a Band calculation. The output file generated is in the Gaussian Cube format. Input to the BAND_DPLOT task is contained within the BAND_DPLOT sub-block.

```
NWPW
...
 BAND_DPLOT
   ...
 END
...
END
```

To run a BAND_DPLOT calculation the following directive is used:

```
TASK BAND BAND_DPLOT
```

Listed below is the format of a BAND_DPLOT sub-block.

```
NWPW
...
 BAND_DPLOT
   VECTORS <string input_wavefunctions default input_movecs>
   DENSITY [total||difference||alpha||beta||laplacian||potential default total] <string density_name no default>
   ELF [restricted|alpha|beta] <string elf_name no default>
   ORBITAL (density || real || complex default density)
         <integer orbital_number no default>
         <integer brillion_number default 1>
         <string orbital_name no default>
   [LIMITXYZ [units <string Units default angstroms>]
    <real X_From> <real X_To> <integer No_Of_Spacings_X>
    <real Y_From> <real Y_To> <integer No_Of_Spacings_Y>
    <real Z_From> <real Z_To> <integer No_Of_Spacings_Z>]
 END
...
END
```

The following list describes the input for the BAND_DPLOT sub-block.

```
VECTORS <string input_wavefunctions default input_movecs>
```

This sub-directive specifies the name of the molecular orbital file. If the second file is optionally given the density is computed as the difference between the corresponding electron densities. The vector files have to match.

```
DENSITY [total||difference||alpha||beta||laplacian||potential default total] <string density_name no default>
```

This sub-directive specifies, what kind of density is to be plotted. The known names for total, difference, alpha, beta, laplacian, and potential.

```
ELF [restricted|alpha|beta] <string elf_name no default>
```

This sub-directive specifies that an electron localization function (ELF) is to be plotted.

```
ORBITAL (density || real || complex default density) <integer orbital_number no default><integer brillion_number default 1> <string orbital_name no default>
```

This sub-directive specifies the molecular orbital number that is to be plotted.

```
LIMITXYZ [units <string Units default angstroms>]
<real X_From> <real X_To> <integer No_Of_Spacings_X>
<real Y_From> <real Y_To> <integer No_Of_Spacings_Y>
<real Z_From> <real Z_To> <integer No_Of_Spacings_Z>
```

By default the grid spacing and the limits of the cell to be plotted are defined by the input wavefunctions. Alternatively the user can use the LIMITXYZ sub-directive to specify other limits. The grid is generated using No_Of_Spacings + 1 points along each direction. The known names for Units are angstroms, au and bohr.

12.1.4 Car-Parrinello

The Car-Parrinello task is used to perform ab initio molecular dynamics using the scheme developed by Car and Parrinello. In this unified ab initio molecular dynamics scheme the motion of the ion cores is coupled to a fictitious motion for the Kohn-Sham orbitals of density functional theory. Constant energy or constant temperature simulations can be performed. A detailed description of this method is described in section Car-Parrinello Scheme for Ab Initio Molecular Dynamics.

Input to the Car-Parrinello simulation is contained within the Car-Parrinello sub-block.

```
NWPW
...
 Car-Parrinello
   ...
 END
...
END
```

To run a Car-Parrinello calculation the following directives are used:

```
TASK PSPW Car-Parrinello
 TASK BAND Car-Parrinello
 TASK PAW Car-Parrinello
```

The Car-Parrinello sub-block contains a great deal of input, including pointers to data, as well as parameter input. Listed below is the format of a Car-Parrinello sub-block.

```
NWPW
...
 Car-Parrinello
   CELL_NAME <string cell_name default 'cell_default'>
   INPUT_WAVEFUNCTION_FILENAME <string input_wavefunctions default file_prefix.movecs>
   OUTPUT_WAVEFUNCTION_FILENAME <string output_wavefunctions default file_prefix.movecs>
   INPUT_V_WAVEFUNCTION_FILENAME <string input_v_wavefunctions default file_prefix.vmovecs>
   OUTPUT_V_WAVEFUNCTION_FILENAME <string output_v_wavefunctions default file_prefix.vmovecs>
   FAKE_MASS <real fake_mass default default 1000.0>
   TIME_STEP <real time_step default 5.0>
   LOOP <integer inner_iteration outer_iteration default 10 1>
   SCALING <real scale_c scale_r default 1.0 1.0>
   ENERGY_CUTOFF <real ecut default (see input description)>
   WAVEFUNCTION_CUTOFF <real wcut default (see input description)>
   EWALD_NCUT <integer ncut default 1>
   EWALD_RCUT <real rcut default (see input description)>
    XC (Vosko      || LDA          || PBE96        || revPBE       || HF           ||
        LDA-SIC    || LDA-SIC/2    || LDA-0.4SIC   || LDA-SIC/4    || LDA-0.2SIC   ||
        PBE96-SIC  || PBE96-SIC/2  || PBE96-0.4SIC  || PBE96-SIC/4  || PBE96-0.2SIC  ||
        revPBE-SIC || revPBE-SIC/2 || revPBE-0.4SIC || revPBE-SIC/4 || revPBE-0.2SIC ||
        PBE0       || revPBE0      || default Vosko)
   [Nose-Hoover <real Period_electron real Temperature_electron
                 real Period_ion      real Temperature_ion
                 integer Chainlength_electron integer Chainlength_ion default 100.0 298.15 100.0 298.15 1 1>]
   [TEMPERATURE <real Temperature_ion      real Period_ion
                 real Temperature_electron real Period_electron
                 integer Chainlength_ion integer Chainlength_electron default 298.15 1200 298.15 1200.0 1 1>]
   [SA_decay <real sa_scale_c sa_scale_r default 1.0 1.0>]
   XYZ_FILENAME <string xyz_filename default file_prefix.xyz>
   ION_MOTION_FILENAME <string ion_motion_filename default file_prefix.ion_motion
   EMOTION_FILENAME <string emotion_filename default file_prefix.emotion>
   HMOTION_FILENAME <string hmotion_filename nodefault>
   OMOTION_FILENAME <string omotion_filename nodefault>
   EIGMOTION_FILENAME <string eigmotion_filename nodefault>
 END
...
END
```

The following list describes the input for the Car-Parrinello sub-block.

- cell_name - name of the the simulation_cell named cell_name . See section Simulation Cell.
- input_wavefunctions - name of the file containing one-electron orbitals
- output_wavefunctions - name of the file that will contain the one-electron orbitals at the end of the run.
- input_v_wavefunctions - name of the file containing one-electron orbital velocities.
- output_v_wavefunctions - name of the file that will contain the one-electron orbital velocities at the end of the run.
- fake_mass - value for the electronic fake mass (\(\mu\)).
- time_step - value for the Verlet integration time step (\(Delta t\)).
- inner_iteration - number of iterations between the printing out of energies.
- outer_iteration - number of outer iterations
- scale_c - value for the initial velocity scaling of the one-electron orbital velocities.
- scale_r - value for the initial velocity scaling of the ion velocities.
- ecut - value for the cutoff energy used to define the density. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- wcut - value for the cutoff energy used to define the one-electron orbitals. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- ncut - value for the number of unit cells to sum over (in each direction) for the real space part of the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.
- rcut - value for the cutoff radius used in the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.

Default set to be \(\frac{MIN(\left| \vec{a_i} \right|)}{\pi}, i=1,2,3\).

- (Vosko || PBE96 || revPBE || …) - Choose between Vosko et al's LDA parameterization or the orginal and revised Perdew, Burke, and Ernzerhof GGA functional. In addition, several hybrid options.
- Nose-Hoover or Temperature - optional subblock which if specified causes the simulation to perform Nose-Hoover dynamics. If this subblock is not specified the simulation performs constant energy dynamics. See section -sec:pspw_nose- for a description of the parameters. Note that the Temperature subblock is just a reordering of the Nose-Hoover subblock.
  - Period_electron \(\equiv P_{electron}\) - estimated period for fictitious electron thermostat.
  - Temperature_electron \(\equiv T_{electron}\) - temperature for fictitious electron motion
  - Period_ion \(\equiv P_{ion}\) - estimated period for ionic thermostat
  - Temperature_ion \(\equiv T_{ion}\) - temperature for ion motion
  - Chainlength_electron - number of electron thermostat chains
  - Chainlength_ion - number of ion thermostat chains
- SA_decay - optional subblock which if specified causes the simulation to run a simulated annealing simulation. For simulated annealing to work the Nose-Hoover subblock needs to be specified. The initial temperature are taken from the Nose-Hoover subblock. See section -sec:pspw_nose- for a description of the parameters.
  - sa_scale_c \(\equiv \tau_{electron}\) - decay rate in atomic units for electronic temperature.
  - sa_scale_r \(\equiv \tau_{ionic}\) - decay rate in atomic units for the ionic temperature.
- xyz_filename - name of the XYZ motion file generated
- emotion_filename - name of the emotion motion file. See section EMOTION motion file for a description of the datafile.
- hmotion_filenameh - name of the hmotion motion file. See section HMOTION motion file for a description of the datafile.
- eigmotion_filename - name of the eigmotion motion file. See section EIGMOTION motion file for a description of the datafile.

- ion_motion_filename - name of the ion_motion motion file. See section ION_MOTION motion file for a description of the datafile.
- MULLIKEN - optional keyword which if specified causes an omotion motion file to be created.
- omotion_filename - name of the omotion motion file. See section OMOTION motion file for a description of the datafile.

When a DPLOT sub-block is specified the following SET directive can be used to output dplot data during a PSPW Car-Parrinello simulation:

```
set pspw_dplot:iteration_list <integer list_of_iteration_numbers>
```

The Gaussian cube files specified in the DPLOT sub-block are appended with the specified iteration number.

For example, the following directive specifies that at the 3,10,11,12,13,14,15, and 50 iterations Gaussian cube files are to be produced.

```
set pspw_dplot:iteration_list 3,10:15,50
```

12.1.4.1 Adding Geometry Constraints to a Car-Parrinello Simulation

The Car-Parrinello module allows users to freeze the cartesian coordinates in a simulation (Note - the Car-Parrinello code recognizes Cartesian constraints, but it does not recognize internal coordinate constraints). The +SET+ directive (Section Applying constraints in geometry optimizations) is used to freeze atoms, by specifying a directive of the form:

```
set geometry:actlist <integer list_of_center_numbers>
```

This defines only the centers in the list as active. All other centers will have zero force assigned to them, and will remain frozen at their starting coordinates during a Car-Parrinello simulation.

For example, the following directive specifies that atoms numbered 1, 5, 6, 7, 8, and 15 are active and all other atoms are frozen:

```
set geometry:actlist 1 5:8 15
```

or equivalently,

```
set geometry:actlist 1 5 6 7 8 15
```

If this option is not specified by entering a +SET+ directive, the default behavior in the code is to treat all atoms as active. To revert to this default behavior after the option to define frozen atoms has been invoked, the +UNSET+ directive must be used (since the database is persistent, see Section NWChem Architecture). The form of the +UNSET+ directive is as follows:

```
unset geometry:actlist
```

In addition, the Car-Parrinello module allows users to freeze bond lengths via a Shake algorithm. The following +SET+ directive shows how to do this.

```
set nwpw:shake_constraint "2 6 L 6.9334"
```

This input fixes the bond length between atoms 2 and 6 to be 6.9334 bohrs. Note that this input only recognizes bohrs.

When using constraints it is usually necessary to turn off center of mass shifting. This can be done by the following +SET+ directive.

```
set nwpw:com_shift .false.
```

12.1.4.2 Car-Parrinello Output Datafiles

12.1.4.2.1 XYZ motion file

Data file that stores ion positions and velocities as a function of time in XYZ format.

```
[line 1: ] n_ion
[line 2: ] do ii=1,n_ion
[line 2+ii: ] atom_name(ii), x(ii),y(ii),z(ii),vx(ii),vy(ii),vz(ii)
end do
[line n_ion+3 ] n_nion
 do ii=1,n_ion
[line n_ion+3+ii: ] atom_name(ii), x(ii),y(ii),z(ii), vx(ii),vy(ii),vz(ii)
end do
[line 2*n_ion+4: ] ....
```

12.1.4.2.2 ION_MOTION motion file

Datafile that stores ion positions and velocities as a function of time

```
[line 1: ] it_out, n_ion, omega, a1.x,a1.y,a1.z, a2.x,a2,y,a2.z, a3.x,a3.y,a3.z
[line 2: ] time
do ii=1,n_ion
[line 2+ii: ] ii, atom_symbol(ii),atom_name(ii), x(ii),y(ii),z(ii), vx(ii),vy(ii),vz(ii)
end do
[line n_ion+3 ] time
do do ii=1,n_ion
[line n_ion+3+ii: ] ii, atom_symbol(ii),atom_name(ii), x(ii),y(ii),z(ii), vx(ii),vy(ii),vz(ii)
end do
[line 2*n_ion+4: ] ....
```

12.1.4.2.3 EMOTION motion file

Datafile that store energies as a function of time.

```
[line 1: ] time, E1,E2,E3,E4,E5,E6,E7,E8,(E9,E10, if Nose-Hoover),eave,evar,have,hvar,ion_Temp
[line 2: ] ...
```

where

```
E1 = total energy
E2 = potential energy
E3 = ficticious kinetic energy
E4 = ionic kinetic energy
E5 = orbital energy
E6 = hartree energy
E7 = exchange-correlation energy
E8 = ionic energy
eave = average potential energy
evar = variance of potential energy
have = average total energy
hvar = variance of total energy
ion_Temp = temperature
```

12.1.4.2.4 HMOTION motion file

Datafile that stores the rotation matrix as a function of time.

```
[line 1: ] time
[line 2: ] ms,ne(ms),ne(ms)
do i=1,ne(ms)
[line 2+i: ] (hml(i,j), j=1,ne(ms)
end do
[line 3+ne(ms): ] time
[line 4+ne(ms): ] ....
```

12.1.4.2.5 EIGMOTION motion file

Datafile that stores the eigenvalues for the one-electron orbitals as a function of time.

```
[line 1: ] time, (eig(i), i=1,number_orbitals)
[line 2: ] ...
```

12.1.4.2.6 OMOTION motion file

Datafile that stores a reduced representation of the one-electron orbitals. To be used with a molecular orbital viewer that will be ported to NWChem in the near future.

12.1.5 Born-Oppenheimer Molecular Dynamics

```
NWPW
...
  BO_STEPS <integer bo_inner_iteration bo_outer_iteration default 10 100>
  BO_TIME_STEP  <real bo_time_step default 5.0>
  BO_ALGORITHM [verlet|| velocity-verlet || leap-frog]
  BO_FAKE_MASS <real bo_fake_mass default 500.0>
END
```

12.1.6 i-PI Socket Communication

```
NWPW
   SOCKET (UNIX || IPI_CLIENT) <string socketname default (see input description)>
END
```

The NWPW module provides native communication via the i-PI socket protocol. The behavior is identical to the i-PI socket communication provided by the DRIVER module. The NWPW implementation of the SOCKET directive is better optimized for plane-wave calculations.

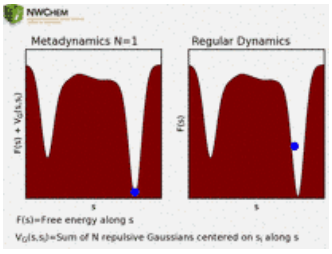
For proper behavior, the TASK directive should be set to GRADIENT , e.g. TASK PSPW GRADIENT or TASK BAND GRADIENT .

12.1.7 Metropolis Monte-Carlo

```
NWPW
...
  MC_STEPS <integer mc_inner_iteration mc_outer_iteration default 10 100>
END
```

12.1.8 Free Energy Simulations

12.1.8.1 MetaDynamics



Metadynamics234 is a powerful, non-equilibrium molecular dynamics method which accelerates the sampling of the multidimensional free energy surfaces of chemical reactions. This is achieved by adding an external time-dependent bias potential that is a function of user defined collective variables, $\mathbf{s}$. The bias potential discourages the system from sampling previously visited values of $\mathbf{s}$ (i.e., encourages the system to explore new values of $\mathbf{s}$). During the simulation the bias potential accumulates in low energy wells which then allows the system to cross energy barriers much more quickly than would occur in standard dynamics. The collective variable $\mathbf{s}$ is a generic function of the system coordinates,$\mathbf{R}$ (e.g. bond distance, bond angle, coordination numbers, etc.) that is capable of describing the chemical reaction of interest. $\mathbf{s}\left(\mathbf{R}\right)$ can be regarded as a reaction coordinate if it can distinguish between the reactant, transition, and products states, and also capture the kinetics of the reaction.

The biasing is achieved by “flooding” the energy landscape with repulsive Gaussian “hills” centered on the current location of $\mathbf{s}\left(\mathbf{R}\right)$ at a constant time intervalΔt. If the height of the Gaussians is constant in time

then we have standard metadynamics; if the heights vary (slowly decreased) over time then we have well-tempered metadynamics. In between the addition of Gaussians, the system is propagated by normal (but out of equilibrium) dynamics. Suppose that the dimension of the collective space is d, i.e. $\mathbf{s}\left(\mathbf{R}\right)=\bigl[s_1\left(\mathbf{R}\right),s_2\left(\mathbf{R}\right),\ldots,s_d\left(\mathbf{R}\right)\bigr]$ and that prior to any time t during the simulation, $N+1$ Gaussians centered on $\mathbf{S}^{t_g}$ are deposited along the trajectory of $\mathbf{s}\left(\mathbf{R}\right)$ at times $t_g = 0, \Delta t, 2\Delta t, \ldots ,N\Delta t$. Then, the value of the bias potential, V, at an arbitrary point, $\mathbf{s}\left(\mathbf{R}\right)=\bigl[s_1\left(\mathbf{R}\right),s_2\left(\mathbf{R}\right),\ldots,s_d\left(\mathbf{R}\right)\bigr]$, along the trajectory of $\mathbf{s}\left(\mathbf{R}\right)$ at time t is given by

$$V_{meta}\left(\pmb{s},t\right) = \sum_{t_g=0}^{t_g<} W(t) \exp\left(-\sum_{i=1}^{d}\frac{({s_i}-{s_i}^{t_g})^2}{2\sigma_i^2}\right)$$

where $W(t)=W_0 \exp\left(-\frac{V_{meta}\left(\mathbf{s},t-\Delta t\right)}{k_B T_{tempered}}\right)$ is the time-dependent Gaussian height. $\sigma_i \,(i=1,2,\ldots,d)$ and W_0 are width and initial height respectively of Gaussians, and $T_{tempered}$ is the tempered metadynamics temperature. $T_{tempered}=0$ corresponds to standard molecular dynamics because $W(t)=0$ and therfore there is no bias. $T_{tempered}=\infty$ corresponds to standard metadynamics since in this case $W(t)=W_0$=constant. A positive, finite value of $T_{tempered}$ (eg. $T_{tempered}$ >=1500 K) corresponds to *well-tempered* metadynamics in which $0 < W(t)<= W_0$.

For sufficiently large t, the history potential $V_{meta}\left(\mathbf{s},t\right)$ will nearly flatten the free energy surface, $F(\mathbf{s})$, along $\mathbf{S}$ and an unbiased estimator of F(s) is given by

$$F(\mathbf{s}) = -\left(1+\frac{T}{T_{tempered}}\right)\lim_{t \to \infty} V_{meta}(\mathbf{s},t)$$

12.1.8.1.1 Input

Input to a metadynamics simulation is contained within the METADYNAMICS sub-block. Listed below is the the format of a METADYNAMICS sub-block,

```
NWPW
 METADYNAMICS
   [
   BOND <integer atom1_index no default> <integer atom2_index no default>
       [W <real w default 0.00005>]
       [SIGMA <real sigma default 0.1>]
       [RANGE <real a b default (see input description)>]
       [NRANGE <integer nrange default 501>]
   ...]
   [
   ANGLE <integer atom1_index no default> <integer atom2_index no default> <integer atom3_index no default>
       [W <real w default 0.00005>]
       [SIGMA <real sigma default 0.1>]
       [RANGE <real a b default 0]
       [NRANGE <integer nrange default 501>]
   ...]
   [
   COORD_NUMBER [INDEX1 <integer_list atom1_indexes no default>][INDEX2 <integer_list atom2_indexes no default>]
       [SPRIK]
       [N <real n default 6.0>]
       [M <real m default 12.0>]
       [R0 <real r0 default 3.0>]

       [W <real w default 0.00005>]
       [SIGMA <real sigma default 0.1>]
       [RANGE <real a b no default>]
       [NRANGE <integer nrange default 501>]
   ...]
   [
   N-PLANE  <integer atom1_index no default> <integer_list atom_indexes no default>
       [W <real w default 0.00005>]
       [SIGMA <real sigma default 0.1>]
       [RANGE <real a b no default>]
       [NRANGE <integer nrange default 501>]
       [NVECTOR <real(3) nx ny nz>]
   ...]
   [UPDATE <integer meta_update default 1>]
   [PRINT_SHIFT <integer print_shift default 0>]
   [TEMPERED <real tempered_temperature no default>]
   [BOUNDARY <real w_boundary sigma_boundary no default>]
  END
 END
```

Multiple collective variables can be defined at the same time, e.g.

```
NWPW
 METADYNAMICS
   BOND 1 8  W 0.0005 SIGMA 0.1
   BOND 1 15 W 0.0005 SIGMA 0.1
  END
END
```

will produce a two-dimensional potential energy surface.

12.1.8.2 TAMD - Temperature Accelerated Molecular Dynamics

12.1.8.2.1 Input

12.1.8.3 Collective Variables

12.1.8.3.1 Bond Distance Collective Variable

This describes the bond distance between any pair of atoms$i$ and j:

$$s\left(r_{ij}\right) = \left \vert \mathbf{r}_{i}-\mathbf{r}_j\right\vert = r_{ij}$$

12.1.8.3.2 Angle Collective Variable

This describes the bond angle formed at$i$ by the triplet $<ijk>$

$$s\left(r_{ij},r_{ik}\right) = \frac{\pmb{r}_{ij}\cdot\pmb{r}_{ik}}{r_{ij}r_{ik}}$$

12.1.8.3.3 Coordination Collective Variable

The coordination number collective variable is defined as

$$s\left(r_{ij},r_{0}\right) =\sum_{i,j}\xi_{ij}$$

where the summation over i and j runs over two types of atoms,ξ_{ij} is the *weighting function*, and r_{0} is the cut-off distance. In the standard procedure for computing the coordination number, ξ_{ij} =1 if $r_{ij} < r_0$, otherwise ξ_{ij} =0, implying that ξ_{ij} is not continuous when $r_{ij}=r_{0}$. To ensure a smooth and continuous definition of the coordination number, we adopt two variants of the weighting function. The first variant is

$$\xi_{ij} = \frac{1-\left(r_{ij}/r_{0}\right)^n}{1-\left(r_{ij}/r_{0}\right)^m}$$

where n and m are integers (m > n) chosen such that$\xi_{ij}\approx 1$ when $r_{ij} < r_0$ and $\xi_{ij}\rightarrow 0$ when r_{ij} is much larger than r_{0}. For example, the parameters of the O-H coordination in water is well described by r_{0} =1.6 Å, $n=6$ and $m=18$. In practice, n and m must tuned for a given r_{0} to ensure that ξ_{ij} is smooth and satisfies the above mentioned properties, particularly the large r_{ij}

The second form of the weighting function, which is due to Sprik, is

$$\xi_{ij}=\frac{1}{1 + \exp\left[n\left(r_{ij}-r_{0}\right)\right]}$$

In this definition ξ_{ij} is analogous to the Fermi function and its width is controlled by the parameter$\frac{1}{n}$. Large and small values of n respectively correspond to sharp and soft transitions at$r_{ij} = r_{0}$. Furthermore ξ_{ij} should approach 1 and 0 when $r_{ij} < 0$ and respectively. In practice n =6-10 Å $^{-1}$. For example, a good set of parameters of the O-H coordination in water is r_{0} =1.4 Å and n =10 Å $^{-1}$.

12.1.8.3.4 N-Plane Collective Variable

The N-Plane collective variable is useful for probing the adsorption of adatom/admolecules to a surface. It is defined as the *average distance* of the adatom/admolecule from a given layer in the slab along the surface normal:

$$s = Z_{ads}-\frac{1}{N_{plane}}\sum_{i=1}^{N_{plane}}Z_i$$

where Z_{ads} denotes the position of the adatom/admolecule/impurity along the surface normal (here, we assume the surface normal to be the z-axis) and the summation over i runs over N_{plane} atoms at Z_i which form the layer. The layer could be on the face or in the interior of the slab.

12.1.8.3.5 User defined Collective Variable

12.1.9 Extended X-Ray Absorption Fine Structure (EXAFS) - Integration with FEFF6L

12.1.10 Frozen Phonon Calculations

12.1.11 Steepest Descent

The functionality of this task is now performed automatically by the PSPW and BAND. For backward compatibility, we provide a description of the input to this task.

The steepest_descent task is used to optimize the one-electron orbitals with respect to the total energy. In addition it can also be used to optimize geometries. This method is meant to be used for coarse optimization of the one-electron orbitals.

Input to the steepest_descent simulation is contained within the steepest_descent sub-block.

```
NWPW
...
 STEEPEST_DESCENT
   ...
 END
...
END
```

To run a steepest_descent calculation the following directive is used:

```
TASK PSPW steepest_descent
TASK BAND steepest_descent
```

The steepest_descent sub-block contains a great deal of input, including pointers to data, as well as parameter input. Listed below is the format of a STEEPEST_DESCENT sub-block.

```
NWPW
...
 STEEPEST_DESCENT
   CELL_NAME <string cell_name>
   [GEOMETRY_OPTIMIZE]
   INPUT_WAVEFUNCTION_FILENAME <string input_wavefunctions default file_prefix.movecs>
   OUTPUT_WAVEFUNCTION_FILENAME <string output_wavefunctions default file_prefix.movecs>
   FAKE_MASS <real fake_mass default 400000.0>
   TIME_STEP <real time_step default 5.8>
   LOOP <integer inner_iteration outer_iteration default 10 1>
   TOLERANCES <real tole tolc tolr default 1.0d-9 1.0d-9 1.0d-4>
   ENERGY_CUTOFF <real ecut default (see input desciption)>
   WAVEFUNCTION_CUTOFF <real wcut default (see input description)>
   EWALD_NCUT <integer ncut default 1>
   EWALD_RCUT <real rcut default (see input description)>
   XC (Vosko      || LDA          || PBE96        || revPBE       || HF           ||
      LDA-SIC    || LDA-SIC/2    || LDA-0.4SIC    || LDA-SIC/4    || LDA-0.2SIC    ||
      PBE96-SIC  || PBE96-SIC/2  || PBE96-0.4SIC  || PBE96-SIC/4  || PBE96-0.2SIC  ||
      revPBE-SIC || revPBE-SIC/2 || revPBE-0.4SIC || revPBE-SIC/4 || revPBE-0.2SIC ||
      PBE0       || revPBE0      || default Vosko)
   [MULLIKEN]
 END
...
END
```

The following list describes the input for the STEEPEST_DESCENT sub-block.

- cell_name - name of the simulation_cell named cell_name . See Simulation Cell.
- GEOMETRY_OPTIMIZE - optional keyword which if specified turns on geometry optimization.
- input_wavefunctions - name of the file containing one-electron orbitals
- output_wavefunctions - name of the file tha will contain the one-electron orbitals at the end of the run.
- fake_mass - value for the electronic fake mass\(\mu\)
- time_step - value for the time step\(Delta t\).
- inner_iteration - number of iterations between the printing out of energies and tolerances
- outer_iteration - number of outer iterations
- tole - value for the energy tolerance.
- tolc - value for the one-electron orbital tolerance.
- tolr - value for the ion position tolerance.
- ecut - value for the cutoff energy used to define the density. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- wcut - value for the cutoff energy used to define the one-electron orbitals. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- ncut - value for the number of unit cells to sum over (in each direction) for the real space part of the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.
- rcut - value for the cutoff radius used in the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.

Default set to be \(\frac{MIN(\left| \vec{a_i} \right|)}{\pi}, i=1,2,3\).

- (Vosko || PBE96 || revPBE || …) - Choose between Vosko et al's LDA parameterization or the orginal and revised Perdew, Burke, and Ernzerhof GGA functional. In addition, several hybrid options (hybrid options are not available in BAND).
- MULLIKEN - optional keyword which if specified causes a Mulliken analysis to be performed at the end of the simulation.

12.1.12 Simulation Cell

The simulation cell parameters are entered by defining a simulation_cell sub-block within the PSPW block. Listed below is the format of a simulation_cell sub-block.

```
NWPW
...
 SIMULATION_CELL  [units <string units default bohrs>]
   CELL_NAME <string name default 'cell_default'>
   BOUNDARY_CONDITIONS (periodic || aperiodic default periodic)
   LATTICE_VECTORS
     <real a1.x a1.y a1.z default 20.0 0.0 0.0>
     <real a2.x a2.y a2.z default 0.0 20.0 0.0>
     <real a3.x a3.y a3.z default 0.0 0.0 20.0>
   NGRID <integer na1 na2 na3 default 32 32 32>
  END
...
END
```

Basically, the user needs to enter the dimensions, gridding and boundary conditions of the simulation cell. The following list describes the input in detail.

- name - user-supplied name for the simulation block.

- periodic - keyword specifying that the simulation cell has periodic boundary conditions.
- aperiodic - keyword specifying that the simulation cell has free-space boundary conditions.
- a1.x a1.y a1.z - user-supplied values for the first lattice vector
- a2.x a2.y a2.z - user-supplied values for the second lattice vector
- a3.x a3.y a3.z - user-supplied values for the third lattice vector
- na1 na2 na3 - user-supplied values for discretization along lattice vector directions.

Alternatively, instead of explicitly entering lattice vectors, users can enter the unit cell using the standard cell parameters, a, b, c, \(alpha\), \(\beta\), and \(\gamma\), by using the LATTICE block. The format for input is as follows:

```
NWPW
...
 SIMULATION_CELL [units <string units default bohrs>]
   ...
   LATTICE
     [lat_a <real a default 20.0>]
     [lat_b <real b default 20.0>]
     [lat_c <real c default 20.0>]
     [alpha <real alpha default 90.0>]
     [beta <real beta default 90.0>]
     [gamma <real gamma default 90.0>]
   END
   ...
 END
...
END
```

The user can also enter the lattice vectors of standard unit cells using the keywords SC, FCC, BCC, for simple cubic, face-centered cubic, and body-centered cubic respectively. Listed below is an example of the format of this type of input.

```
NWPW
...
 SIMULATION_CELL [units <string units default bohrs>]
   SC 20.0
   ....
 END
...
END
```

Finally, the lattice vectors from the unit cell can also be defined using the fractional coordinate input in the GEOMETRY input (see section Geometry Lattice Parameters). Listed below is an example of the format of this type of input for an 8 atom silicon carbide unit cell.

```
geometry units au
 system crystal
   lat_a 8.277
   lat_b 8.277
   lat_c 8.277
   alpha 90.0
   beta 90.0
   gamma 90.0
 end
 Si -0.50000 -0.50000 -0.50000
 Si  0.00000  0.00000 -0.50000
 Si  0.00000 -0.50000  0.00000
 Si -0.50000  0.00000  0.00000
 C  -0.25000 -0.25000 -0.25000
 C   0.25000  0.25000 -0.25000
 C   0.25000 -0.25000  0.25000
 C  -0.25000  0.25000  0.25000
end
```

Warning - Currently only the “system crystal” option is recognized by NWPW. The “system slab” and “system polymer” options will be supported in the future.

12.1.13 Unit Cell Optimization

The PSPW module using the DRIVER geometry optimizer can optimize a crystal unit cell. Currently this type of

optimization works only if the geometry is specified in fractional coordinates. The following SET directive is used to tell the DRIVER geometry optimizer to optimize the crystal unit cell in addition to the geometry.

```
set includestress .true.
```

12.1.14 SMEAR - Fractional Occupation of the Molecular Orbitals

The smear keyword to turn on fractional occupation of the molecular orbitals in PSPW and BAND calculations

```
SMEAR <sigma default 0.001> [TEMPERATURE <temperature>]
                [FERMI || GAUSSIAN || MARZARI-VANDERBILT default FERMI]
                [ORBITALS <integer orbitals default 4>]
```

Fermi-Dirac (FERMI), Gaussian, and Marzari-Vanderbilt broadening functions are available. The ORBITALS keyword is used to change the number of virtual orbitals to be used in the calculation. Note to use this option the user must currently use the SCF minimizer. The following SCF options are recommended for running fractional occupation

```
SCF Anderson outer_iterations 0 Kerker 2.0
```

12.1.15 Spin Penalty Functions

Spin-penalty functions makes it easier to define antiferromagnetic structures. These functions are implemented by adding a scaling factor to the non-local psp for up/down spins on atoms and angular momentum that you specify.

Basically, the pseudopotential energy

$$E_{psp}= \sum_{\sigma=\uparrow,\downarrow} \sum_{i=1}^{n_{elc}^\sigma} \sum_{I=1}^{n_{ions}} \left(<\psi_i^\sigma|V_{local}^{I}|\psi_i^\sigma> + \sum_{l=0}^{l_{max}^I} \sum_{m=-l}^{l} \sum_{n=1}^{n_{max}^I} \sum_{n'=1}^{n_{max}^I} <\psi_i^\sigma|P_{nlm}^I> h_{l,n,n'}^I <P_{n'lm}^{I}|\psi_i^\sigma> \right)$$

was modified to

$$E_{psp}= \sum_{\sigma=\uparrow,\downarrow} \sum_{i=1}^{n_{elc}^\sigma} \sum_{I=1}^{n_{ions}} \left(<\psi_i^\sigma|V_{local}^{I}|\psi_i^\sigma> + \sum_{l=0}^{l_{max}^I} \sum_{m=-l}^{l} \sum_{n=1}^{n_{max}^I} \sum_{n'=1}^{n_{max}^I} \left(1-\delta_{l,l^\sigma} \delta_{I,ionlist^\sigma}(\xi^\sigma-1)\right) <\psi_i^\sigma|P_{nlm}^I> h_{l,n,n'}^I <P_{n'lm}^{I}|\psi_i^\sigma> \right)$$

An example input is as follows:

```
title "hematite 10 atoms"

start hema10

memory 1900 mb

permanent_dir ./perm
scratch_dir   ./perm

geometry center noautosym noautoz print
 system crystal
   lat_a 5.42
   lat_b 5.42
   lat_c 5.42
   alpha 55.36
   beta  55.36
   gamma 55.36
 end
Fe                0.355000   0.355000   0.355000
Fe                0.145000   0.145000   0.145000
Fe               -0.355000  -0.355000  -0.355000
Fe                0.855000   0.855000   0.855000
O                 0.550000  -0.050000   0.250000
O                 0.250000   0.550000  -0.050000
O                -0.050000   0.250000   0.550000
O                -0.550000   0.050000  -0.250000
O                -0.250000  -0.550000   0.050000
O                 0.050000  -0.250000  -0.550000\
end

nwpw
  virtual 8
  odft
  ewald_rcut 3.0
  ewald_ncut 8
  xc pbe96
  lmbfgs
  mult 1
  dplot
     density diff diff1.cube
  end

  #spin penalty functions
  pspspin up   d -1.0 1:2
  pspspin down d -1.0 3:4
end
task pspw energy
task pspw pspw_dplot

nwpw
  pspspin off
  dplot
     density diff diff2.cube
  end
end
task pspw energy
task pspw pspw_dplot
```

12.1.16 AIMD/MM (QM/MM)

A QM/MM capability is available that is integrated with PSPW module and can be used with Car-Parrinello simulations. Currently, the input is not very robust but it is straightforward. The first step to run a QM/MM simulations is to define the MM atoms in the geometry block. The MM atoms must be at the end of the geometry and a carat, ” ^ “, must be appended to the end of the atom name, e.g.

```
geometry units angstrom nocenter noautosym noautoz print xyz
 C -0.000283 0.000106 0.000047
 Cl -0.868403 1.549888 0.254229
 Cl 0.834043 -0.474413 1.517103
 Cl -1.175480 -1.275747 -0.460606
 Cl 1.209940 0.200235 -1.310743
 O^ 0.3226E+01 -0.4419E+01 -0.5952E+01
 H^ 0.3193E+01 -0.4836E+01 -0.5043E+01
 H^ 0.4167E+01 -0.4428E+01 -0.6289E+01
 O^ 0.5318E+01 -0.3334E+01 -0.1220E+01
 H^ 0.4978E+01 -0.3040E+01 -0.2113E+01
 H^ 0.5654E+01 -0.2540E+01 -0.7127E+00
end
```

Another way to specify the MM atoms is to use the mm_tags option which appends the atoms with a " ^ ".

```
geometry units angstrom nocenter noautosym noautoz print xyz
 C -0.000283 0.000106 0.000047
 Cl -0.868403 1.549888 0.254229
 Cl 0.834043 -0.474413 1.517103
 Cl -1.175480 -1.275747 -0.460606
 Cl 1.209940 0.200235 -1.310743
 O 0.3226E+01 -0.4419E+01 -0.5952E+01
 H 0.3193E+01 -0.4836E+01 -0.5043E+01
 H 0.4167E+01 -0.4428E+01 -0.6289E+01
 O 0.5318E+01 -0.3334E+01 -0.1220E+01
 H 0.4978E+01 -0.3040E+01 -0.2113E+01
 H 0.5654E+01 -0.2540E+01 -0.7127E+00
end
NWPW
 QMMM
   mm_tags 6:11
 END
END
```

The option "mm_tags off" can be used to remove the " ^ " from the atoms, i.e.

```
NWPW
 QMMM
   mm_tags 6:11 off
 END
END
```

Next the pseudopotentials have be defined for the every type of MM atom contained in the geometry blocks. The following local pseudopotential suggested by Laio, VandeVondele and Rothlisberger can be automatically generated.

$$\begin{matrix}V(\vec{r}) = -Z_{ion}\frac{{r_c}^{n_{\sigma}} - r^{n_{\sigma}}}{-sign(Z_{ion})*{r_c}^{n_{\sigma}+1}-r^{n_{\sigma}+1}}\end{matrix}$$

The following input To define this pseudopo the O^ MM atom using the following input

```
NWPW
 QMMM
   mm_psp O^ -0.8476 4 0.70
 END
END
```

defines the local pseudopotential for the O^ MM atom , where $Z_{ion}=-0.8476$, $n_{\sigma}=4$, and $r_c=0.7$. The following input can be used to define the local pseudopotentials for all the MM atoms in the geometry block defined above

```
NWPW
 QMMM
   mm_psp O^ -0.8476 4 0.70
   mm_psp H^ 0.4238 4 0.40
 END
END
```

Next the Lenard-Jones potentials for the QM and MM atoms need to be defined. This is done as as follows

```
NWPW
 QMMM
   lj_ion_parameters C 3.41000000d0 0.10d0
   lj_ion_parameters Cl 3.45000000d0 0.16d0
   lj_ion_parameters O^ 3.16555789d0 0.15539425d0
 END
END
```

Note that the Lenard-Jones potential is not defined for the MM H atoms in this example. The final step is to define the MM fragments in the simulation. MM fragments are a set of atoms in which bonds and angle harmonic potentials are defined, or alternatively shake constraints are defined. The following input defines the fragments for the two water molecules in the above geometry,

```
NWPW
 QMMM
  fragment spc
    size 3                #size of fragment
    index_start 6:9:3     #atom index list that defines the start of
                          # the fragments (start:final:stride)
    bond_spring 1 2 0.467307856 1.889726878   #bond i j Kspring r0
    bond_spring 1 3 0.467307856 1.889726878   #bond i j Kspring r0
    angle_spring 2 1 3 0.07293966 1.910611932 #angle i j k Kspring theta0
  end
 END
END
```

The fragments can be defined using shake constraints as

```
NWPW
 QMMM
  fragment spc
    size 3                #size of fragment
    index_start 6:9:3     #atom index list that defines the start of
                          # the fragments (start:final:stride)
    shake units angstroms 1 2 3 cyclic 1.0 1.632993125 1.0
  end
 END
END
```

Alternatively, each water could be defined independently as follows.

```
NWPW
 QMMM
  fragment spc1
    size 3                #size of fragment
    index_start 6         #atom index list that defines the start of
                          #the fragments
    bond_spring 1 2 0.467307856 1.889726878 #bond i j Kspring r0
    bond_spring 1 3 0.467307856 1.889726878 #bond i j Kspring r0
    angle_spring 2 1 3 0.07293966 1.910611932 #angle i j k Kspring theta0
  end
  fragment spc2
    size 3                #size of fragment
    index_start 9         #atom index list that defines the start of
                          #the fragments
    shake units angstroms 1 2 3 cyclic 1.0 1.632993125 1.0
  end
 END
END
```

12.1.17 PSP_GENERATOR

A one-dimensional pseudopotential code has been integrated into NWChem. This code allows the user to modify and develop pseudopotentials. Currently, only the Hamann and Troullier-Martins norm-conserving pseudopotentials can be generated. In future releases, the pseudopotential library (section Pseudopotential and PAW basis Libraries) will be more complete, so that the user will not have explicitly generate pseudopotentials using this module.

Input to the PSP_GENERATOR task is contained within the PSP_GENERATOR sub-block.

```
NWPW
...
 PSP_GENERATOR
  ...
 END
...
END
```

To run a PSP_GENERATOR calculation the following directive is used:

```
TASK PSPW PSP_GENERATOR
```

Listed below is the format of a PSP_GENERATOR sub-block.

```
NWPW
...
 PSP_GENERATOR
  PSEUDOPOTENTIAL_FILENAME: <string psp_name>
  ELEMENT: <string element>
  CHARGE: <real charge>
  MASS_NUMBER: <real mass_number>
  ATOMIC_FILLING: <integer ncore nvalence> ( (1||2||...) (s||p||d||f||...) <real filling>  ...)

  [CUTOFF: <integer lmax> ( (s||p||d||f||g) <real rcut> ...) ]

  PSEUDOPOTENTIAL_TYPE: (TROULLIER-MARTINS || HAMANN default HAMANN)
  SOLVER_TYPE: (PAULI || SCHRODINGER default PAULI)
  EXCHANGE_TYPE: (dirac || PBE96 default DIRAC)
  CORRELATION_TYPE: (VOSKO || PBE96 default VOSKO)
  [SEMICORE_RADIUS: <real rcore>]

 END
...
END
```

The following list describes the input for the PSP_GENERATOR sub-block.

- psp_name - name that points to a.
- element - Atomic symbol.
- charge - charge of the atom
- mass - mass number for the atom
- ncore - number of core states
- nvalence - number of valence states.
- ATOMIC_FILLING:.....(see below)
- filling - occupation of atomic state
- CUTOFF:....(see below)
- rcore - value for the semicore radius (see below)

12.1.17.1 ATOMIC_FILLING Block

This required block is used to define the reference atom which is used to define the pseudopotential. After the ATOMIC_FILLING: line, the core states are listed (one per line), and then the valence states are listed (one per line). Each state contains two integer and a value. The first integer specifies the radial quantum number, \(n\), the second integer specifies the angular momentum quantum number, \(l\), and the third value specifies the occupation of the state.

For example to define a pseudopotential for the Neon atom in the\(1s^2 2s^2 2p^6\) state could have the block

```
ATOMIC_FILLING: 1 2
 1 s 2.0 #core state - 1s^2
 2 s 2.0 #valence state - 2s^2
 2 p 6.0 #valence state - 2p^6
```

for a pseudopotential with a \(2s\) and \(2p\) valence electrons or the block

```
ATOMIC_FILLING: 3 0
 1 s 2.0 #core state
 2 s 2.0 #core state
 2 p 6.0 #core state
```

could be used for a pseudopotential with no valence electrons.

12.1.17.2 CUTOFF

This optional block specifies the cutoff distances used to match the all-electron atom to the pseudopotential atom. For Hamann pseudopotentials \(r_{cut}(l)\) defines the distance where the all-electron potential is matched to the

pseudopotential, and for Troullier-Martins pseudopotentials $r_{cut}(l)$ defines the distance where the all-electron orbital is matched to the pseudowavefunctions. Thus the definition of the radii depends on the type of pseudopotential. The cutoff radii used in Hamann pseudopotentials will be smaller than the cutoff radii used in Troullier-Martins pseudopotentials.

For example to define a softened Hamann pseudopotential for Carbon would be

```
ATOMIC_FILLING: 1 2
 1 s 2.0
 2 s 2.0
 2 p 2.0
CUTOFF: 2
 s 0.8
 p 0.85
 d 0.85
```

while a similarly softened Troullier-Marting pseudopotential for Carbon would be

```
ATOMIC_FILLING: 1 2
 1 s 2.0
 2 s 2.0
 2 p 2.0
CUTOFF: 2
 s 1.200
 p 1.275
 d 1.275
```

12.1.17.3 SEMICORE_RADIUS

Specifying the SEMICORE_RADIUS option turns on the semicore correction approximation proposed by Louie et al (S.G. Louie, S. Froyen, and M.L. Cohen, Phys. Rev. B, **26**(, 1738, (1982)). This approximation is known to dramatically improve results for systems containing alkali and transition metal atoms.

The implementation in the PSPW module defines the semi-core density,$\rho_{semicore}$, by using the sixth-order polynomial

$$\rho_{semicore}(r) = \begin{cases} \rho_{core} \mbox{if } r \ge r_{semicore} \\ c_0 + c_3 r^3 + c_4 r^4 + c_5 r^5 + c_6 r^6 \mbox{if } r < r_{semicore} \end{cases}$$

This expansion was suggested by Fuchs and Scheffler (M. Fuchs, and M. Scheffler, Comp. Phys. Comm.,**119**,67 (1999)), and is better behaved for taking derivatives (i.e. calculating ionic forces) than the expansion suggested by Louie et al.

12.1.18 PAW Tasks: Legacy Implementation

(*This capability is now available in PSPW. It is recommended that this module only be used for testing purposes.*)

All input to the PAW Tasks is contained within the compound NWPW block,

```
NWPW
 ...
END
```

To perform an actual calculation a TASK PAW directive is used (Task).

```
TASK PAW
```

In addition to the directives listed inTask, i.e.

```
TASK paw energy
TASK paw gradient
TASK paw optimize
TASK paw saddle
TASK paw freqencies
TASK paw vib
```

there are additional directives that are specific to the PSPW module, which are:

```
TASK PAW [Car-Parrinello || steepest_descent ]
```

Once a user has specified a geometry, the PAW module can be invoked with no input directives (defaults invoked throughout). There are sub-directives which allow for customized application; those currently provided as options for the PAW module are:

```
NWPW
 CELL_NAME <string cell_name default 'cell_default'>
 [GEOMETRY_OPTIMIZE]
 INPUT_WAVEFUNCTION_FILENAME <string input_wavefunctions default input_movecs>
 OUTPUT_WAVEFUNCTION_FILENAME <string output_wavefunctions default input_movecs>
 FAKE_MASS <real fake_mass default 400000.0>
 TIME_STEP <real time_step default 5.8>
 LOOP <integer inner_iteration outer_iteration default 10 100>
 TOLERANCES <real tole tolc default 1.0e-7 1.0e-7>
 CUTOFF <real cutoff>
 ENERGY_CUTOFF <real ecut default (see input description)>
 WAVEFUNCTION_CUTOFF <real wcut default (see input description)>
 EWALD_NCUT <integer ncut default 1>]
 EWALD_RCUT <real rcut default (see input description)>
 XC (Vosko || PBE96 || revPBE default Vosko)
 DFT||ODFT||RESTRICTED||UNRESTRICTED
 MULT <integer mult default 1>
 INTEGRATE_MULT_L <integer imult default 1>
 SIMULATION_CELL
   ... (see input description)
 END
 CAR-PARRINELLO
   ... (see input description)
 END
 MAPPING <integer mapping default 1>
END
```

The following list describes these keywords.

- cell_name - name of the the simulation_cell named cell_name . The current version of PAW only accepts periodic unit cells. See Simulation Cell.
- GEOMETRY_OPTIMIZE - optional keyword which if specified turns on geometry optimization.
- input_wavefunctions - name of the file containing one-electron orbitals
- output_wavefunctions - name of the file that will contain the one-electron orbitals at the end of the run.
- fake_mass - value for the electronic fake mass\(\mu\). This parameter is not presently used in a conjugate gradient simulation
- time_step - value for the time step (\(\Delta t\)). This parameter is not presently used in a conjugate gradient simulation.
- inner_iteration - number of iterations between the printing out of energies and tolerances
- outer_iteration - number of outer iterations
- tole - value for the energy tolerance.
- tolc - value for the one-electron orbital tolerance.
- cutoff - value for the cutoff energy used to define the wavefunction. In addition using the CUTOFF keyword automatically sets the cutoff energy for the density to be twice the wavefunction cutoff.
- ecut - value for the cutoff energy used to define the density. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- wcut - value for the cutoff energy used to define the one-electron orbitals. Default is set to be the maximum value that will fix within the simulation_cell cell_name .
- ncuth - value for the number of unit cells to sum over (in each direction) for the real space part of the smooth compensation summation.
- rcut - value for the cutoff radius used in the smooth compensation summation.

Default set to be \(\frac{MIN(\left| \vec{a_i} \right|)}{\pi}, i=1,2,3\).

- (Vosko || PBE96 || revPBE) - Choose between Vosko et al's LDA parametrization or the original and revised Perdew, Burke, and Ernzerhof GGA functional.
- MULT - optional keyword which if specified allows the user to define the spin multiplicity of the system
- INTEGRATE_MULT_L - optional keyword which if specified allows the user to define the angular XC integration of the augmented region
- SIMULATION_CELL (see Simulation Cell)
- CAR-PARRINELLO (see Car-Parrinello)
- mapping - for a value of 1 slab FFT is used, for a value of 2 a 2d-Hilbert FFT is used.

12.1.19 Pseudopotential and PAW basis Libraries

A library of pseudopotentials used by PSPW and BAND is currently available in the directory

$NWCHEM_TOP/src/nwpw/libraryp/pspw_default

The elements listed in the following table are present:

```
 H                                  He
-------                          ----------
 Li Be                            B  C  N  O  F  Ne
-------                          ----------
 Na Mg                             Al Si P  S  Cl Ar
---------------------------------------------------------
 K  Ca Sc Ti V  Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
---------------------------------------------------------
 Rb Sr Y  Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I  Xe
---------------------------------------------------------
 Cs Ba La Hf Ta W  Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
---------------------------------------------------------
 Fr Ra .
----------
      -----------------------------------------
      . . . . . Gd . . . . . .
      -----------------------------------------
      . . U . Pu . . . . . .
      -----------------------------------------
```

The pseudopotential libraries are continually being tested and added. Also, the PSPW program can read in pseudopotentials in CPI and TETER format generated with pseudopotential generation programs such as the OPIUM package of Rappe et al. The user can request additional pseudopotentials from Eric J. Bylaska at (Eric.Bylaska@pnl.gov).

Similarly, a library of PAW basis used by PAW is currently available in the directory $NWCHEM_TOP/src/nwpw/libraryp/paw_default

```
 H                                  He
-------                          ----------
 Li Be                            B  C  N  O  F  Ne
-------                          ----------
 Na Mg                             Al Si P  S  Cl Ar
---------------------------------------------------------
 K  Ca Sc Ti V  Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
---------------------------------------------------------
 . . . . . .
---------------------------------------------------------
 . . . . . .
---------------------------------------------------------
 . . .
----------
      -----------------------------------------
      . . . . . .
      -----------------------------------------
      . . . . . .
      -----------------------------------------
```

Currently there are not very many elements available for PAW. However, the user can request additional basis sets from Eric J. Bylaska at (Eric.Bylaska@pnl.gov).

A preliminary implementation of the HGH pseudopotentials (Hartwigsen, Goedecker, and Hutter) has been implemented into the PSPW module. To access the pseudopotentials the pseudopotentials input block is used. For example, to redirect

the code to use HGH pseudopotentials for carbon and hydrogen, the following input would be used.

```
nwpw
...
 pseudopotentials
  C library HGH_LDA
  H library HGH_LDA
 end
...
end
```

The implementation of HGH pseudopotentials is rather limited in this release. HGH pseudopotentials cannot be used to optimize unit cells, and they do not work with the MULLIKEN option. They also have not yet been implemented into the BAND structure code.
To read in pseudopotentials in CPI format the following input would be used.

```
nwpw
...
 pseudopotentials
  C CPI c.cpi
  H CPI h.cpi
 end
...
end
```

In order for the program to recognize the CPI format the CPI files, e.g. c.cpi have to be prepended with the""keyword.

To read in pseudopotentials in TETER format the following input would be used.

```
nwpw
...
 pseudopotentials
  C TETER c.teter
  H TETER h.teter
 end
...
end
```

In order for the program to recognize the TETER format the TETER files, e.g. c.teter have to be prepended with the" " keyword.

If you wish to redirect the code to a different directory other than the default one, you need to set the environmental variable NWCHEM_NWPW_LIBRARY to the new location of the libraryps directory.

12.1.20 NWPW RTDB Entries and Miscellaneous DataFiles

Input to the PSPW and Band modules are contained in both the RTDB and datafiles. The RTDB is used to store input that the user will need to directly specify. Input of this kind includes ion positions, ion velocities, and simulation cell parameters. The datafiles are used to store input, such the one-electron orbitals, one-electron orbital velocities, formatted pseudopotentials, and one-dimensional pseudopotentials, that the user will in most cases run a program to generate.

12.1.20.1 Ion Positions

The positions of the ions are stored in the default geometry structure in the RTDB and must be specified using the GEOMETRY directive.

12.1.20.2 Ion Velocities

The velocities of the ions are stored in the default geometry structure in the RTDB, and must be specified using the GEOMETRY directive.

12.1.20.3 Wavefunction Datafile

The one-electron orbitals are stored in a wavefunction datafile. This is a binary file and cannot be directly edited. This datafile is used by steepest_descent and Car-Parrinello tasks and can be generated using the wavefunction_initializer or wavefunction_expander tasks.

12.1.20.4 Velocity Wavefunction Datafile

The one-electron orbital velocities are stored in a velocity wavefunction datafile. This is a binary file and cannot be directly edited. This datafile is automatically generated the first time a Car-Parrinello task is run.

12.1.20.5 Formatted Pseudopotential Datafile

The pseudopotentials in Kleinman-Bylander form expanded on a simulation cell (3d grid) are stored in a formatted pseudopotential datafile (PSPW-“atomname.vpp”, BAND-“atomname.cpp”, PAW-“atomname.jpp”). These are binary files and cannot be directly edited. These datafiles are automatically generated.

12.1.20.6 One-Dimensional Pseudopotential Datafile

The one-dimensional pseudopotentials are stored in a one-dimensional pseudopotential file (“atomname.psp”). This is an ASCII file and can be directly edited with a text editor or can be generated using the pspw_generator task. However, these datafiles are usually atomatically generated.

The data stored in the one-dimensional pseudopotential file is

```
character*2 element :: element name
integer charge :: valence charge of ion
real mass :: mass of ion
integer lmax :: maximum angular component
real rcut(lmax) :: cutoff radii used to define pseudopotentials
integer nr :: number of points in the radial grid
real dr :: linear spacing of the radial grid
real r(nr):: one-dimensional radial grid
real Vpsp(nr,lmax) :: one-dimensional pseudopotentials
real psi(nr,lmax) :: one-dimensional pseudowavefunctions
real r_semicore :: semicore radius
real rho_semicore(nr) :: semicore density
```

and the format of it is:

```
[line 1: ] element [line 2: ] charge mass lmax
[line 3: ] (rcut(l), l=1,lmax)
[line 4: ] nr dr
[line 5: ] r(1) (Vpsp(1,l), l=1,lmax)
[line 6: ] ....
[line nr+4: ] r(nr) (Vpsp(nr,l), l=1,lmax)
[line nr+5: ] r(1) (psi(1,l), l=1,lmax) [line nr+6: ] ....
[line 2*nr+4:] r(nr) (psi(nr,l), l=1,lmax)
[line 2*nr+5:] r_semicore
if (r_semicore read) then
 [line 2*nr+6:] r(1) rho_semicore(1)
 [line 2*nr+7:] ....
 [line 3*nr+5:] r(nr) rho_semicore(nr)
end if
```

12.1.21 Car-Parrinello Scheme for Ab Initio Molecular Dynamics

Car and Parrinello developed a unified scheme for doing *ab initio* molecular dynamics by combining the motion of the ion cores and a fictitious motion for the Kohn-Sham orbitals of density-functional theory (R. Car and M. Parrinello, Phys. Rev. Lett. **55**, 2471, (1985) - simple introductioncpmd-lecture.pdf). At the heart of this method they introduced a fictitious kinetic energy functional for the Kohn-Sham orbitals.

$$\begin{align}KE \left (\bigl[\psi_{i,\sigma}(\vec{r}) \bigr] \right) = \sum_{i,\sigma}^{occ} \int d\vec{r}\ \mu \left | \dot{\psi}_{i,\sigma}(\vec{r}) \right | ^2\end{align}$$

Given this kinetic energy the constrained equations of motion are found by taking the first variation of the auxiliary Lagrangian.

$$ L = \sum_{i,\sigma}^{occ} \int d\vec{r}\ \mu \left | \dot{\psi}_{i,\sigma}(\vec{r}) \right | ^2; + \frac{1}{2} \sum_{I} M_I \left | \dot{\vec{R}}_{I} \right | ^2 - E \left [\bigl[\psi_{i,\sigma}(\vec{r}) \bigr] , \bigl[\vec{R}_I \bigr] \right] \\ + \sum_{ij,\sigma} \Lambda_{ij,\sigma} \left (\int d\vec{r}\ \psi_{i,\sigma}^{*}(\vec{r}) \psi_{j,\sigma}(\vec{r}) - \delta_{ij\sigma} \right)$$

Which generates a dynamics for the wavefunctions $\psi_{i,\sigma}(\vec{r})$ and atoms positions $\vec{R}_I$ through the constrained equations of motion:

$$\begin{matrix}\mu \ddot{\psi}_{i,\sigma}(\vec{r},t) = -\frac{\delta E}{\delta \psi_{i,\sigma }^{*} \left(\vec{r},t \right) } + \sum\limits_j \Lambda_{ij,\sigma} \psi_{j,\sigma} \left(\vec{r},t \right) \end{matrix}$$
$$\begin{matrix}M_I \ddot{\vec{R}}_I = -\frac{\partial E}{\partial \vec{R}_I} \end{matrix}$$

where μ is the fictitious mass for the electronic degrees of freedom andM_I are the ionic masses. The adjustable parameter μ is used to describe the relative rate at which the wavefunctions change with time.$\Lambda_{ij,\sigma}$ are the Lagrangian multipliers for the orthonormalization of the single-particle orbitals $\psi_{i,\sigma}(\vec{r})$. They are defined by the orthonormalization constraint conditions and can be rigorously found. However, the equations of motion for the Lagrange multipliers depend on the specific algorithm used to integrate the Eqns. above.

For this method to give ionic motions that are physically meaningful the kinetic energy of the Kohn-Sham orbitals must be relatively small when compared to the kinetic energy of the ions. There are two ways where this criterion can fail. First, the numerical integrations for the Car-Parrinello equations of motion can often lead to large relative values of the kinetic energy of the Kohn-Sham orbitals relative to the kinetic energy of the ions. This kind of failure is easily fixed by requiring a more accurate numerical integration, i.e. use a smaller time step for the numerical integration. Second, during the motion of the system a the ions can be in locations where there is an Kohn-Sham orbital level crossing, i.e. the density-functional energy can have two states that are nearly degenerate. This kind of failure often occurs in the study of chemical reactions. This kind of failure is not easily fixed and requires the use of a more sophisticated density-functional energy that accounts for low-lying excited electronic states.

12.1.21.1 Verlet Algorithm for Integration

Integrating the Eqns. above using the Verlet algorithm results in

$$\begin{matrix}\psi_{i,\sigma}^{t+ \Delta t} \leftarrow 2 \psi_{i,\sigma}^{t} - \psi_{i,\sigma}^{t-\Delta t} + \frac{(\Delta t)^2}{\mu} \left[\frac{\delta E}{\delta \psi_{i,\sigma}^{*}} + \sum_{j} \psi_{j,\sigma} \Lambda_{ji,\sigma} \right]_{t} \end{matrix}$$
$$\begin{matrix}\vec{R}_I^{t+\Delta t} \leftarrow 2 \vec{R}_I^{t} - \vec{R}_I^{t-\Delta t} + \frac{(\Delta t)^2}{M_I} \frac{\partial E} {\partial \vec{R}_I}\end{matrix}$$

In this molecular dynamic procedure we have to know variational derivative

$$\frac{\delta E}{\delta \psi_{i,\sigma}^{*}}$$

and the matrix $\Lambda_{ij,\sigma}$.
The variational derivative

$$\frac{\delta E}{\delta \psi_{i,\sigma}^{*}}$$

can be analytically found and is

$$\begin{aligned}\frac{\delta E}{\delta \psi_{i,\sigma}^{*}} = -\frac{1}{2} \nabla^2 \psi_{i,\sigma}(\vec{r}) \\ + \int d\vec{r^{\prime}} W_{ext}(\vec{r},\vec{r^{\prime}}) \psi_{i,\sigma}(\vec{r^{\prime}}) \\ + \int d\vec{r^{\prime}} \frac{n(\vec{r^{\prime}})}{|\vec{r}-\vec{r^{\prime}}|} \psi_{i,\sigma}(\vec{r})\\ + \mu_{xc}^{\sigma}(\vec{r}) \psi_{i,\sigma}(\vec{r}) \\ \equiv \hat{H} \psi_{i,\sigma} \end{aligned}$$

To find the matrix $\Lambda_{ij,\sigma}$ impose the orthonormality constraint on$\psi_{i,\sigma}^{t+\Delta t}$ obtain a matrix Riccatti equation, and then Riccatti equation is solved by an iterative solution.

12.1.21.2 Constant Temperature Simulations: Nose-Hoover Thermostats

Nose-Hoover Thermostats for the electrons and ions can also be added to the Car-Parrinello simulation. In this type of simulation thermostats variables x_e and x_R are added to the simulation by adding the auxiliary energy functionals to the total energy.

$$ION_{THERMOSTAT}(x_R) = \frac{1}{2} Q_R \dot{x_R} + E_{R0}x_R$$
$$ELECTRON_{THERMOSTAT}(x_e) = \frac{1}{2} Q_e \dot{x_e} + E_{e0}x_e$$

In these equations, the average kinetic energy for the ions is

$$\begin{matrix}E_{R0} = \frac{1}{2} f k_B T \end{matrix}$$

where f is the number of atomic degrees of freedom,k_B is Boltzmann's constant, and T is the desired temperature. Defining the average fictitious kinetic energy of the electrons is not as straightforward. Blöchl and Parrinello (P.E. Blöchl and M. Parrinello, Phys. Rev. B, **45**, 9413, (1992)) have suggested the following formula for determining the average fictitious kinetic energy

$$\begin{matrix}E_{e0} = 4 k_B T \frac{\mu}{M} \sum_{i} <\psi_i|-\frac{1}{2} \nabla^2 |\psi_{i}> \end{matrix}$$

where μ is the fictitious electronic mass, M is average mass of one atom, and $\sum_{i} <\psi_i|-\frac{1}{2} \nabla^2 |\psi_{i}>$ is the kinetic energy of the electrons.

Blöchl and Parrinello suggested that the choice of mass parameters,Q_e), and Q_R) should be made such that the period of oscillating thermostats should be chosen larger than the typical time scale for the dynamical events of interest but shorter than the simulation time.

$$P_{ion} = 2\pi \sqrt{\frac{Q_R}{4E_{R0}}}$$
$$P_{electron} = 2\pi \sqrt{\frac{Q_e}{4E_{e0}}}$$

where P_{ion} and $P_{electron}$ are the periods of oscillation for the ionic and fictitious electronic thermostats.

In simulated annealing simulations the electronic and ionic Temperatures are scaled according to an exponential cooling schedule,

$$T_e(t) = T_e^0 \exp^{-\frac{t}{\tau_e}}$$
$$T_{ionic}(t) = T_{ionic}^0 \exp^{-\frac{t}{\tau_{ionic}}}$$

where T_e^0 and T_{ionic}^0 are the initial temperatures, and τ_e and τ_{ionic} are the cooling rates in atomic units.

12.1.22 NWPW Tutorial 1: S_2 dimer examples with PSPW

A description of all the examples in NWPW Tutorial 1 can be found in the attached pdf nwpwexample1.pdf

12.1.22.1 Total energy of S_2 dimer with LDA approximation

(input:Media:s2-example1.nw, output:Media:s2-example1.nwout)

In this example, the total energy of the S_2 dimer using LDA approximation for the exchange-correlation functional is calculated.

```
echo
title "total energy of s2-dimer LDA/30Ry with PSPW method"
scratch_dir   ./scratch
permanent_dir ./perm
start s2-pspw-energy
geometry
S 0.0 0.0 0.0
S 0.0 0.0 1.88
end
nwpw
  simulation_cell
    SC 20.0
  end
  cutoff 15.0
  mult 3
  xc lda
  lmbfgs
end
task pspw energy
```

The energies from the simulation will be

```
...
==  Summary Of Results  ==

 number of electrons: spin up=    7.00000  down=    5.00000 (real space)

 total     energy    : -0.2041363137E+02 (   -0.10207E+02/ion)
 total orbital energy:  -0.4944372503E+01 (   -0.41203E+00/electron)
 hartree   energy    :  0.1680529987E+02 (    0.14004E+01/electron)
 exc-corr  energy    : -0.4320620600E+01 (   -0.36005E+00/electron)
 ion-ion   energy    :  0.8455644190E-02 (    0.42278E-02/ion)

 kinetic (planewave) :  0.7529965882E+01 (    0.62750E+00/electron)
 V_local (planewave) : -0.4506036741E+02 (   -0.37550E+01/electron)
 V_nl    (planewave) :  0.4623635248E+01 (    0.38530E+00/electron)
 V_Coul  (planewave) :  0.3361059973E+02 (    0.28009E+01/electron)
 V_xc.   (planewave) : -0.5648205953E+01 (   -0.47068E+00/electron)
 Virial Coefficient  : -0.1656626150E+01

 orbital energies:
   -0.2001309E+00 (  -5.446eV)
   -0.2001309E+00 (  -5.446eV)
   -0.3294434E+00 (  -8.965eV)    -0.2991148E+00 (  -8.139eV)
   -0.3294435E+00 (  -8.965eV)    -0.2991151E+00 (  -8.139eV)
   -0.3582269E+00 (  -9.748eV)    -0.3352434E+00 (  -9.123eV)
   -0.5632339E+00 ( -15.326eV)    -0.5246249E+00 ( -14.276eV)
   -0.7642738E+00 ( -20.797eV)    -0.7413909E+00 ( -20.174eV)

 Total PSPW energy   : -0.2041363137E+02
...
```

12.1.22.2 Structural optimization of S_2 dimer with LDA approximation

(input:Media:s2-example2.nw, output:Media:s2-example2.nwout)

In this example, the structure of the S_2 dimer using results generated from prior energy calculation is calculated. Since most of the parameters are already stored in the run-time database the input is very simple.

```
echo
title "optimization of s2-dimer LDA/30Ry with PSPW method"
scratch_dir   ./scratch
permanent_dir ./perm
restart s2-pspw-energy
driver
  maxiter 20
  xyz s2
end
task pspw optimize
```

As the optimization process consists of series of total energy evaluations the contents of the output file are very much similar to that in Example I. At each step the total energy and force information will be outputed as follows

```
 Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
 ---- ---------------- -------- -------- -------- -------- -------- --------
@    1    -20.41364254 -7.1D-05  0.00004  0.00004  0.00605  0.01048      7.8
```

The best way to keep track of the optimization calculation is to run the following grep command on the output file.

```
grep @ outputfile

@ Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
@ ---- ---------------- -------- -------- -------- -------- -------- --------
@    0    -20.41357202  0.0D+00  0.00672  0.00672  0.00000  0.00000      1.5
@    1    -20.41364254 -7.1D-05  0.00004  0.00004  0.00605  0.01048      7.8
@    2    -20.41364256 -2.3D-08  0.00020  0.00020  0.00003  0.00005      9.7
@    2    -20.41364256 -2.3D-08  0.00020  0.00020  0.00003  0.00005      9.7
```

The optimized energy and geometry will be

```
...
     ----------------------
     Optimization converged
     ----------------------


  Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
  ---- ---------------- -------- -------- -------- -------- -------- --------
@    2    -20.41364256 -2.3D-08  0.00020  0.00020  0.00003  0.00005      9.7
                           ok       ok       ok       ok


                        Z-matrix (autoz)
                        --------

  Units are Angstrom for bonds and degrees for angles

     Type          Name      I    J    K    L    M      Value     Gradient
     ----------- --------  ----- ----- ----- ----- ----- ---------- ----------
   1 Stretch                  1     2                    1.89115    0.00020


                    Geometry "geometry" -> "geometry"
                    ---------------------------------

  Output coordinates in angstroms (scale by  1.889725989 to convert to a.u.)

  No.       Tag         Charge          X              Y              Z
 ---- ---------------- ---------- -------------- -------------- --------------
    1 S               16.0000     0.00000000     0.00000000    -0.94557591
    2 S               16.0000     0.00000000     0.00000000     0.94557591

...
```

12.1.22.3 Frequency calculation of S_2 dimer with LDA approximation

(input:Media:s2-example3.nw, output:Media:s2-example3.nwout)

In this example, the vibrational frequency of the S_2 dimer using results generated from prior geometry optimization is calculated. Since most of the parameters are already stored in the run-time database the input is very simple.

```
echo
title "frequency calculation of s2-dimer LDA/30Ry with PSPW method"
scratch_dir   ./scratch
permanent_dir ./perm
restart s2-pspw-energy
freq
  animate
end
task pspw freq
```

The frequency and thermodynamic analysis generated

```
...
 Temperature                   =   298.15K
 frequency scaling parameter   =   1.0000


 Linear Molecule

 Zero-Point correction to Energy  =    1.034 kcal/mol  (  0.001647 au)
 Thermal correction to Energy     =    2.579 kcal/mol  (  0.004110 au)
 Thermal correction to Enthalpy   =    3.171 kcal/mol  (  0.005054 au)

 Total Entropy                 =   52.277 cal/mol-K
  - Translational              =   38.368 cal/mol-K (mol. weight =  63.9441)
  - Rotational                 =   13.630 cal/mol-K (symmetry #  =       2)
  - Vibrational                =    0.279 cal/mol-K

 Cv (constant volume heat capacity) =    5.750 cal/mol-K
  - Translational              =    2.979 cal/mol-K
  - Rotational                 =    1.986 cal/mol-K
  - Vibrational                =    0.785 cal/mol-K
...
 ------------------------------------------------------------------------------
 Normal Eigenvalue ||           Projected Infra Red Intensities
  Mode   [cm**-1]  || [atomic units] [(debye/angs)**2] [(KM/mol)] [arbitrary]
 ------ ---------- || -------------- ----------------- ---------- -----------
    1       0.000 ||    0.000030          0.001        0.029       0.000
    2       0.000 ||    2.466908         56.914     2404.864      15.000
    3       0.000 ||    2.466908         56.914     2404.864      15.000
    4       0.000 ||    2.466908         56.914     2404.864      15.000
    5       0.000 ||    2.466908         56.914     2404.864      15.000
    6     723.419 ||    0.000000          0.000        0.000       0.000
 ------------------------------------------------------------------------------
...
```

12.1.22.4 Ab initio molecular dynamics simulation (Car-Parrinello) of S_2 dimer using the LDA approximation

(input:Media:s2-example4.nw, output:Media:s2-example4.nwout Media:s2-md.xyz Media:s2-md.emotion.dat)

In this example, a constant energy Car-Parrinello simulation of S_2 dimer using LDA approximation is calculated. A brief introduction to the Car-Parrinello method can be found in cpmd-lecture.pdf

```
echo
title "AIMD simulation of s2-dimer"
scratch_dir   ./scratch
permanent_dir ./perm
start s2-md
geometry
S 0.0 0.0 0.0
S 0.0 0.0 1.95
end
nwpw
  simulation_cell
    SC 20.0
  end
  cutoff 15.0
  mult 3
  xc lda
  lmbfgs
  car-parrinello
    time_step 5.0
    fake_mass 600.0
    loop 1 1000
    xyz_filename s2-md.xyz
  end
end
task pspw energy
task pspw car-parrinello
```

A plotting program (e.g. gnuplot, xmgrace) can be used to look at the total, potential, kinetic energies, contained in the s2-md.emotion file (see section EMOTION motion file for datafile format) i.e.,

```
seattle-1604% gnuplot

        G N U P L O T
        Version 4.0 patchlevel 0
        last modified Thu Apr 15 14:44:22 CEST 2004
        System: Linux 2.6.18-194.8.1.el5

        Copyright (C) 1986 - 1993, 1998, 2004
        Thomas Williams, Colin Kelley and many others

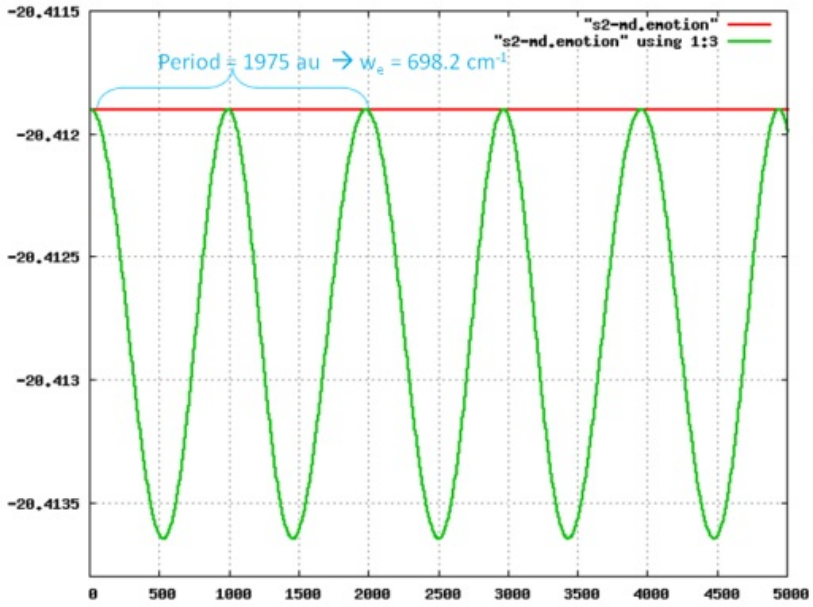
        This is gnuplot version 4.0.  Please refer to the documentation
        for command syntax changes.  The old syntax will be accepted
        throughout the 4.0 series, but all save files use the new syntax.

        Type help to access the on-line reference manual.
        The gnuplot FAQ is available from
                <http://www.gnuplot.info/faq/>

        Send comments and requests for help to
                <gnuplot-info@lists.sourceforge.net>
        Send bugs, suggestions and mods to
                <gnuplot-bugs@lists.sourceforge.net>


Terminal type set to 'x11'
gnuplot> plot "s2-md.emotion","s2-md.emotion" using 1:3
gnuplot>
```

The following plot shows the Car-Parrinello \(^3\Sigma_g^-\) S\(_2\) energy surface generated from the simulation.



12.1.22.5 Ab initio molecular dynamics simulation (Born-Oppenheimer) of S<sub2 dimer using the LDA approximation

(input:Media:s2-example5.nw, output:Media:s2-example5.nwout Media:s2-bomd.xyz Media:s2-bomd.emotion.dat) In this example, a constant energy Born-Oppenheimer simulation of S2 dimer using LDA approximation is calculated.

```
title "AIMD simulation of s2-dimer"
echo

scratch_dir   ./scratch
permanent_dir ./perm

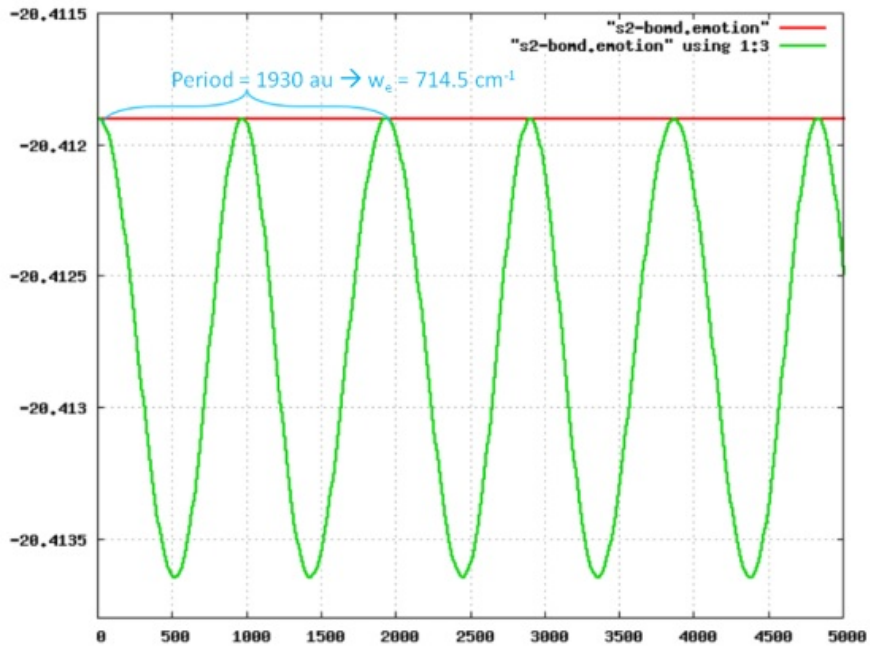
start s2-bomd

geometry
S 0.0 0.0 0.0
S 0.0 0.0 1.95
end

nwpw
  simulation_cell
    SC 20.0
  end
  cutoff 15.0
  mult 3
  xc lda
  lmbfgs
end
task pspw energy

nwpw
   bo_steps 1 500
   bo_time_step 10.0
end
task pspw born-oppenheimer
```

The following plot shows the \(^3\Sigma_g^{-}_{2}\) energy surface generated from the simulation.

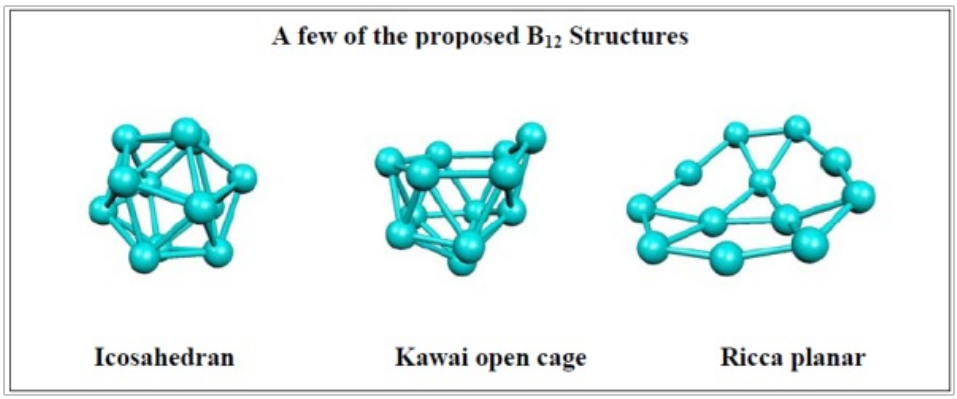


12.1.23 NWPW Tutorial 2: Using PSPW Car-Parrinello Simulated Annealing Simulations to Optimize Structures

In principle quantum mechanical calculations can be used to determine the structure of any chemical system. One chooses a structure, calculates the total energy of the system, and repeats the calculation for all possible geometries. Of course the major limitation of this approach is that the number of local minima structures increases dramatically with system size and the cost of quantum mechanical calculations also increases dramatically with system size. Not surprisingly most quantum mechanical calculations limit the number of structures to be calculated by using experimental results or chemical intuition. One could speed up the calculations by using simplified inter-atomic force fields instead of quantum mechanical calculations. However, inter-atomic forces fields have many simplifying assumptions that can severely limit their predictability. Another approach is to use ab initio molecular dynamics methods combined with simulated annealing. These methods are quite robust and allow strongly interacting many body systems to be studied by direct dynamics simulation

without the introduction of empirical interactions. In these methods, the atomic forces are calculated from ab initio calculations that are performed “on-the-fly” as the molecular dynamics trajectory is generated.

The following examples demonstrate how to use the ab initio molecular dynamics methods and simulated annealing strategies of NWChem to determine the lowest energy structures of the B_{12} cluster. This example is based on a study performed by Kiran Boggavarapu et al.. One might expect from chemical intuition that lowest energy structure of B_{12} will be an icosahedron, since B_{12} icosahedra are a common structural unit found in many boron rich materials. Despite this prevalence, ab initio calculations performed by several researchers have suggested that B_{12}, as well as B_{12}^{+} and B_{12}^{-}, will have a more open geometry.



12.1.23.1 Simulated Annealing Using Constant Energy Simulation

(input:Media:b12-example2a.nw, output:Media:b12-example2a.nwout Media:b12.00.xyz Media:b12.00.emotion.dat Media:b12.01.xyz Media:b12.01.emotion.dat)

This example uses a series of constant energy Car-Parrinello simulations with velocity scaling to do simulated annealing. The initial four Car-Parrinello simulations are used to heat up the system to several thousand Kelvin. Then the system is cooled down thru a series of constant energy simulations in which the electronic and ionic velocities are scaled by 0.99 at the start of each Car-Parrinello simulation. Energy minimization calculations are used periodically in this simulation to bring the system back down to Born-Oppenheimer surface. This is necessary because of electronic heating.

The Car-Parrinello keyword “scaling” scales the wavefunction and ionic velocities at the start of the simulation. The following input is used to increase the ionic velocities by a factor of two at the start of the Car-Parrinello simulation.

Key Input

```
...
Car-Parrinello
fake_mass 500.0
time_step 5.0
loop 10 100
** scaling 1.0 2.0**
emotion_filename b12.00.emotion
xyz_filename     b12.00.xyz
end
...
```

Output

```
...
    wavefnc cutoff= 10.000  fft= 42x 42x 42(     6027 waves     1004 per task)

technical parameters:
    translation contrained
    time step=      5.00     ficticious mass=     500.0
    **cooling/heatting rates: 0.10000E+01 (psi)
0.20000E+01
(ion)**
    maximum iterations =    1000 (   10 inner  100 outer )
    initial kinetic energy:  0.99360E-05 (psi)   0.27956E-03 (ion)
                                 0.20205E-28 (c.o.m.)
    **after scaling: 0.99360E-05 (psi) 0.11182E-02
(ion)**
    **increased energy: 0.00000E+00 (psi)
0.83868E-03 (ion)**

Constant Energy Simulation
...
```

The program checks to see if the initial input ionic velocities have a non-zero center of mass velocity. If there is a non-zero center of mass velocity in the system then by default the program removes it. To turn off this feature set the following

```
nwpw
  translation on
end
```

or

```
set nwpw:com_shift .false.
```

12.1.23.2 Simulated Annealing Using Constant Temperature Simulation

(input:Media:b12-example2b.nw, output:Media:b12-example2b.nwout Media:b12.10.xyz Media:b12.10.emotion.dat Media:b12.11.xyz.gz Media:b12.11.emotion.dat)

(mpeg movie of simulation: Media:boron.mpg)

The simulated annealing calculation in this example uses a constant temperature Car-Parrinello simulation with an exponential cooling schedule,

$$T(t)=T_0e^{-t/\tau}$$

where T0 and τ are an initial temperature and a time scale of cooling, respectively. In the present calculations T0=3500K and τ=4.134e+4 au (1.0 ps) were used and the thermostat masses were kept fixed to the initial values determined by T=Te=3500K and (2π/ω)=250 a.u. (6 fs). Annealing proceeded for 50000 steps, until a temperature of 10K was reached. After which, the metastable structure is optimized using the driver optimizer. The keyword SA_decay is used to enter the decay rates, τelectron and τion, used in the simulated annealing algorithm in the constant temperature car-parrinello simulation. The decay rates are in units of au (conversion 1 au = 2.41889e-17 seconds).

Key Input

```
....
Car-Parrinello
SA_decay 4.134d4 4.134d4 #decay rate in units of au (1au=2.41889e-17seconds)
....
```

12.1.24 NWPW Tutorial 3: using isodesmic reaction energies to estimate gas-phase thermodynamics

(isodesmic.pdf isodesmic.tgz)

The development of a computational scheme that can accurately predict reaction energies requires some care. As shown in Table 1 energy errors associated with ab initio calculations can be quite high. Even though ab initio electronic structure

methods are constantly being developed and improved upon, these methods are rarely able to give heat of formations of a broad class of molecules with error limits of less than a few kcal/mol. Only when very large basis sets such as the correlation-consistent basis sets, high level treatments of correlation energy such as coupled cluster methods (CCSD(T)), and small correction factors such as core-valence correlation energies and relativistic effects, are included will the heat of formation from ab initio electronic structure methods be accurate to within one kcal/mol. Although one can now accurately calculate the heats of formation of molecules with up to 6 first row atoms, such high-level calculations are extremely demanding and scale computationally as N^7 for N basis functions.

Examples of these types of large errors are shown in the following Table, where the enthalpies of formation of CCl_3SH are calculated by using atomization energies from different levels of ab initio theory.

	MP2/cc-pVDZ	LDA/DZVP2	BP91/DZVP2	B3LYP/DZVP2	G2 Theory
$\Delta H\ (_f^o)$	+4.9	-80.0	-2.6	+26.5	-13.0

Table 1: Standard enthalpy of formation ($\Delta H(_f^o)$(298K) for CCl_3SH in kcal/mol from atomization energies with various electronic structure methods. Results taken from reference [2].

Differences of up to 106.5 kcal/mol are found between different levels of theory. This example demonstrates that care must be taken in choosing the appropriate method for calculating the heats of formation from total atomization energies.

The difficulties associated with calculating absolute heats of formation from atomization energies can be avoided by using a set of isodesmic reactions[1]. The defining property of an isodesmic reaction - that there are an equal number of like bonds on the left-hand and right-hand sides of the reaction - helps to minimize the error in the reaction energy. These reactions are designed to separate out the interactions between molecular subsistents and non-bonding electrons from the direct bonding interactions by having the direct bonding interactions largely canceling one another. This separation is quite attractive. Most ab initio methods give substantial errors when estimating direct bonding interactions due to the computational difficulties associated with electron pair correlation, whereas ab initio methods are expected to be more accurate for estimating neighboring interactions and long-range through-bond effects.

The following isodesmic reaction can be used determine the enthalpy of formation for CCl_3SH that is significantly more accurate than the estimates based on atomization energies.

$CCl_3SH + CH_4 \rightarrow CH_3SH + CCl_3H$, ΔH_r(calc).

The first step is to calculate the reaction enthalpy of this reaction from electronic, thermal and vibrational energy differences at 298.15K at a consistent level of theory. The defining property of an isodesmic reaction that there are an equal number of like bonds on the left-hand and right-hand sides of the reaction helps to minimize the error in the calculation of the reaction energy. The enthalpy of formation of CCl_3SH can then be calculated by using Hess's law with the calculated enthalpy change and the experimentally known heats of formation of the other 3 species (see Table 3).

$\Delta H_f(CCl_3SH) = \Delta H_f(CH_3SH)(exp) + \Delta H_f(CCl_3H)(exp) - \Delta H_f(CH_4))(exp) - \Delta H_r$(calc).

In this example, try to design and run NWPW simulations that can be used to estimate the enthalpy of formation for CCl_3SH using its atomization energy and using the reaction enthalpy of the isodesmic reaction and compare your results to Table 2. Be careful to make sure that you use the same cutoff energy for all the simulations (.e.g. cutoff 35.0). You might also try to estimate enthalpies of formation for $CHCl_2SH$ and CH_2ClSH. Also try designing simulations that use the SCF, DFT, MP2, and TCE modules.

$CCl_3SH + CH_4 \rightarrow CH_3SH + CCl_3H$

Un-optimized geometries for CCl_3SH, CH_3SH, CCl_3H and CH_4 which are needed to design your simulations are contained in the file Media:thermodynamics.xyz. You will also need to calculate the energies for the H, C, S, and Cl atoms to calculate the atomization energies. The multiplicities for these atoms are 2, 3, 3 and 2 respectively. You

will also need to calculate the enthalpy of a molecule. The enthalpy of a molecule at 298.15K is sum of the total energy and a thermal correction to the enthalpy. A good estimate for the thermal correction to the enthalpy can be obtained from a frequency calculation, i.e.

H = E + H\(_{correction}\)

Thermodynamic output from a frequency calculation:

```
Temperature                 =   298.15K
frequency scaling parameter     =   1.0000

Zero-Point correction to Energy  =   27.528 kcal/mol  (  0.043869 au)
Thermal correction to Energy     =   29.329 kcal/mol  (  0.046739 au)
```

The following line contains the value for H\(_{correction}\)

```
Thermal correction to Enthalpy   =   29.922 kcal/mol  (  0.047683 au)

Total Entropy               =   44.401 cal/mol-K
 - Translational            =   34.246 cal/mol-K (mol. weight =  16.0313)
 - Rotational               =   10.060 cal/mol-K (symmetry #  =       12)
 - Vibrational              =    0.095 cal/mol-K

Cv (constant volume heat capacity) =    6.503 cal/mol-K
 - Translational            =    2.979 cal/mol-K
 - Rotational               =    2.979 cal/mol-K
 - Vibrational              =    0.544 cal/mol-K
```

Compounds	MP2/cc-pVDZ	LDA/DZVP2	BP91/DZVP2	B3LYP/DZVP2	G2	Experiment
	(isodesmic)	(isodesmic)	(isodesmic)	(isodesmic)	(atomization)	
CCl$_3SH	-13.40	-11.86	-8.68	-7.64	-12.95	
CHCl\(_2\)SH	-11.48	-11.07	-8.66	-7.92	-11.52	
CH\(_2\)ClSH	-7.01	-6.66	-5.44	-5.20	-6.98	
CH\(_3\)SH					-4.76	-5.34

Compounds	ΔH\(_f^o\)(298K)
H	52.095
C	171.291
S	66.636
Cl	29.082
CCl\(_4\)	-24.59
CCl\(_3\)H	-24.65
CCl\(_2\)H\(_2\)	-22.10
CClH\(_3\)	-19.32
CH\(_4\)	-17.88
CH\(_3\)SH	-5.34

Table 1: Gas-phase standard enthalpies of formation (ΔH

Table 3: Miscellaneous experimental gas-phase enthalpies of formation (kcal/mol) taken from [3].

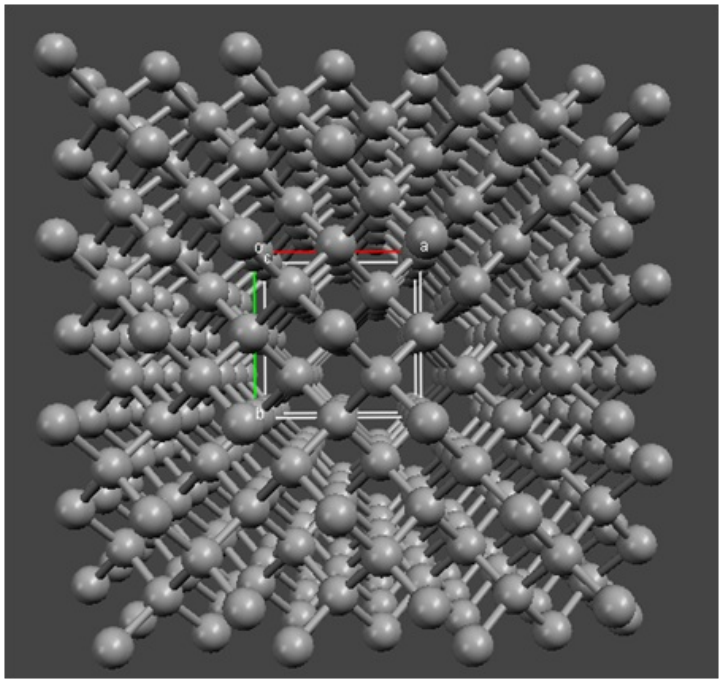
1. Hehre, W. J., L. Radom, P.v.R. Schleyer, and J.A. Pople Ab Initio Molecular Orbital Theory; John Wiley & Sons: New York, 1986).
2. E.J. Bylaska, D.A. Dixon, and A.R. Felmy(2000), “The Free Energies of Reactions of Chlorinated Methanes with Aqueous Monovalent Anions: Application of ab initio Electronic Structure Theory”, J. Phys. Chem. A, 104(3), 610-617.
3. Chase, M. W., Jr. Phys. Chem. Ref. Data, Monograph No. 9 1998, 9, 1-1951.

12.1.25 NWPW Tutorial 4: AIMD/MM simulation of CCl\(_4\) + 64 H\(_2\)O

(input:Media:ccl4-64water.nw, output:Media:ccl4-64water.nwout)

In this section we show how use the PSPW module to perform a Car-Parrinello AIMD/MM simulation for a CCl\(_4\) molecule in a box of 64 H\(_2\)O. Before running a PSPW Car-Parrinello simulation the system should be on the Born-Oppenheimer surface, i.e. the one-electron orbitals should be minimized with respect to the total energy (i.e. task pspw energy). In this example, default pseudopotentials from the pseudopotential library are used for C, Cl, O^ and H^, exchange correlation functional is PBE96, The boundary condition is periodic, and with a side length of 23.577 Bohrs and has a cutoff energy is 50 Ry). The time step and fake mass for the Car-Parrinello run are specified to be 5.0 au and 600.0 au, respectively.

12.1.26 NWPW Tutorial 5: Optimizing the Unit Cell and Geometry of Diamond



The PSPW and BAND codes can be used to determine structures and energies for a wide range of crystalline systems. It can also be used to generate band structure and density of state plots.

12.1.26.1 Optimizing the Unit Cell and Geometry for an 8 Atom Supercell of Diamond with PSPW

(input:Media:diamond-pspw.nw, output:Media:diamond-pspw.nwout, Media:diamond.opt.cif)

(input:Media:catom-pspw.nw, output:Media:catom-pspw.nwout)

The following example uses the PSPW module to optimize the unit cell and geometry for a diamond crystal. The fractional coordinates and the unit cell are defined in the geometry block. The simulation_cell block is not needed since NWPW automatically uses the unit cell defined in the geometry block.

```
title "Diamond 8 atom cubic cell - geometry and unit cell optimization"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamond

memory 950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 3.56d0
    lat_b 3.56d0
    lat_c 3.56d0
    alpha 90.0d0
    beta  90.0d0
    gamma 90.0d0
  end
  C -0.50000d0 -0.50000d0 -0.50000d0
  C  0.00000d0  0.00000d0 -0.50000d0
  C  0.00000d0 -0.50000d0  0.00000d0
  C -0.50000d0  0.00000d0  0.00000d0
  C -0.25000d0 -0.25000d0 -0.25000d0
  C  0.25000d0  0.25000d0 -0.25000d0
  C  0.25000d0 -0.25000d0  0.25000d0
  C -0.25000d0  0.25000d0  0.25000d0
end

nwpw
  ewald_rcut 3.0
  ewald_ncut 8  #The default value of 1 needs to be increased for small cells
  lmbfgs
  xc pbe96
end

driver
  clear
  maxiter 40
end

set nwpw:cif_filename diamond.opt  # create a CIF file containing optimization history
set includestress .true.           # this option tells driver to optimize the unit cell
task pspw optimize ignore
```

The optimized energy and geometry will be

...

Optimization converged

Step	Energy	Delta E	Gmax	Grms	Xrms	Xmax	Walltime
@ 6	-45.07688304	-1.1D-07	0.00037	0.00021	0.00002	0.00003	174.5
		ok	ok	ok	ok		

Geometry "geometry" -> "geometry"

Output coordinates in angstroms (scale by 1.889725989 to convert to a.u.)

No.	Tag	Charge	X	Y	Z
1	C	6.0000	1.82723789	1.82729813	1.82705440
2	C	6.0000	0.00000857	-0.00006053	1.82730027
3	C	6.0000	-0.00000584	1.82706061	0.00002852
4	C	6.0000	1.82712018	0.00006354	-0.00002544
5	C	6.0000	2.74074195	2.74072805	2.74088522
6	C	6.0000	0.91366407	0.91370055	2.74064976
7	C	6.0000	0.91351181	2.74080771	0.91352917
8	C	6.0000	2.74078843	0.91348115	0.91365446

Lattice Parameters

---

lattice vectors in angstroms (scale by 1.889725989 to convert to a.u.)

a1=< 3.654 0.000 0.000 >
a2=< 0.000 3.654 0.000 >
a3=< 0.000 0.000 3.654 >
a= 3.654 b= 3.654 c= 3.654
alpha= 90.000 beta= 90.000 gamma= 90.000
omega= 48.8

reciprocal lattice vectors in a.u.

b1=< 0.910 0.000 0.000 >
b2=< 0.000 0.910 0.000 >
b3=< 0.000 0.000 0.910 >

Atomic Mass

C 12.000000

==========

internuclear distances

---

center one	center two	atomic units	angstroms
5 C	1 C	2.99027	1.58238
6 C	1 C	2.99027	1.58238
6 C	2 C	2.99027	1.58238
7 C	1 C	2.99026	1.58238
7 C	3 C	2.99027	1.58238
8 C	1 C	2.99027	1.58238
8 C	4 C	2.99027	1.58238

---

number of included internuclear distances: 7

==========

==========

internuclear angles

---

center 1	center 2	center 3	degrees
5 C	1 C	6 C	109.46
5 C	1 C	7 C	109.48
5 C	1 C	8 C	109.48
6 C	1 C	7 C	109.47
6 C	1 C	8 C	109.46
7 C	1 C	8 C	109.48
1 C	6 C	2 C	109.48
1 C	7 C	3 C	109.47
1 C	8 C	4 C	109.47

---

number of included internuclear angles: 9

========== ...

The C-C bond distance after the geometry optimization is 1.58 Angs. and agrees very well with the experimental value of 1.54 Angs.. Another quantity that can be calculated from this simulation is the cohesive energy.The cohesive energy of a crystal is the energy needed to separate the atoms of the solid into isolated atoms, i.e.

$$ E_{coh}= -\left(E_{solid} - \sum_{a} E_{atom}^a \right) $$

where E_{solid} is the energy of the solid andE_{atom}^a are the energies of the isolated atoms. In order to calculate the cohesive energy the energy of an isolated carbon atom at the same level of theory and cutoff energy will need to be calculated. The following input can be used to the energy of an isolated carbon atom.

(input:file:catom-pspw.nw, output:file:catom-pspw.nwout)

```
title "triplet carbon atom at pbe96 level using a large unit cell"
start c1-pspw
memory 1400 mb

permanent_dir ./perm
scratch_dir   ./scratch

geometry
C 0 0 0
end

nwpw
  simulation_cell
    FCC 38.0     #large unit cell
    boundary_conditions   periodic  # periodic boundary conditions are used by default.
    #boundary_conditions aperiodic  # free-space (or aperiodic) boundary conditions could also be used.
  end
  xc pbe96
  mult 3
 lmbfgs
end
task pspw energy
```

The total energy from the simulation will be

```
 Total PSPW energy : -0.5421213534E+01
```

Using this energy and energy of diamond the cohesive energy per atom is calculated to be

$$\begin{align} E_{coh}= -\left(-45.07688304au/8 - (-5.421213534au) \right)= 0.2133968 au = 5.8 eV \end{align}$$

This value is substantially lower than the experimental value of$7.37eV$! It turns out this error is a result of the unit cell being too small for the diamond calculation (or too small of a Brillioun zone sampling). In the next section, we show how increasing the Brillouin zone sampling reduces the error in the calculated cohesive energy.

12.1.26.2 Optimizing the Unit Cell for an 8 Atom Supercell of Diamond with BAND

(input:Media:diamond-band.nw, output:Media:diamond-band.nwout)

In this example the BAND module is used to optimize the unit cell and geometry for a diamond crystal at different Brillouin zone samplings.

```
title "Diamond 8 atom cubic cell - geometry and unit cell optimization"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamond-band

memory 1950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 3.58d0
    lat_b 3.58d0
    lat_c 3.58d0
    alpha 90.0d0
    beta  90.0d0
    gamma 90.0d0
  end
  C -0.50000d0 -0.50000d0 -0.50000d0
  C  0.00000d0  0.00000d0 -0.50000d0
  C  0.00000d0 -0.50000d0  0.00000d0
  C -0.50000d0  0.00000d0  0.00000d0
  C -0.25000d0 -0.25000d0 -0.25000d0
  C  0.25000d0  0.25000d0 -0.25000d0
  C  0.25000d0 -0.25000d0  0.25000d0
  C -0.25000d0  0.25000d0  0.25000d0
end
set includestress    .true.   # option tells driver to optimize the unit cell
set nwpw:zero_forces .true.   # option zeros the forces on the atoms--> only lattice parameters optimized

nwpw
  ewald_rcut 3.0
  ewald_ncut 8    #The default value of 1 needs to be increased
  lmbfgs
  xc pbe96
end

#1x1x1 k-point mesh
nwpw
  monkhorst-pack 1 1 1
end
set nwpw:cif_filename diamond111.opt
driver; clear; maxiter 40; end; task band optimize ignore

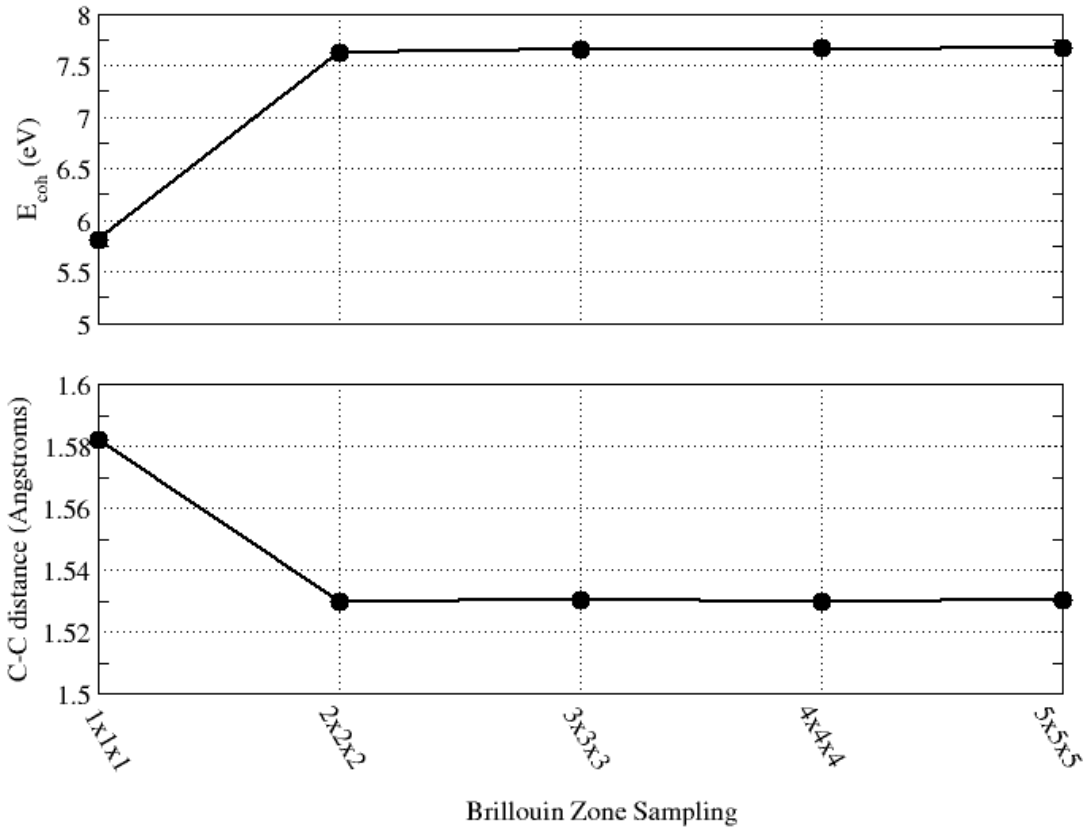
#2x2x2 k-point mesh
nwpw
  monkhorst-pack 2 2 2
end
set nwpw:cif_filename diamond222.opt
driver; clear; maxiter 40; end; task band optimize ignore

#3x3x3 k-point mesh
nwpw
  monkhorst-pack 3 3 3
end
set nwpw:cif_filename diamond333.opt
driver; clear; maxiter 40; end; task band optimize ignore

#4x4x4 k-point mesh
nwpw
  monkhorst-pack 4 4 4
end
set nwpw:cif_filename diamond444.opt
driver; clear; maxiter 40; end; task band optimize ignore

#5x5x5 k-point mesh
nwpw
  monkhorst-pack 5 5 5
end
set nwpw:cif_filename diamond555.opt
driver; clear; maxiter 40; end; task band optimize ignore
```

The following figure shows a plot of the cohesive energy and C-C bond distance versus the Brillouin zone sampling. As can be seen in this figure the cohesive energy (w/o zero-point correction) and C-C bond distance agree very well with the experimental values of 7.37 eV (including zero-point correction) and 1.54 Angs.



12.1.26.3 Using BAND to Optimize the Unit Cell for a 2 Atom Primitive Cell of Diamond

(input:Media:diamond-fcc.nw, output:Media:diamond-fcc.nwout.gz)

In this example the BAND module is used to optimize a 2 atom unit cell for a diamond crystal at different Brillouin zone samplings. The optimized energy and geometry will be (Monkhorst-Pack sampling of 11x11x11)

```
 ----------------------
 Optimization converged
 ----------------------

  Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
  ---- ---------------- -------- -------- -------- -------- -------- --------
@    1   -11.40586236  5.2D-07  0.00039  0.00018  0.00002  0.00003    662.0
                            ok       ok       ok       ok


                  Geometry "geometry" -> "geometry"
                  ---------------------------------

 Output coordinates in angstroms (scale by  1.889725989 to convert to a.u.)

  No.       Tag         Charge          X              Y              Z
 ---- ---------------- ---------- -------------- -------------- --------------
   1 C                  6.0000     0.00000000     0.00000000     0.00000000
   2 C                  6.0000     0.72201500     1.25056532     0.51054180

     Lattice Parameters
     ------------------

     lattice vectors in angstroms (scale by  1.889725989 to convert to a.u.)

     a1=<   2.165   1.251   0.001 >
     a2=<   0.001   2.500   0.001 >
     a3=<   0.722   1.251   2.041 >
     a=       2.500 b=       2.500 c=       2.500
     alpha=  59.966 beta=  59.966 gamma=  59.966
     omega=    11.0

     reciprocal lattice vectors in a.u.

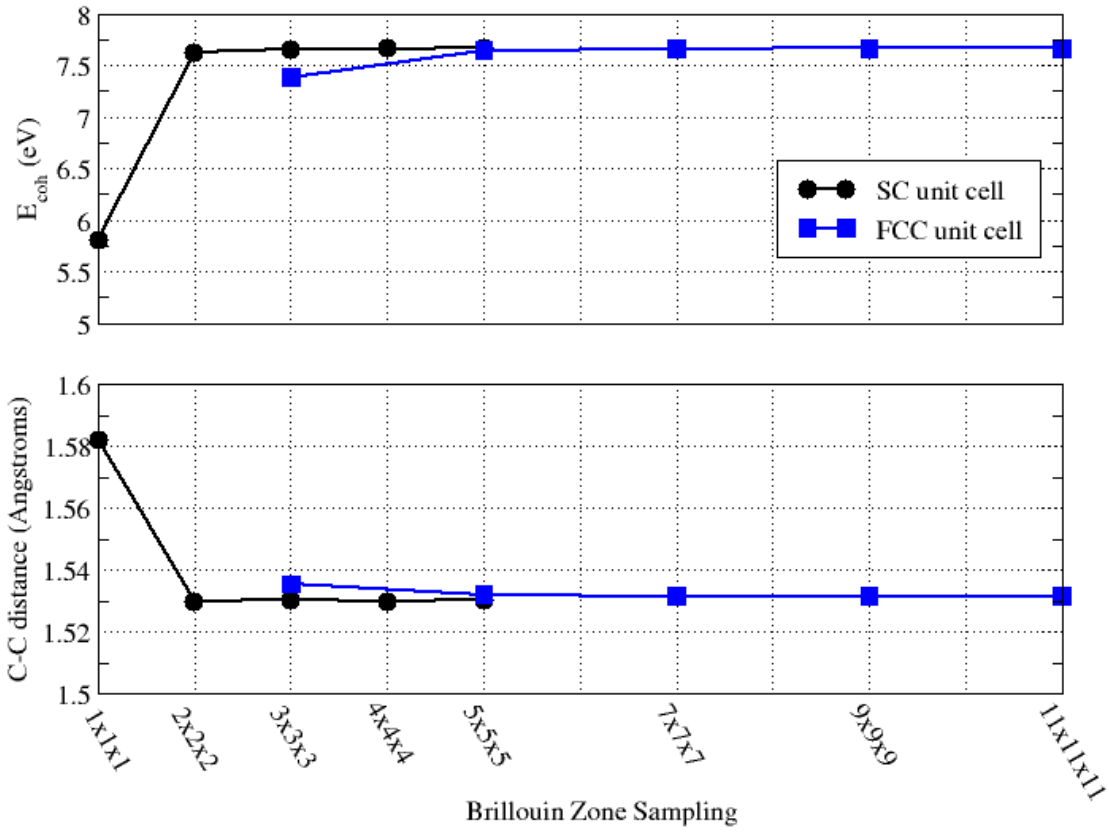
     b1=<   1.536  -0.768   0.000 >
     b2=<   0.000   1.330   0.000 >
     b3=<  -0.543  -0.543   1.629 >

     Atomic Mass
     -----------

     C                12.000000


 ================================================================================
                          internuclear distances
 --------------------------------------------------------------------------------
      center one      |      center two      | atomic units |  angstroms
 --------------------------------------------------------------------------------
   2 C                |   1 C                |     2.89435  |     1.53162
 --------------------------------------------------------------------------------
                     number of included internuclear distances:          1
 ================================================================================
```

The following figure shows a plot of the cohesive energy and C-C bond distance versus the Brillouin zone sampling for the 8 atom SC unit cell and the 2 atom FCC unit cell.



12.1.26.4 Using BAND to Calculate the Band Structures of Diamond

(input:Media:diamond-structure.nw, output:Media:diamond-structure.nwout, file:diamondfcc.restricted_band.dat)

The following example uses the BAND module to calculate the band structure for the FCC cell of the a diamond crystal. The fractional coordinates and the unit cell are defined in the geometry block. The simulation_cell block is not needed since NWPW automatically uses the unit cell defined in the geometry block.

```
title "Diamond 2 atom fcc cell Brillouin sampling=9x9x9 M-P - Band structure plot"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamondfcc

memory 1950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 2.500d0
    lat_b 2.500d0
    lat_c 2.500d0
    alpha 60.0d0
    beta  60.0d0
    gamma 60.0d0
  end
 C  0.00000d0  0.00000d0  0.00000d0
 C  0.25000d0  0.25000d0  0.25000d0
end

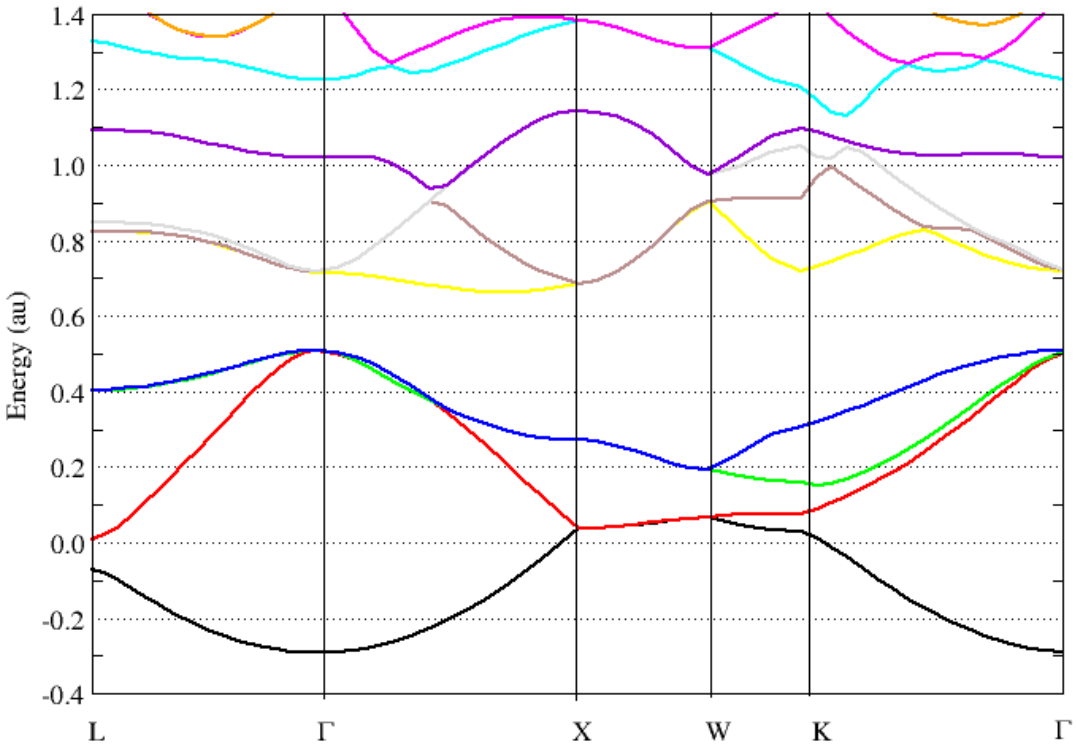
nwpw
  ewald_rcut 3.0
  ewald_ncut 8    #The default value of 1 needs to be increased
  lmbfgs
  xc pbe96

  monkhorst-pack 9 9 9
end

#need to run "task band energy" before "task band structure" can be run
task band energy

nwpw
   virtual 16
   brillouin_zone
     zone_name fccpath
     path fcc l gamma x w k gamma
   end
   zone_structure_name fccpath
end
task band structure
```

This calculation outputs the file:diamondfcc.restricted_band.dat) data file in the permanent_directory. A plotting (e.g. gnuplot or xmgrace) can be used to display the band structure.



12.1.26.5 Using BAND to Calculate the Density of States of Diamond

(2 atom cell - input:diamond-dos.nw output:diamond-dos.nwout, diamond-dos.dos.dat (8 atom cell - input:diamond-dos8.nw output: diamond-dos8.nwout.gz, diamond-dos8.dos.dat

There are two possible ways to use the BAND module to calculate the density and projected density of states. The first approach just uses the eigenvalues generated from an energy calculation to generate a density of states. The following example uses this strategy to calculate the density of states and projected density of states of diamond.

```
title "Diamond 2 atom fcc cell Brillouin sampling=9x9x9 M-P - density of states plot"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamond-dos

memory 1950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 2.500d0
    lat_b 2.500d0
    lat_c 2.500d0
    alpha 60.0d0
    beta  60.0d0
    gamma 60.0d0
  end
 C  0.00000d0  0.00000d0  0.00000d0
 C  0.25000d0  0.25000d0  0.25000d0
end

nwpw
  ewald_rcut 3.0
  ewald_ncut 8    #The default value of 1 needs to be increased
  lmbfgs
  xc pbe96

  monkhorst-pack 9 9 9
  dos                 # dos keyword tells the code to calculate dos at the end of an energy calculation
  mulliken              # turn on projected density of states
  virtual 8             # include 8 virtual states
end

task band energy
```

The other approach uses the band structure code to calculate the eigenvalues given a precomputed density. The approach is slower than the first approach, however, it can be used to substantially increase the number of k-points and virtual orbitals used to generate the density of states. The following example demonstrates this capability to calculate the density of states and projected density of states of the diamond crystal.

```
title "Diamond 2 atom fcc cell Brillouin sampling=9x9x9 M-P - density of states plot"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamond-dos

memory 1950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 2.500d0
    lat_b 2.500d0
    lat_c 2.500d0
    alpha 60.0d0
    beta  60.0d0
    gamma 60.0d0
  end
 C  0.00000d0  0.00000d0  0.00000d0
 C  0.25000d0  0.25000d0  0.25000d0
end

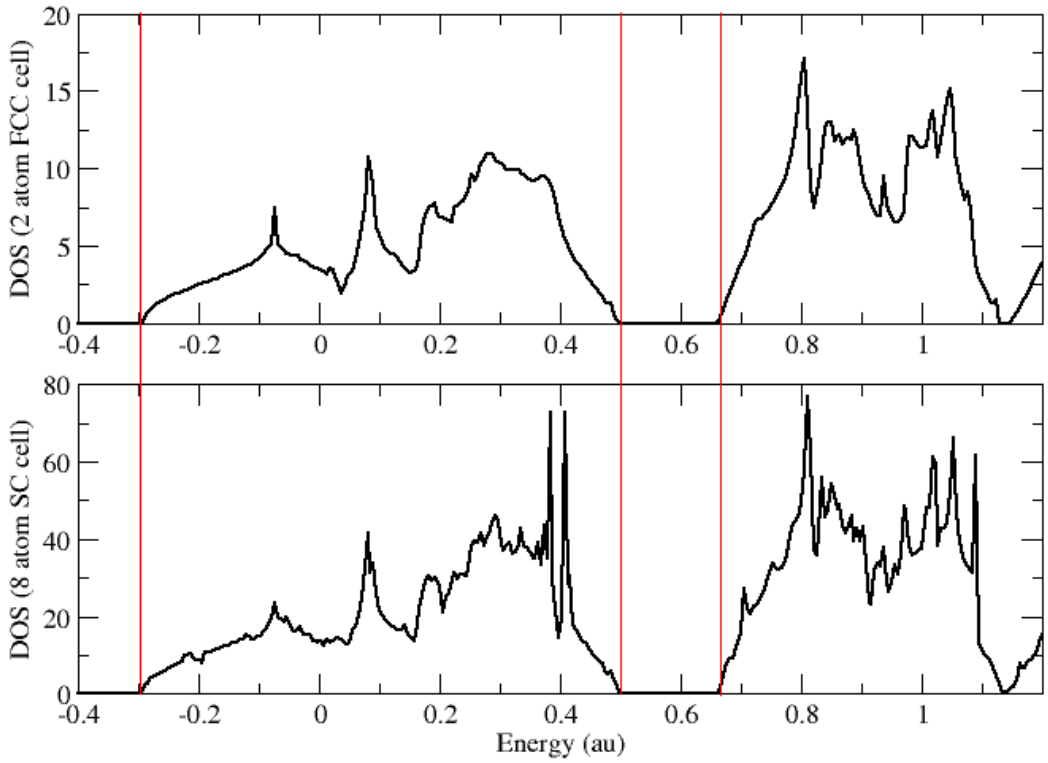
nwpw
  ewald_rcut 3.0
  ewald_ncut 8    #The default value of 1 needs to be increased
  lmbfgs
  xc pbe96

  monkhorst-pack 9 9 9
end

#need to run "task band energy" before "task band dos" can be run
task band energy

nwpw
   virtual 26                   #26 virtual orbitals included in the DOS calculation
   dos 0.002 700 -1.00000 2.0000  #alpha npoints emin emax,....,change default energy range and gridding. note alpha not used in task band dos calculations
   dos-grid 11 11 11
   mulliken                     # mulliken keyword used to turn on projected density of states
end
task band dos
```

This calculation outputs the data file in the permanent_directory. A plotting (e.g. gnuplot or xmgrace) can be used to display the density of states.



12.1.26.6 Calculate the Phonon Spectrum of Diamond

```
title "Diamond 2 atom fcc cell Brillouin sampling=9x9x9 M-P - Phonon spectra"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamond-dos

memory 1950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 2.500d0
    lat_b 2.500d0
    lat_c 2.500d0
    alpha 60.0d0
    beta  60.0d0
    gamma 60.0d0
  end
 C  0.00000d0  0.00000d0  0.00000d0
 C  0.25000d0  0.25000d0  0.25000d0
end

nwpw
  ewald_rcut 3.0
  ewald_ncut 8    #The default value of 1 needs to be increased
  lmbfgs
  xc pbe96

  monkhorst-pack 9 9 9
end

task band energy
task band freq
```

12.1.27 NWPW Tutorial 6: optimizing the unit cell of nickel with fractional

occupation

(input:Media:Ni-band.nw output:Media:Ni-band.nwout) The following example demonstrates how to uses the BAND module to optimize the unit cell and geometry for FCC cell of Nickel metal

```
title "Ni FCC metal, monkhorst-pack=3x3x3, 5x5x5, and 7x7x7, fermi smearing, xc=pbe96"
echo

start Ni-band

memory 1900 mb

permanent_dir ./perm
scratch_dir   ./scratch

geometry units angstroms center noautosym noautoz print
 system crystal
   lat_a 3.5451d0
   lat_b 3.5451d0
   lat_c 3.5454d0
   alpha 90.0d0
   beta  90.0d0
   gamma 90.0d0
  end

Ni 0.000000   0.000000   0.000000
Ni 0.000000   0.500000   0.500000
Ni 0.500000   0.000000   0.500000
Ni 0.500000   0.500000   0.000000
end
set nwpw:cif_filename Ni-band
set nwpw:zero_forces .true.
set includestress    .true.

#turn on pseudopotential filtering
set nwpw:kbpp_ray .true.
set nwpw:kbpp_filter .true.

nwpw
   #fractional occupation
   smear fermi

   #scf option used with smear
   scf anderson outer_iterations 0 kerker 2.0

   ewald_ncut 8
   ewald_rcut 3.0
   xc pbe96
   monkhorst-pack 3 3 3
   np_dimensions -1 -1 4
end

#generate initial wavefunctions w/ low cutoff energy
nwpw
   loop 10 10
   cutoff 10.0
end
task band energy

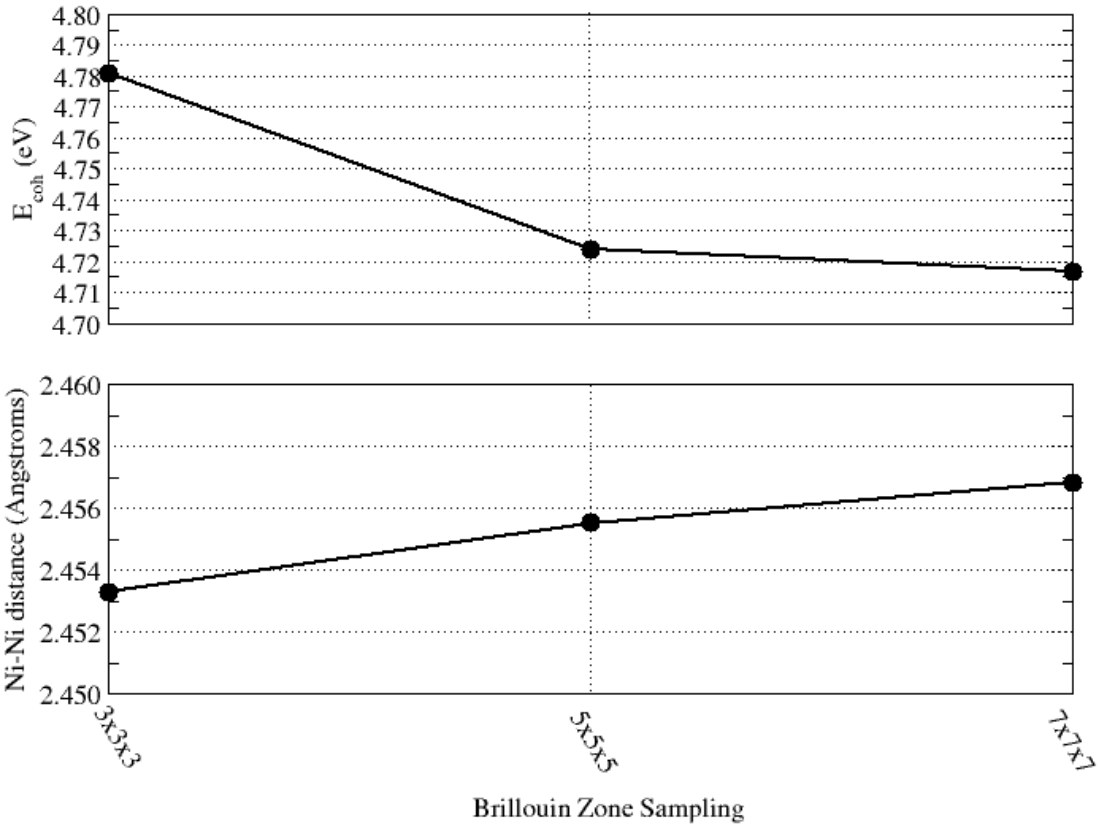
#increase cutoff energy and number of iterations
nwpw
   cutoff 50.0
   loop 10 100
end

#3x3x3 k-point mesh
nwpw
  monkhorst-pack 3 3 3
end
set nwpw:cif_filename nickel333.opt
driver; clear; maxiter 40; end; task band optimize ignore

#5x5x5 k-point mesh
nwpw
  monkhorst-pack 5 5 5
end
set nwpw:cif_filename nickel555.opt
driver; clear; maxiter 40; end; task band optimize ignore

#7x7x7 k-point mesh
nwpw
  monkhorst-pack 7 7 7
end
set nwpw:cif_filename nickel777.opt
driver; clear; maxiter 40; end; task band optimize ignore
```

The following figure shows a plot of the cohesive energy and Ni-Ni bond distance versus the Brillouin zone sampling. As can be seen in this figure the cohesive energy (w/o zero-point correction) and Ni-Ni bond distance agree very well with the experimental values of 4.44 eV (including zero-point correction) and 2.49 Angs.



12.1.28 NWPW Tutorial 7: Optimizing the unit cells with symmetry: Diamond with Fd-3m symmetry and Brucite with P-3m1 symmetry

(Diamond example, input:Media:diamond-symmetry.nw, output:Media:diamond-symmetry.nwout)

(Brucite example, input:Media:brucite-symmetry.nw, output:Media:brucite-symmetry.nwout)

The following example uses the BAND module to optimize the unit cell and geometry for a Diamond crystal with Fd-3m symmetry. The fractional coordinates, unit cell, and symmetry are defined in the geometry block.

```
title "Diamond 8 atom cubic cell generated using Fd-3m symmetry - geometry and unit cell optimization"
echo

memory 1500 mb

permanent_dir ./perm
scratch_dir   ./scratch

start diamond-symmetry


geometry nocenter noautosym noautoz print
 system crystal
   lat_a 3.58
   lat_b 3.58
   lat_c 3.58
   alpha 90.0
   beta  90.0
   gamma 90.0
 end
symmetry Fd-3m
C 0.0 0.0 0.0
end
set nwpw:cif_filename diamond-symmetry

#turn on pseudopotential filtering
set nwpw:kbpp_ray    .true.
set nwpw:kbpp_filter .true.

#***** setup the nwpw Band code - 3x3x3 k-point mesh ****
nwpw
  ewald_rcut 3.0
  ewald_ncut 8
  xc pbe96
  lmbfgs
  monkhorst-pack 3 3 3
  np_dimensions -1 -1 4
end

set includestress  .true.   # tell driver to optimize unit cell
set includelattice .true.   # tell driver to optimize with a,b,c,alpha,beta,gamma
task band optimize ignore
```

The optimized geometry will also contain the information about the symmetry being used

```
....
     ----------------------
     Optimization converged
     ----------------------


 Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
 ---- ---------------- -------- -------- -------- -------- -------- --------
@    7    -45.62102901 -4.1D-07  0.00010  0.00003  0.00019  0.00060    287.1
                                  ok       ok       ok       ok


                    Geometry "geometry" -> "geometry"
                    ---------------------------------

 Output coordinates in angstroms (scale by  1.889725989 to convert to a.u.)

 No.       Tag         Charge          X              Y              Z
---- ---------------- ---------- -------------- -------------- --------------
   1 C                6.0000     0.00000000     0.00000000     0.00000000
   2 C                6.0000     0.00000000     1.76715074     1.76715074
   3 C                6.0000     1.76715074     1.76715074     0.00000000
   4 C                6.0000     1.76715074     0.00000000     1.76715074
   5 C                6.0000     2.65072611     0.88357537     2.65072611
   6 C                6.0000     0.88357537     0.88357537     0.88357537
   7 C                6.0000     0.88357537     2.65072611     2.65072611
   8 C                6.0000     2.65072611     2.65072611     0.88357537

     Lattice Parameters
     ------------------

     lattice vectors in angstroms (scale by  1.889725989 to convert to a.u.)

     a1=<   3.534   0.000   0.000 >
     a2=<   0.000   3.534   0.000 >
     a3=<   0.000   0.000   3.534 >
     a=       3.534 b=       3.534 c=       3.534
     alpha=  90.000 beta=  90.000 gamma=  90.000
     omega=    44.1
```

reciprocal lattice vectors in a.u.

b1=< 0.941 0.000 0.000 >
b2=< 0.000 0.941 0.000 >
b3=< 0.000 0.000 0.941 >

Atomic Mass

C 12.000000

Symmetry information

Group name Fd-3m
Group number 227
Group order 192
No. of unique centers 1
Setting number 1

Symmetry unique atoms

1

internuclear distances

center one	center two	atomic units	angstroms
5 C	4 C	2.89203	1.53040
6 C	1 C	2.89203	1.53040
6 C	2 C	2.89203	1.53040
6 C	3 C	2.89203	1.53040
6 C	4 C	2.89203	1.53040
7 C	2 C	2.89203	1.53040
8 C	3 C	2.89203	1.53040

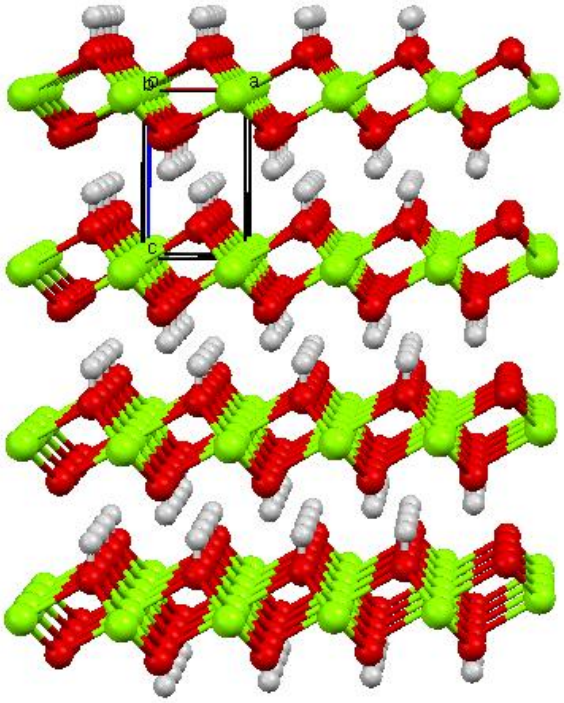
number of included internuclear distances: 7

internuclear angles

center 1	center 2	center 3	degrees
6 C	2 C	7 C	109.47
6 C	3 C	8 C	109.47
5 C	4 C	6 C	109.47
1 C	6 C	2 C	109.47
1 C	6 C	3 C	109.47
1 C	6 C	4 C	109.47
2 C	6 C	3 C	109.47
2 C	6 C	4 C	109.47
3 C	6 C	4 C	109.47

number of included internuclear angles: 9

The following example uses the BAND module to optimize the unit cell and geometry for a Brucite crystal ($Mg(OH)_2$ with P-3m1 symmetry.




```
title "brucite testing - using P-3m1 symmetry"
echo

memory 1500 mb

permanent_dir ./perm
scratch_dir      ./scratch

geometry nocenter noautosym noautoz print
 system crystal
   lat_a 3.14979
   lat_b 3.14979
   lat_c 4.7702
   alpha 90.0
   beta  90.0
   gamma 120.0
 end
symmetry P-3m1
Mg   0.00000   0.00000  0.00000
O   0.33333   0.66667   0.22030
H   0.33333   0.66667   0.41300
end
set nwpw:cif_filename brucite

#turn on pseudopotential filtering
set nwpw:kbpp_ray .true.
set nwpw:kbpp_filter .true.

#***** setup the nwpw gamma point code ****
nwpw
  ewald_rcut 3.0
  ewald_ncut 8
  xc pbe96
  lmbfgs
  monkhorst-pack 3 3 2
  #np_dimensions -1 -1 4
end

driver
  clear
  maxiter 31
end

set includestress  .true.          # tell driver to optimize unit cell
set includelattice .true.

task band optimize ignore
```

Optimizing Brucite, which is a soft layered material (2.5-3 Mohs scale), is more difficult to optimize than a hard material such as Diamond. For these types of materials using symmetry can often result in a faster optimization. For example, with symmetry the optimization converges within 20 to 30 geometry optimization steps,

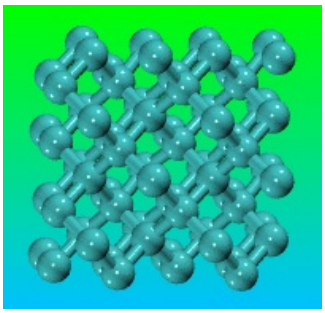
```
@ Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
@ ---- ---------------- -------- -------- -------- -------- -------- --------
@    0    -34.39207476  0.0D+00  0.24673  0.10223  0.00000  0.00000    172.7
@    1    -34.39340208 -1.3D-03  0.00872  0.00302  0.00198  0.00485    328.5
....
@   20    -34.39042736 -1.2D-05  0.00195  0.00083  0.00440  0.01964   3019.2
@   21    -34.39043463 -7.3D-06  0.00028  0.00011  0.00493  0.02042   3150.6
@   22    -34.39043484 -2.1D-07  0.00043  0.00014  0.00002  0.00008   3278.5
@   22    -34.39043484 -2.1D-07  0.00043  0.00014  0.00002  0.00008   3278.5
```

whereas, without symmetry the optimization may not be converged even at 100 geometry steps (input:Media:brucite-nosymmetry.nw, output:Media:brucite-nosymmetry.nwout).

```
@ Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
@ ---- ---------------- -------- -------- -------- -------- -------- --------
@    0    -34.39207476  0.0D+00  0.24673  0.10250  0.00000  0.00000    18.4
@    1    -34.39340765 -1.3D-03  0.02963  0.00715  0.00202  0.00500    30.7
...
@   49    -34.39027641 -2.1D-06  0.01870  0.00646  0.00074  0.00202   595.7
@   50    -34.39027503  1.4D-06  0.01962  0.00669  0.00069  0.00197   608.4
...
@  100    -34.39034236 -3.8D-07  0.00380  0.00150  0.00036  0.00132  1155.3
@  101    -34.39034431 -1.9D-06  0.00305  0.00118  0.00012  0.00045  1166.8
@  102    -34.39034449 -1.8D-07  0.00370  0.00144  0.00006  0.00020  1177.9
...
```

12.1.29 NWPW Tutorial 8: NVT Metropolis Monte-Carlo Simulations

In this example the PSPW module is used to run an NVT simulation for a diamond crystal using the a Metropolis Monte-Carlo algorithm.



```
title "Metropolis NVT simulation of diamond - this input is used to put the system in equilibrium"
echo

start diamond-nvt

#permanent_dir ./perm
#scratch_dir   ./perm

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
system crystal
  lat_a 3.56d0
  lat_b 3.56d0
  lat_c 3.56d0
  alpha 90.0d0
  beta  90.0d0
  gamma 90.0d0
end
C -0.50000d0 -0.50000d0 -0.50000d0
C  0.00000d0  0.00000d0 -0.50000d0
C  0.00000d0 -0.50000d0  0.00000d0
C -0.50000d0  0.00000d0  0.00000d0
C -0.25000d0 -0.25000d0 -0.25000d0
C  0.25000d0  0.25000d0 -0.25000d0
C  0.25000d0 -0.25000d0  0.25000d0
C -0.25000d0  0.25000d0  0.25000d0
end
set nwpw:cif_filename diamond_nvt_234

###### setup the nwpw gamma point code ######
set nwpw:kbpp_ray .true.
set nwpw:kbpp_filter .true.
set nwpw:frozen_lattice:thresh 999.0
nwpw
  lmbfgs
  ewald_rcut 3.0
  ewald_ncut 8
  xc pbe
end
task pspw energy

##### optimize the unit cell #####
set includestress  .true.  #this option tells driver to optimize the unit cell
set includelattice .true.  #this option tells driver to optimize cell using a,b,c,alpha,beta,gamma
driver
  clear
  maxiter 51
end
task pspw optimize ignore

###############################################################################################
###### setup Metropolis NVT code - input will change in a forthcoming release ######
###############################################################################################
set nwpw:mc_seed 234              # Seed for random number generator
set nwpw:mc_algorithm 1           # 1-NVT; 2-NPT
set nwpw:mc_aratio 0.234          # targeted acceptance ratio
set nwpw:mc_ddx 0.1               # parameter used to adjust geometry dispacement to have sampling with targeted acceptance
set nwpw:mc_temperature 300.0     # Temperature in K
set nwpw:mc_step_size   0.250     # initial geometry displacement step size

nwpw
  mc_steps 10 100  #total number of iterations = 10*100, number of iterations between step size adjustments = 10
  cpmd_properties on
end
task pspw Metropolis
```

12.1.30 NWPW Tutorial 9: NPT Metropolis Monte-Carlo Simulations

In this example the PSPW module is used to run an NPT simulation for a diamond crystal using the a Metropolis Monte-Carlo algorithm.

(input:Media:diamond-metropolis.nw, output:Media:diamond-metropolis.nwout.gz, datafiles:Media:diamond-metropolis.emotion.gz, Media:diamond-metropolis.ion_motion.gz, Media:diamond-metropolis.xyz.gz, Media:diamond_metropolis_1234.cif.gz)

```
title "Metropolis NPT simulation of diamond - this input is used to put the system in equilibrium"
echo

start diamond-metropolis

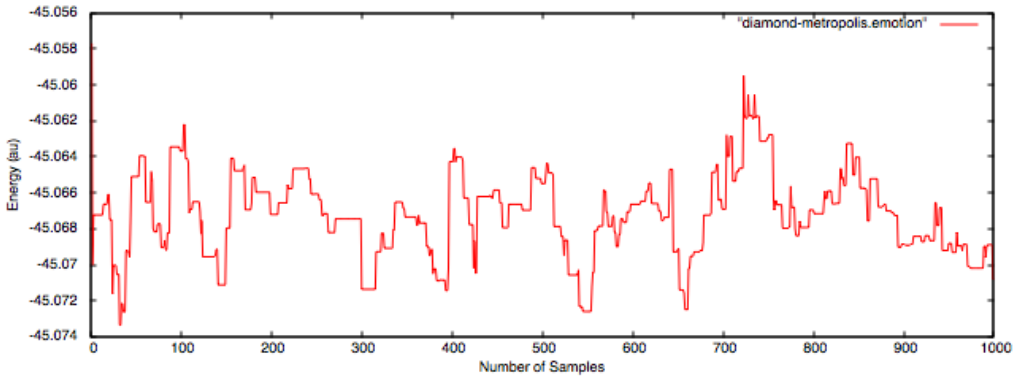
#permanent_dir ./perm
#scratch_dir   ./perm

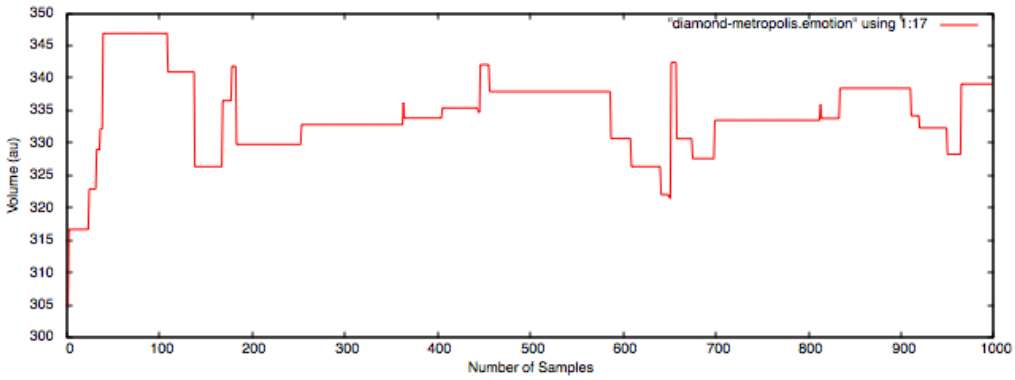
#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 3.56d0
    lat_b 3.56d0
    lat_c 3.56d0
    alpha 90.0d0
    beta  90.0d0
    gamma 90.0d0
  end
 C -0.50000d0 -0.50000d0 -0.50000d0
 C  0.00000d0  0.00000d0 -0.50000d0
 C  0.00000d0 -0.50000d0  0.00000d0
 C -0.50000d0  0.00000d0  0.00000d0
 C -0.25000d0 -0.25000d0 -0.25000d0
 C  0.25000d0  0.25000d0 -0.25000d0
 C  0.25000d0 -0.25000d0  0.25000d0
 C -0.25000d0  0.25000d0  0.25000d0
end
set nwpw:cif_filename pspw_metropolis

###### setup the nwpw gamma point code ######
set nwpw:kbpp_ray .true.
set nwpw:kbpp_filter .true.
set nwpw:frozen_lattice:thresh 999.0
nwpw
   lmbfgs
   ewald_rcut 3.0
   ewald_ncut 8
   xc pbe
end
task pspw energy


################################################################################
###### setup Metropolis NPT code - input will change in a forthcoming release ######
################################################################################
set nwpw:mc_seed 1234              # Seed for random number generator
set nwpw:mc_algorithm 2            # 1-NVT; 2-NPT
set nwpw:mc_aratio 0.234           # targeted acceptance ratio
set nwpw:mc_ddx 0.1                # parameter used to adjust geometry dispacement to have sampling with targeted acceptance
set nwpw:mc_ddv 0.1                # parameter used to adjust volume change to have sampling with targeted acceptance
set nwpw:mc_temperature 300.0      # Temperature in K
set nwpw:mc_step_size   0.250      # geometry displacement step size
set nwpw:mc_volume_step 0.130      # volume displacement step size

nwpw
   bo_steps 10 100  #total number of iterations = 10*100, number of iterations between step size adjustments = 10
end
task pspw Metropolis
```





(inputs:Media:diamond-metropolis-sampling.nw.tgz)

(python analysis program:Media:makehistogram.gz)

```
[WE27972:~/Projects/NWChem/Metropolis] bylaska% makehistogram -t 300 -c 2 1235/diamond-metropolis-1235.emotion 1236/diamond-metropolis-1236.emotion
1237/diamond-metropolis-1237.emotion 1238/diamond-metropolis-1238.emotion 1239/diamond-metropolis-1239.emotion 1240/diamond-metropolis-1240.emotion
1241/diamond-metropolis-1241.emotion 1242/diamond-metropolis-1242.emotion 1243/diamond-metropolis-1243.emotion 1244/diamond-metropolis-1244.emotion
1245/diamond-metropolis-1245.emotion 1246/diamond-metropolis-1246.emotion 1248/diamond-metropolis-1248.emotion 1249/diamond-metropolis-1249.emotion
makehistogram Program
len(args)= 14

unitconversion  = 1.0
temperature (K) = 300
RT (au)         = 0.000949482834326  ( 0.5958  kcal/mol)

data columns -1 = [1]
data rows (n)   = 52000

delta (au)      = 0.01
xmin-delta (au)  = -45.08080365
xmax+delta (au)  = -45.05079515

data averaging:
- xbar       (au)          = -45.0668093497
- S2_{n-1}   (au)          = 1.08378343101e-05
- <exp((x-xmin)/RT)> (au)   = 5293374903.39
- <exp((x-xbar)/RT)> (au)   = 2102.44405413
- Free energy               = -45.0595449934
- Free energy1              = -45.0595449934

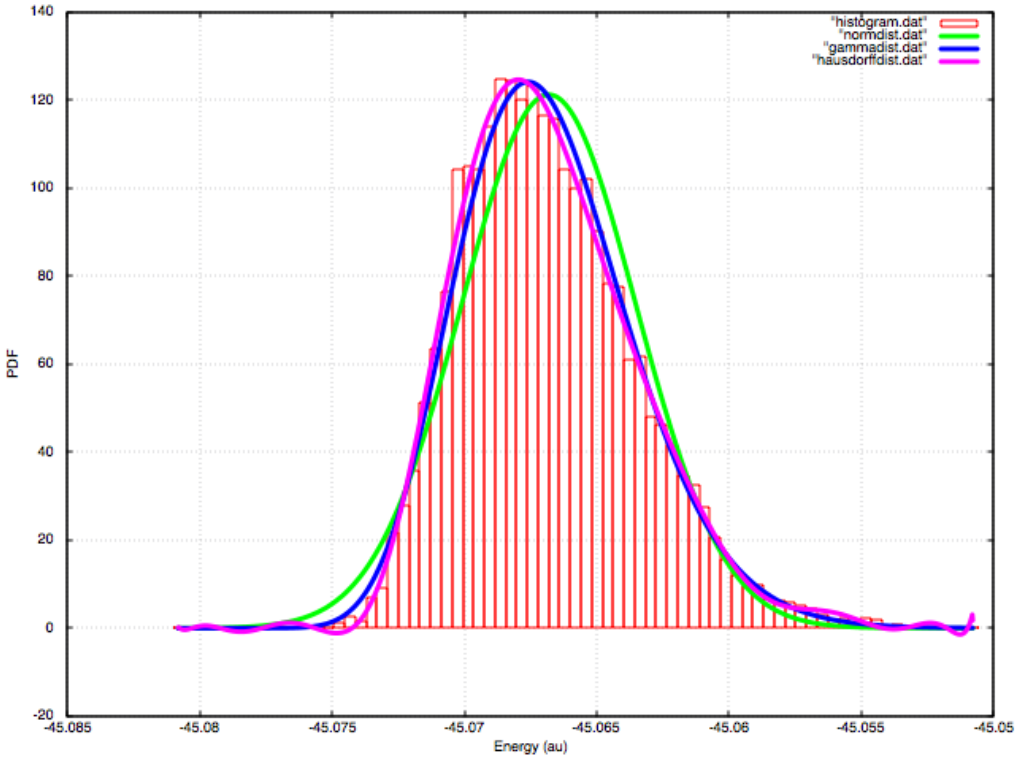
histogram distribution parameters:
- number of bins (Rice k) = 75
- bin width               = 0.00040552027027
- norm                    = 1.0
- xbar (au)               = -45.0668107987  (error= -1.44908364064e-06 )
- S2_{n-1} (au)           = 1.0858459744e-05  (error= 2.06254339582e-08 )
- <exp((x-xmin)/RT)> (au) = 5184600342.01  (error= -108774561.378 )
- <exp((x-xbar)/RT)> (au) = 2062.38570923  (error= -40.0583449011 )
- Free energy             = -45.0595647078  (error= -1.9714360235e-05 )
- Free energy1            = -45.0595647078  (error= -1.9714360235e-05 )
- histogram plot file     = histogram.dat

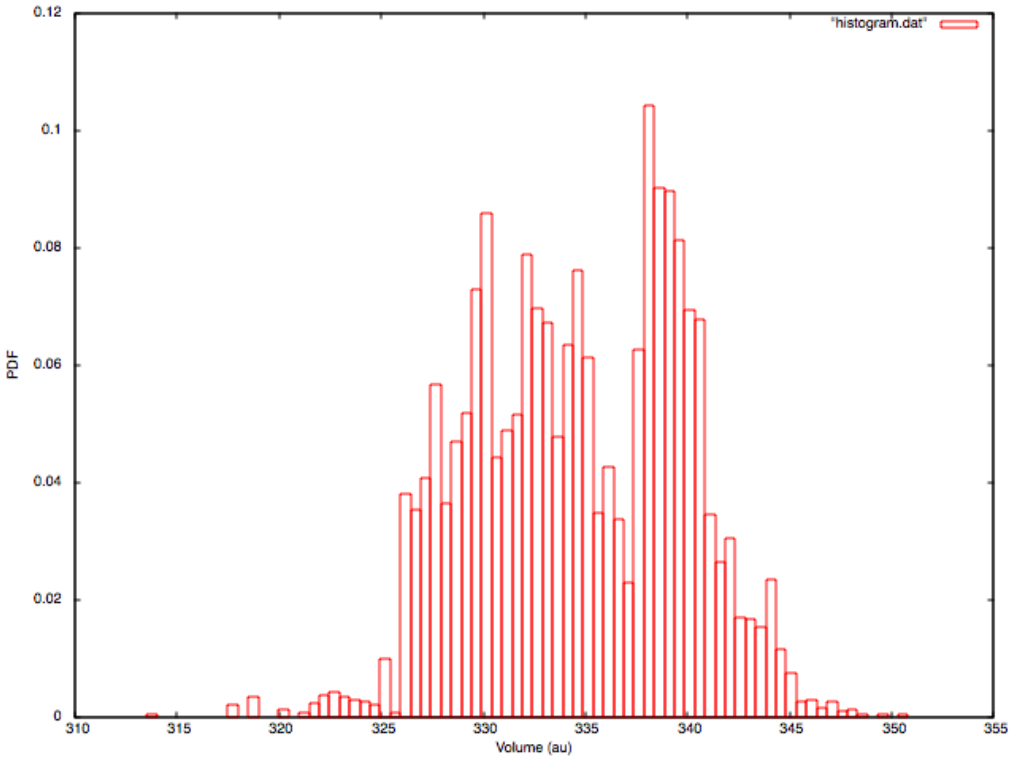
normal distribution parameters:
- average x (input xbar)                = -45.0668093497
- unbiased sample variance (input S2_(n-1))= 1.08378343101e-05
- xbar-xmin                             = 0.0139943003357
- norm                    = 0.99998877936
- xbar (au)               = -45.0663035243  (error= 0.000505825397738 )
- S2_{n-1} (au)           = 1.1091077321e-05  (error= 2.53243010936e-07 )
- <exp((x-xmin)/RT)> (au) = 943482808.939  (error= -4349892094.45 )
- <exp((x-xbar)/RT)> (au) = 219.968603653  (error= -1882.47545048 )
- Free energy             = -45.061182503  (error= -0.00163750957643 )
- Free energy1            = -45.061182503  (error= -0.00163750957643 )
```

- normal distribution plot file = normdist.dat
- number data points = 1500

gamma distribution parameters:
- alpha0= 18.0700715921
- beta0 = 1291.24508969
- xmin + alpha0/beta0 = -45.0668093497
- alpha = 18.5003178357
- beta = 1321.98948086
- xmin + alpha/beta = -45.0668093497
- norm = 0.999923464137 0.99993569948
- xbar (au) = -45.0633614482 -45.0639126423 (error= 0.00344790150088 0.00289670733491)
- S2_{n-1} (au) = 2.27110055327e-05 1.89632753897e-05 (error= 1.18731712226e-05 8.12544107961e-06)
- <exp((x-xmin)/RT)> (au) = 7932775654.26 7060892836.07 (error= 2639400750.87 1767517932.68)
- <exp((x-xbar)/RT)> (au) = 83.43400035 132.707151194 (error= -2019.01005378 -1969.73690294)
- Free energy = -45.059160883 -45.0592714327 (error= 0.000384110406969 0.000273560709338)
- Free energy1 = -45.059160883 -45.0592714327 (error= 0.000384110406969 0.000273560709338)
- gamma distribution plot file = gammadist.dat
- number data points = 1500

Hausdorff distribution parameters:
- xmin = -45.08080365
- xmax = -45.05079515
- number moments = 15
  -- < x^0 > = 1.000000000000000
  -- < x^1 > = 0.466344546904007
  -- < x^2 > = 0.229512222180349
  -- < x^3 > = 0.119040323347820
  -- < x^4 > = 0.064946164109284
  -- < x^5 > = 0.037186896798964
  -- < x^6 > = 0.022287980659815
  -- < x^7 > = 0.013942929105868
  -- < x^8 > = 0.009076370636747
  -- < x^9 > = 0.006128509645342
  -- < x^10 > = 0.004278147917961
  -- < x^11 > = 0.003077410986590
  -- < x^12 > = 0.002273768533280
  -- < x^13 > = 0.001720304299285
  -- < x^14 > = 0.001328990330385
- norm = 1.0000000003
- xbar (au) = -45.066809363 (error= -1.33426993898e-08)
- S2_{n-1} (au) = 1.08376258908e-05 (error= -2.08419282206e-10)
- <exp((x-xmin)/RT)> (au) = 5423305875.35 (error= 129930971.958)
- <exp((x-xbar)/RT)> (au) = 2154.08083332 (error= 51.6367791881)
- Free energy = -45.0595219689 (error= 2.30245307122e-05)
- Free energy1 = -45.0595219689 (error= 2.30245307122e-05)
- Hausdorff moment history file = moment_hist.dat
- Hausdorff distribution plot file = hausdorff.dat
- number data points = 1500





12.1.31 NWPW Tutorial 9: Free Energy Simulations

A description of using the WHAM method for generating free energy of the gas-phase dissociation reaction CH\(_3\)Cl \(\rightarrow\) CH\(_3\)+Cl can be found in the attached pdf (nwchem-new-pmf.pdf)

12.1.32 PAW Tutorial

12.1.32.1 Optimizing a water molecule

The following input deck performs for a water molecule a PSPW energy calculation followed by a PAW energy calculation and a PAW geometry optimization calculation. The default unit cell parameters are used (SC=20.0, ngrid 32 32 32). In this simulation, the first PAW run optimizes the wavefunction and the second PAW run optimizes the wavefunction and geometry in tandem.

```
title "paw steepest descent test"
start paw_test
charge 0
geometry units au nocenter noautoz noautosym
O      0.00000    0.00000    0.01390
H     -1.49490    0.00000   -1.18710
H      1.49490    0.00000   -1.18710
end
nwpw
  time_step 15.8
  ewald_rcut 1.50
  tolerances 1.0d-8 1.0d-8
end
set nwpw:lcao_iterations 1
set nwpw:minimizer 2
task pspw energy
task paw energy
nwpw
  time_step 5.8
  geometry_optimize
  ewald_rcut 1.50
  tolerances 1.0d-7 1.0d-7 1.0d-4
end
task paw steepest_descent
task paw optimize
```

12.1.32.2 Optimizing a unit cell and geometry for Silicon-Carbide

The following example demonstrates how to uses the PAW module to optimize the unit cell and geometry for a silicon-carbide crystal.

```
title "SiC 8 atom cubic cell - geometry and unit cell optimization"
start SiC
#**** Enter the geometry using fractional coordinates ****
geometry units au center noautosym noautoz print
 system crystal
   lat_a 8.277d0
   lat_b 8.277d0
   lat_c 8.277d0
   alpha 90.0d0
   beta  90.0d0
   gamma 90.0d0
 end
Si    -0.50000d0  -0.50000d0  -0.50000d0
Si     0.00000d0   0.00000d0  -0.50000d0
Si     0.00000d0  -0.50000d0   0.00000d0
Si    -0.50000d0   0.00000d0   0.00000d0
C     -0.25000d0  -0.25000d0  -0.25000d0
C      0.25000d0   0.25000d0  -0.25000d0
C      0.25000d0  -0.25000d0   0.25000d0
C     -0.25000d0   0.25000d0   0.25000d0
end
#***** setup the nwpw gamma point code ****
nwpw
  simulation_cell
    ngrid 16 16 16
  end
  ewald_ncut 8
end
set nwpw:minimizer 2
set nwpw:psi_nolattice .true.  # turns of unit cell checking for wavefunctions
driver
  clear
  maxiter 40
end
set includestress .true.         # this option tells driver to optimize the unit cell
set nwpw:stress_numerical .true. #currently only numerical stresses implemented in paw
task paw optimize
```

12.1.32.3 Running a Car-Parrinello Simulation

In this section we show how use the PAW module to perform a Car-Parrinello molecular dynamic simulation for a C_2 molecule at the LDA level. Before running a PAW Car-Parrinello simulation the system should be on the Born-Oppenheimer surface, i.e. the one-electron orbitals should be minimized with respect to the total energy (i.e. task pspw energy). The input needed is basically the same as for optimizing the geometry of a C_2 molecule at the LDA level,except that and additional Car-Parrinello sub-block is added.

In the following example we show the input needed to run a Car-Parrinello simulation for a C_2 molecule at the LDA level. In this example, default pseudopotentials from the pseudopotential library are used for C, the boundary condition is free-space, the exchange correlation functional is LDA, The boundary condition is free-space, and the simulation cell cell is aperiodic and cubic with a side length of 10.0 Angstroms and has 40 grid points in each direction (cutoff energy is 44 Ry). The time step and fake mass for the Car-Parrinello run are specified to be 5.0 au and 600.0 au, respectively.

```
start c2_paw_lda_md
title "C2 restricted singlet dimer, LDA/44Ry - constant energy Car-Parrinello simulation"
geometry
 C    -0.62 0.0 0.0
 C     0.62 0.0 0.0
end
pspw
  simulation_cell units angstroms
     boundary_conditions aperiodic
     lattice
       lat_a 10.00d0
       lat_b 10.00d0
       lat_c 10.00d0
     end
     ngrid 40 40 40
  end
  Car-Parrinello
    fake_mass 600.0
    time_step 5.0
    loop 10 10
  end
end
set nwpw:minimizer 2
task paw energy
task paw Car-Parrinello
```

12.1.33 NWPW Capabilities and Limitations

- Hybrid Functionals (e.g. PBE0, LDA-SIC) only work in PSPW.
- Wannier orbital task only works in PSPW.
- AIMD/MM simulation only works with PSPW.

12.1.34 Questions and Difficulties

Questions and encountered problems should be reported to the NWChem Community Forum or to Eric J. Bylaska, Eric.Bylaska@pnl.gov

11.6 Tensor Contraction Engine Module: CI, MBPT, and CC

11.6.1 Overview

The Tensor Contraction Engine (TCE) Module of NWChem implements a variety of approximations that converge at the exact solutions of Schrödinger equation. They include configuration interaction theory through singles, doubles, triples, and quadruples substitutions, coupled-cluster theory through connected singles, doubles, triples, and quadruples substitutions, and many-body perturbation theory through fourth order in its tensor formulation. Not only optimized parallel programs of some of these high-end correlation theories are new, but also the way in which they have been developed is unique. The working equations of all of these methods have been derived completely automatically by a symbolic manipulation program called a Tensor Contraction Engine (TCE), and the optimized parallel programs have also been computer-generated by the same program, which were interfaced to NWChem. The development of the TCE program and this portion of the NWChem program has been financially supported by the United States Department of Energy, Office of Science, Office of Basic Energy Science, through the SciDAC program.

The capabilities of the module include:

- Restricted Hartree-Fock, unrestricted Hartree-Fock, and restricted open-shell Hartree-Fock references,
- Restricted KS DFT and unrestricted KS DFT references,
- Unrestricted configuration interaction theory (CISD, CISDT, and CISDTQ),

- Unrestricted coupled-cluster theory (LCCD, CCD, LCCSD, CCSD, QCISD, CCSDT, CCSDTQ),
- Unrestricted iterative many-body perturbation theory [MBPT(2), MBPT(3), MBPT(4)] in its tensor formulation,
- Unrestricted coupled-cluster singles and doubles with perturbative connected triples {CCSD(T), CCSD[T]},
- Unrestricted equation-of-motion coupled-cluster theory (EOM-CCSD, EOM-CCSDT, EOM-CCSDTQ) for excitation energies, transition moments and oscillator strengths, and excited-state dipole moments,
- Unrestricted coupled-cluster theory (CCSD, CCSDT, CCSDTQ) for dipole moments.
- Several variants of active-space CCSDt and EOMCCSDt methods that employ limited set of triply excited cluster amplitudes defined by active orbitals.
- Ground-state non-iterative CC approaches that account for the effect of triply and/or quadruply excited connected clusters: the perturbative approaches based on the similarity transformed Hamiltonian: CCSD(2), $CCSD(2)_T$, $CCSDT(2)_Q$, the completely and locally renormalized methods: CR-CCSD(T), LR-CCSD(T), LR-CCSD(TQ)-1.
- Excited-state non-iterative corrections due to triples to the EOMCCSD excitation energies: the completely renormalized EOMCCSD(T) method (CR-EOMCCSD(T)).
- Dynamic dipole polarizabilities at the CCSD and CCSDT levels using the linear response formalism.
- Ground- and excited- states the iterative second-order model CC2.
- Dynamic dipole polarizabilities at the CCSDTQ level using the linear response formalism.
- State-specific Multireference Coupled Cluster methods (MRCC) (Brillouin-Wigner (BW-MRCC) and Mukherjee (Mk-MRCC) approaches).
- Universally State Selective corrections to the BW-MRCC and Mk-MRCC methods (diagonal USS(2) and perturbative USS(pt) methods).
- Electron affinity/Ionization potential EOMCCSD formulations (EA/IP-EOMCC; available for RHF reference only).

The distributed binary executables do not contain CCSDTQ and its derivative methods, owing to their large volume. The source code includes them, so a user can reinstate them by setenv CCSDTQ yes and recompile TCE module. The following optimizations have been used in the module:

- Spin symmetry (spin integration is performed wherever possible within the unrestricted framework, making the present unrestricted program optimal for an open-shell system. The spin adaption was not performed, although in a restricted calculation for a closed-shell system, certain spin blocks of integrals and amplitudes are further omitted by symmetry, and consequently, the present unrestricted CCSD requires only twice as many operations as a spin-adapted restricted CCSD for a closed-shell system),
- Point-group symmetry,
- Index permutation symmetry,
- Runtime orbital range tiling for memory management,
- Dynamic load balancing (local index sort and matrix multiplications) parallelism,
- Multiple parallel I/O schemes including fully in-core algorithm using Global Arrays,
- Frozen core and virtual approximation.
- DIIS extrapolation and Jacobi update of excitation amplitudes
- Additional algorithms for the 2-e integral transformation, including efficient and scalable spin-free out-of-core N^5 algorithms.
- Hybrid I/O schemes for both spin-orbital and spin-free calculations which eliminate the memory bottleneck of the 2-e integrals in favor of disk storage. Calculations with nearly 400 basis functions at the CCSD(T) can be performed on workstation using this method.
- Parallel check-pointing and restart for ground-state (including property) calculations at the CCSD, CCSDT and CCSDTQ levels of theory.

11.6.2 Performance of CI, MBPT, and CC methods

For reviews or tutorials of these highly-accurate correlation methods, the user is referred to:

- Trygve Helgaker, Poul Jorgensen and Jeppe Olsen, Molecular Electronic-Structure Theory.
- A. Szabo and N. S. Ostlund, Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory.
- B. O. Roos (editor), Lecture Notes in Quantum Chemistry.

For background on development of the symbolic algebra tools which help create the code used in this model see:

- S. Hirata, J. Phys. Chem. A 107, 9887 (2003).
- S. Hirata, J. Chem. Phys. 121, 51 (2004).
- S. Hirata, Theor. Chem. Acc. 116, 2 (2006).

For details of particular CC implementations, see:

- S. Hirata, P.-D. Fan, A.A. Auer, M. Nooijen, P. Piecuch, J. Chem. Phys. 121, 12197 (2004).
- K. Kowalski, S. Hirata, M. Wloch, P. Piecuch, T.L. Windus, J. Chem. Phys. 123, 074319 (2005).
- K. Kowalski, P. Piecuch, J. Chem. Phys. 115, 643 (2001).
- K. Kowalski, P. Piecuch, Chem. Phys. Lett. 347, 237 (2001).
- P. Piecuch, S.A. Kucharski, and R.J. Bartlett, J. Chem. Phys. 110, 6103 (1999).
- P. Piecuch, N. Oliphant, and L. Adamowicz, J. Chem. Phys. 99, 1875 (1993).
- N. Oliphant and L. Adamowicz, Int. Rev. Phys. Chem. 12, 339 (1993).
- P. Piecuch, N. Oliphant, and L. Adamowicz, J. Chem. Phys. 99, 1875 (1993).
- K. Kowalski, P. Piecuch, J. Chem. Phys. 120, 1715 (2004).
- K. Kowalski, J. Chem. Phys. 123, 014102 (2005).
- P. Piecuch, K. Kowalski, I.S.O. Pimienta, M.J. McGuire, Int. Rev. Phys. Chem. 21, 527 (2002).
- K. Kowalski, P. Piecuch, J. Chem. Phys. 122, 074107 (2005)
- K. Kowalski, W. de Jong, J. Mol. Struct.: THEOCHEM, 768, 45 (2006).
- O. Christiansen, H. Koch, P. Jørgensen, Chem. Phys. Lett. 243, 409 (1995).
- K. Kowalski, J. Chem. Phys. 125, 124101 (2006).
- K. Kowalski, S. Krishnamoorthy, O. Villa, J.R. Hammond, N. Govind, J. Chem. Phys. 132, 154103 (2010).
- J.R. Hammond, K. Kowalski, W.A. de Jong, J. Chem. Phys. 127, 144105 (2007).
- J.R. Hammond, W.A. de Jong, K. Kowalski, J. Chem. Phys. 128, 224102 (2008).
- J.R. Hammond, K. Kowalski, J. Chem. Phys. 130, 194108 (2009).
- W. Ma. S. Krishnamoorthy, O. Villa, J. Chem. Theory Comput. 7, 1316 (2011).
- J.Brabec, J. Pittner, H.J.J. van Dam, E. Apra, K. Kowalski, J. Chem. Theory Comput. 8, 487 (2012).
- K. Bhaskaran-Nair, J. Brabec, E. Apra, H.J.J. van Dam. J. Pittner, K. Kowalski, J. Chem. Phys. 137, 094112 (2012).
- K. Bhaskaran-Nair, K. Kowalski, J. Moreno, M. Jarrell, W.A. Shelton, J. Chem. Phys. 141, 074304 (2014).
- J. Brabec, S. Banik, K. Kowalski, J. Pittner, J. Chem. Phys. 145, 164106 (2016).
- S. Rajbhandari, F. Rastello, K. Kowalski, S. Krishnamoorthy, P. Sadayappan, Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming 327 (2017).

11.6.3 Algorithms of CI, MBPT, and CC methods

11.6.3.1 Spin, spatial, and index permutation symmetry

The TCE thoroughly analyzes the working equation of many-electron theory models and automatically generates a program that takes full advantage of these symmetries at the same time. To do so, the TCE first recognizes the index permutation symmetries among the working equations, and perform strength reduction and factorization by carefully monitoring the index permutation symmetries of intermediate tensors. Accordingly, every input and output tensor (such as integrals, excitation amplitudes, residuals) has just two independent but strictly ordered index strings, and each intermediate tensor has just four independent but strictly ordered index strings. The operation cost and storage size of tensor contraction is minimized by using the index range restriction arising from these index permutation symmetries and also spin and spatial symmetry integration.

11.6.3.2 Runtime orbital range tiling

To maintain the peak local memory usage at a manageable level, in the beginning of the calculation, the orbitals are rearranged into tiles (blocks) that contains orbitals with the same spin and spatial symmetries. So the tensor contractions in these methods are carried out at the tile level; the spin, spatial, and index permutation symmetry is employed to reduce the operation and storage cost at the tile level also. The so-called tile-structure of all tensors used in CC equations is also the key-factor determining the parallel structure of the TCE CC codes . The tiling scheme corresponds to partitioning of the spin-orbital domain into smaller subsets containing the spin-orbitals of the same spin and spatial symmetries (the so-called tiles). This partitioning of the spin-orbital domain entails the blocking of all tensors corresponding to one- and two-electron integrals, cluster amplitudes, and all recursive intermediates, into smaller blocks of the size defined by the size of the tile (or tilesize for short). Since the parallel scheme used in all TCE generated codes is deeply rooted in dynamic load balancing techniques, the tile-structure defines the granularity of the work to be distributed. The size of tiles (tilesize) defines also the local memory requirements in all TCE derived CC implementations. For CI/CC/EOMCC/LR-CC models based on the sinlges and doubles models (CISD,CCSD,EOMCCSD,LR-CCSD) the peak local memory requirement is proportional to the $_{\text{tilesize}}{}^{4}$. In approaches accounting for triples, either in iterative or non-iterative fashion, the local memory usage is proportional to $_{\text{tilesize}}{}^{6}$. This means that in the CCSD(T), CCSDt, CCSDT, CR-EOMCCSD(T), EOMCCSDt, EOMCCSDT, LR-CCSDT caluclations the tilesize cannot be defined too large.

11.6.3.3 Dynamic load balancing parallelism

In a parallel execution, dynamic load balancing of tile-level local tensor index sorting and local tensor contraction (matrix multiplication) will be invoked.

11.6.3.4 Parallel I/O schemes

Each process is assigned a local tensor index sorting and tensor contraction dynamically. It must first retrieve the tiles of input tensors, and perform these local operations, and accumulate the output tensors to the storage. We have developed a uniform interface for these I/O operations to either (1) a global file on a global file system, (2) a global memory on a global or distributed memory system, and (3) semi-replicated files on a distributed file systems. Some of these operations depend on the ParSoft library.

11.6.4 Input syntax

The keyword to invoke the many-electron theories in the module is TCE. To perform a single-point energy calculation, include

```
TASK TCE ENERGY
```

in the input file, which may be preceded by the TCE input block that details the calculations:

```
TCE
 [(DFT||HF||SCF) default HF=SCF]
 [FREEZE [[core] (atomic || <integer nfzc default 0>)] \
        [virtual <integer nfzv default 0>]]
 [(LCCD||CCD||CCSD||CC2||LR-CCSD||LCCSD||CCSDT||CCSDTA||CCSDTQ|| \
   CCSD(T)||CCSD[T]||CCSD(2)_T||CCSD(2)||CCSDT(2)_Q|| \
   CR-CCSD[T]||CR-CCSD(T)|| \
   LR-CCSD(T)||LR-CCSD(TQ)-1||CREOMSD(T)|| \
   QCISD||CISD||CISDT||CISDTQ|| \
   MBPT2||MBPT3||MBPT4||MP2||MP3||MP4) default CCSD]
 [THRESH <double thresh default 1e-6>]
 [MAXITER <integer maxiter default 100>]
 [PRINT (none||low||medium||high||debug)
   <string list_of_names ...>]
 [IO (fortran||eaf||ga||sf||replicated||dra||ga_eaf) default ga]
 [DIIS <integer diis default 5>]
 [LSHIFT <double lshift default is 0.0d0>]
 [NROOTS <integer nroots default 0>]
 [TARGET <integer target default 1>]
 [TARGETSYM <character targetsym default 'none'>]
 [SYMMETRY]
 [2EORB]
 [2EMET <integer fast2e default 1>]
 [T3A_LVL]
 [ACTIVE_OA]
 [ACTIVE_OB]
 [ACTIVE_VA]
 [ACTIVE_VB]
 [DIPOLE]
 [TILESIZE <no default (automatically adjusted)>]
 [(NO)FOCK <logical recompf default .true.>]
 [FRAGMENT <default -1 (off)>]
END
```

Also supported are energy gradient calculation, geometry optimization, and vibrational frequency (or hessian) calculation, on the basis of numerical differentiation. To perform these calculations, use

```
TASK TCE GRADIENT
```

or

```
TASK TCE OPTIMIZE
```

or

```
TASK TCE FREQUENCIES
```

The user may also specify the parameters of reference wave function calculation in a separate block for either HF (SCF) or DFT, depending on the first keyword in the above syntax.

Since every keyword except the model has a default value, a minimal input file will be

```
GEOMETRY
  Be 0.0 0.0 0.0
END
BASIS
  Be library cc-pVDZ
END
TCE
  ccsd
END
TASK TCE ENERGY
```

which performs a CCSD/cc-pVDZ calculation of the Be atom in its singlet ground state with a spin-restricted HF reference.

New implementations of the iterative CCSD and EOMCCSD methods based on the improved task scheduling can be enable by the `set tce:nts T` command as in the following example:

```
geometry/basis set specifications
tce
freeze atomic
creomccsd(t)
tilesize 20
2eorb
2emet 13
eomsol 2
end

set tce:nts T

task tce energy
```

New task scheduling should reduce time to solutions and provide better parallel performance especially in large CCSD/EOMCCSD runs.

11.6.5 Keywords of TCE input block

11.6.5.1 HF, SCF, or DFT: the reference wave function

This keyword tells the module which of the HF (SCF) or DFT module is going to be used for the calculation of a reference wave function. The keyword HF and SCF are one and the same keyword internally, and are default. When these are used, the details of the HF (SCF) calculation can be specified in the SCF input block, whereas if DFT is chosen, DFT input block may be provided.

For instance, RHF-RCCSDT calculation (R standing for spin-restricted) can be performed with the following input blocks:

```
SCF
  SINGLET
  RHF
END
TCE
  SCF
  CCSDT
END
TASK TCE ENERGY
```

This calculation (and any correlation calculation in the TCE module using a RHF or RDFT reference for a closed-shell system) skips the storage and computation of all β spin blocks of integrals and excitation amplitudes. ROHF-UCCSDT (U standing for spin-unrestricted) for an open-shell doublet system can be requested by

```
SCF
  DOUBLET
  ROHF
END
TCE
  SCF
  CCSDT
END
TASK TCE ENERGY
```

and likewise, UHF-UCCSDT for an open-shell doublet system can be specified with

```
SCF
  DOUBLET
  UHF
END
TCE
  SCF
  CCSDT
END
TASK TCE ENERGY
```

The operation and storage costs of the last two calculations are identical. To use the KS DFT reference wave function for a UCCSD calculation of an open-shell doublet system,

```
DFT
  ODFT
  MULT 2
END
TCE
  DFT
  CCSD
END
TASK TCE ENERGY
```

Note that the default model of the DFT module is LDA.

11.6.5.2 CCSD,CCSDT,CCSDTQ,CISD,CISDT,CISDTQ, MBPT2,MBPT3,MBPT4, etc.: the correlation models

These keywords stand for the following models:

- LCCD: linearized coupled-cluster doubles,
- CCD: coupled-cluster doubles,
- LCCSD: linearized coupled-cluster singles & doubles,
- CCSD: coupled-cluster singles & doubles (also EOM-CCSD),
- CCSD_ACT: coupled-cluster singles & active doubles (also active-space EOMCCSD),
- LR-CCSD: locally renormalized EOMCCSD method.
- EACCSD: Electron affinity EOMCCSD method.
- IPCCSD: Ionization potential EOMCCSD method.
- CC2: second-order approximate coupled cluster with singles and doubles model
- CCSDT: coupled-cluster singles, doubles, & triples (also EOM-CCSDT),
- CCSDTA: coupled-cluster singles, doubles, & active triples (also EOM-CCSDT). Three variants of the active-space CCSDt and EOMCCSDt approaches can be selected based on various definitions of triply excited clusters: (1) version I (keyword T3A_LVL 1) uses the largest set of triply excited amplitudes defined by at least one occupied and one unoccupied active spinorbital labels. (2) Version II (keyword T3A_LVL 2) uses triply excited amplitudes that carry at least two occupied and unoccupied active spinorbital labels. (3) Version III (keyword T3A_LVL 3) uses triply excited amplitudes that are defined by active indices only. Each version requires defining relevant set of occupied active α and β spinorbitals (ACTIVE_OA and ACTIVE_OB) as well as active unoccupied α and β spinorbitals (ACTIVE_VA and ACTIVE_VB).
- CCSDTQ: coupled-cluster singles, doubles, triples, & quadruples (also EOM-CCSDTQ),
- CCSD(T): CCSD and perturbative connected triples,
- CCSD[T]: CCSD and perturbative connected triples,
- CR-CCSD[T]: completely renormalized CCSD[T] method,
- CR-CCSD(T): completely renormalized CCSD(T) method,
- CCSD(2)_T: CCSD and perturbative CCSD(T)_T correction,
- CCSD(2)_TQ: CCSD and perturbative CCSD(2) correction,
- CCSDT(2)_Q: CCSDT and perturbative CCSDT(2)_Q correction.
- LR-CCSD(T): CCSD and perturbative locally renormalized CCSD(T) correction,
- LR-CCSD(TQ)-1: CCSD and perturbative locally renormalized CCSD(TQ) (LR-CCSD(TQ)-1) correction,
- CREOMSD(T): EOMCCSD energies and completely renormalized EOMCCSD(T)(IA) correction. In this option NWCHEM prints two components: (1) total energy of the K-th state E_K=E_K EOMCCSD+δ_K $^{\text{CR-EOMCCSD(T),IA}}$(T) and (2) the so-called δ-corrected EOMCCSD excitation energy ω_K $^{\text{CR-EOMCCSD(T),IA}}$=$\omega_K$ $^{\text{EOMCCSD}}$+δ_K $^{\text{CR-EOMCCSD(T),IA}}$(T).
- CREOM(T)AC: active-space CR-EOMCCSD(T) approach,

- QCISD: quadratic configuration interaction singles & doubles,
- CISD: configuration interaction singles & doubles,
- CISDT: configuration interaction singles, doubles, & triples,
- CISDTQ: configuration interaction singles, doubles, triples, & quadruples,
- MBPT2=MP2: iterative tensor second-order many-body or Møller-Plesset perturbation theory,
- MBPT3=MP3: iterative tensor third-order many-body or Møller-Plesset perturbation theory,
- MBPT4=MP4: iterative tensor fourth-order many-body or Møller-Plesset perturbation theory,

All of these models are based on spin-orbital expressions of the amplitude and energy equations, and designed primarily for spin-unrestricted reference wave functions. However, for a restricted reference wave function of a closed-shell system, some further reduction of operation and storage cost will be made. Within the unrestricted framework, all these methods take full advantage of spin, spatial, and index permutation symmetries to save operation and storage costs at every stage of the calculation. Consequently, these computer-generated programs will perform significantly faster than, for instance, a hand-written spin-adapted CCSD program in NWChem, although the nominal operation cost for a spin-adapted CCSD is just one half of that for spin-unrestricted CCSD (in spin-unrestricted CCSD there are three independent sets of excitation amplitudes, whereas in spin-adapted CCSD there is only one set, so the nominal operation cost for the latter is one third of that of the former. For a restricted reference wave function of a closed-shell system, all β spin block of the excitation amplitudes and integrals can be trivially mapped to the all α spin block, reducing the ratio to one half).

While the MBPT (MP) models implemented in the TCE module give identical correlation energies as conventional implementation for a canonical HF reference of a closed-shell system, the former are intrinsically more general and theoretically robust for other less standard reference wave functions and open-shell systems. This is because the zeroth order of Hamiltonian is chosen to be the full Fock operator (not just the diagonal part), and no further approximation was invoked. So unlike the conventional implementation where the Fock matrix is assumed to be diagonal and a correlation energy is evaluated in a single analytical formula that involves orbital energies (or diagonal Fock matrix elements), the present tensor MBPT requires the iterative solution of amplitude equations and subsequent energy evaluation and is generally more expensive than the former. For example, the operation cost of many conventional implementation of MBPT(2) scales as the fourth power of the system size, but the cost of the present tensor MBPT(2) scales as the fifth power of the system size, as the latter permits non-canonical HF reference and the former does not (to reinstate the non-canonical HF reference in the former makes it also scale as the fifth power of the system size).

11.6.5.3 State-Specific Multireference Coupled Cluster methods (MRCC)

Several State-Specific MRCC methods have been implemented in 6.3 release of nwchem. These include:

- Iterative Brillouin-Wigner MRCC method employing single and double excitations (BW-MRCCSD)
- Iterative Mukherjee MRCC method employing single and double excitations (Mk-MRCCSD)
- Non-iterative methods accounting for triple excitations in MRCC formalisms: BW-MRCCSD(T) and Mk-MRCCSD(T)

The current implementation can be used in studies of systems composed of an even number of correlated electrons (this limitation will be removed in the next release). This includes typical examples of diradical, open-shell singlets, and bond-forming/breaking processes where the corresponding wavefunctions have strong quasidegenerate character.

To enable the compilation of the MRCC codes one has to set the following variable before the compilation of NWChem

```
export MRCC_METHODS=y
```

To run MRCC calculations the user has to define two groups in the input file. First, the TCE group and secondly the MRCCDATA group. In the TCE group the iterative level of theory is defined, e.g. BWCCSD or MKCCSD. This implementation was designed for complete model spaces (CMS) which means that the modelspace contains all Slater determinants of all possible (in the context of the spatial and spin symmetry, M_S) distributions of active electrons among active spin orbitals. The user can define the modelspace in two ways. As a first approach the model space can be defined

by hand, as shown in the two examples below. The input of the model space starts with the `NREF` keyword followed by the number of reference configurations that will be used, which should equal the number of strings for references below. In the input `2` refers to doubly occupied orbitals, `a` to alpha electrons, `b` to beta electrons and `0` identifies an unoccupied orbital. When the model space is defined by hand the occupation strings have to include the frozen orbitals as well. In the second way the CMS can be generated using the keyword `CAS` followed by the number of active electrons and the number of active orbitals. When using the `CAS` keyword we strongly recommend that users check the references that are generated.

As the model space typically includes multiple configurations it is possible to use the MRCC method to calculate excited states instead of the ground state. For this reason it is required to specify the root of interest. The `ROOT` keyword followed by the root number specifies the state the code calculates. The lowest root, the ground state, is identified as `root 1`. If one wants to calculate the third root the keyword `ROOT 3` should be used. An example is given below.

```
echo
start tce_mrcc_bwcc
memory stack 1000 mb heap 100 mb global 500 mb verify
geometry units au
  H         0.00000000    -2.27289450    -1.58834700
  O         0.00000000     0.00000000    -0.01350000
  H         0.00000000     2.27289450    -1.58834700
end
basis spherical
  O library cc-pvdz
  H library cc-pvdz
end
charge 0
scf
  rohf
  singlet
  thresh 1.0e-10
  tol2e 1.0e-10
end
tce
  bwccsd
  thresh 1.0e-7
  targetsym a1
  io ga
  tilesize 18
end
mrccdata
  root 1
  nref 4
  222220
  222202
  2222ab
  2222ba
end
task tce energy
```

```
echo
start tce_mrcc_mkcc
memory stack 1000 mb heap 100 mb global 500 mb verify
geometry units au
  H          0.00000000    -2.27289450    -1.58834700
  O          0.00000000     0.00000000    -0.01350000
  H          0.00000000     2.27289450    -1.58834700
end
basis spherical
  O library cc-pvdz
  H library cc-pvdz
end
charge 0
scf
  rohf
  singlet
  thresh 1.0e-10
  tol2e 1.0e-10
end
tce
  mkccsd
  thresh 1.0e-5
  targetsym a1
  maxiter 100
  io ga
  tilesize 18
end
mrccdata
  root 1
  cas 2 2 # Please make sure the references generated are correct.
end
task tce energy
```

This version of MRCC works only with GA as specified by the `IO GA` option. In addition this code works only with the spin-orbit 4-index transformation, however in all circumstances an RHF Hartree-Fock initial calculation has to be used. In this implementation the effective Hamiltonian operator contains only scalar, one- and two-body many body components. Finally, in our implementation the BWCCSD methods use the energy threshold for the convergence, whereas the MKCCSD method uses the norm of the residual.

In addition to the iterative single-double calculations the code can calculate non-iterative triples corrections. To request these triples corrections the keyword `SE4T` should be added to the MRCCDATA block. The implementation details and the from of the triples correction are given in equation 20 [J. Chem. Phys. 137, 094112 (2012)].

```
echo
start tce_mrcc_bwcc
memory stack 1000 mb heap 100 mb global 500 mb verify
geometry units au
  H          0.00000000    -2.27289450    -1.58834700
  O          0.00000000     0.00000000    -0.01350000
  H          0.00000000     2.27289450    -1.58834700
end
basis spherical
  O library cc-pvdz
  H library cc-pvdz
end
charge 0
scf
  rohf
  singlet
  thresh 1.0e-10
  tol2e 1.0e-10
end
tce
  bwccsd
  thresh 1.0e-7
  targetsym a1
  io ga
  tilesize 18
end
mrccdata
  se4t
  no_aposteriori
  root 1
  nref 4
  222220
  222202
  2222ab
  2222ba
end
task tce ener
```

```
echo
start tce_mrcc_mkcc
memory stack 1000 mb heap 100 mb global 500 mb verify
geometry units au
  H         0.00000000    -2.27289450    -1.58834700
  O         0.00000000     0.00000000    -0.01350000
  H         0.00000000     2.27289450    -1.58834700
end
basis spherical
  O library cc-pvdz
  H library cc-pvdz
end
charge 0
scf
  rohf
  singlet
  thresh 1.0e-10
  tol2e 1.0e-10
end
tce
  mkccsd
  thresh 1.0e-5
  targetsym a1
  io ga
  tilesize 18
  maxiter 100
end
mrccdata
  se4t
  root 1
  nref 4
  222220
  222202
  2222ab
  2222ba
end
task tce ener
```

11.6.5.4 Implementation notes for reference-level-parallelism in MRCC

The current version of the MRCC codes contains also a pilot implementation of the reference-level-parallelism based on the use of processor groups for BWCCSD and BWCCSD(T) approaches. The main ideas of this approach have been described in

- J.Brabec, J. Pittner, H.J.J. van Dam, E. Apra, K. Kowalski,J. Chem. Theory Comput. 8, 487 (2012).
- K. Bhaskaran-Nair, J. Brabec, E. Apra, H.J.J. van Dam. J. Pittner, K. Kowalski, J. Chem. Phys. 137, 094112 (2012).

Two essential keywords have to be added to the `mrccdata` block of the input:

```
subgroupsize n
improvetiling
```

and

```
diis 0
```

in tce block. The line `subgroupsize n` defines the size of the subgroup and `improvetiling` refers to the data representation in the MRCC subgroup algorithm. For example, if user has 4 references and total 32 cores/CPU then n should be defined as 32/4=8. If user has 10 references and 1200 cores/CPU available then the size of the subgroupsize (n) is 120.

```
echo
start tce_mrcc_bwcc_subgroups
memory stack 1000 mb heap 100 mb global 500 mb verify
geometry units au
  H          0.00000000    -2.27289450    -1.58834700
  O          0.00000000     0.00000000    -0.01350000
  H          0.00000000     2.27289450    -1.58834700
end
basis spherical
  O library cc-pvdz
  H library cc-pvdz
end
charge 0
scf
  rohf
  singlet
  thresh 1e-12
  tol2e 1e-12
end
tce
  bwccsd
  targetsym a1
  io ga
  diis 0
  thresh 1e-7
  tilesize 18
end
mrccdata
  subgroupsize 2 # Please read the documentation below.
  improvetiling
  root 1
  cas 2 2
end
task tce ener
```

CAUTION: Before using the subgroup-based algorithm the users should perform the GA subgroup test in `$NWCHEM_TOP/src/tools/ga-5-6-3/global/testing/pgtest.x` and `pg2test.x` in the same location. Additionally it is strongly encouraged to run the NWChem QA tests from the $NWCHEM_TOP/QA/tests/tce_mrcc_bwcc_subgroups directory with various combinations of subgroup size and total number of CPU.

The USS corrections can be enabled by using `usspt` directive keyword in the mrccdata

```
tce
  mkccsd
  thresh 1.0e-10
  targetsym a1
  maxiter 600
  io ga
end

mrccdata
  usspt
  root 1
  cas 2 2
end
```

In effect both diagonal and perturbative USS corrections will be calculated after the completion of iterative Mk-MRCCSD or BW-MRCCSD calculations.

11.6.5.5 Electron affinity, ionization potential EOMCCSD methods

The EA/IP-EOMCCSD methodologies are available in the 6.5 NWChem release. These implementation are available for the RHF type of the reference function. To enable the compilation of the EA/IP-EOMCCSD codes one has to set the following variable before the compilation of NWChem

```
export EACCSD=y
export IPCCSD=y
```

Two input examples for the EA/IP-EOMCCSD calculations are shown below.

- EA-EOMCCSD calculations for the ozone molecule

```
start tce_eaccsd_ozone
title "tce_eaccsd_ozone"
echo

memory stack 1000 mb heap 200 mb global 500 mb

geometry units bohr
symmetry c1
O   0.0000000000     0.0000000000     0.0000000000
O   0.0000000000    -2.0473224350    -1.2595211660
O   0.0000000000     2.0473224350    -1.2595211660
end

basis spherical
 * library cc-pvdz
end

scf
thresh 1.0e-10
tol2e 1.0e-10
singlet
rhf
end

tce
eaccsd
nroots 2
freeze atomic
tilesize 20
thresh 1.0d-6
end

task tce energy
```

- IP-EOMCCSD calculations for the F2 molecule

```
start  tce_ipccsd_f2
title "tce_ipccsd_f2"
echo

memory stack 1000 mb heap 200 mb global 500 mb

geometry units angstroms
symmetry c1
 F    0.0000000000     0.0000000000   0.7059650
 F    0.0000000000     0.0000000000  -0.7059650
end

basis spherical
 * library cc-pvdz
end

scf
thresh 1.0e-10
tol2e 1.0e-10
singlet
rhf
end

tce
ipccsd
nroots 1
freeze atomic
thresh 1.0e-7
end

task tce energy
```

As in the EOMCCSD input we can request any number of roots.

In analogy to the EOMCC calculations we can customize the number of initial guesses by using`set tce:maxeorb` directive. For example for system with the symmetry with the orbital energy structure shown below *ea ip* one can use the energy window (in the sense of the absolute value of the HF orbital energies) to pinpoint the initial guesses. If one is interested in calculating one EA-EOMCCSD root of the a1 symmetry the

```
set tce:maxeorb 0.1
```

should be used. This means that the number of starting vectors will be equal to the number of the unoccupied a1 symmetry orbitals with the corresponding orbital energies less than 0.1 (in our example there will be only one such a vector

corresponding to the unoccupied orbital energy 0.072). If one looks for two roots

```
set tce:maxeorb 0.16
```

option should be used(there are two a1 unoccupied orbitals with energies less than 0.16).

For the IP-EOMCCSD case the `set tce:maxeorb` option works in a similar way. For example if one is looks for 1 IP-EOMCCSD root of a1 symmetry ,

```
set tce:maxeorb 0.24
```

directive should be used (there is only one occupied orbital of a1 symmetry with the absolute value of orbital energy less than 0.24), etc.

11.6.5.6 THRESH: the convergence threshold of iterative solutions of amplitude equations

This keyword specifies the convergence threshold of iterative solutions of amplitude equations, and applies to all of the CI, CC, and MBPT models. The threshold refers to the norm of residual, namely, the deviation from the amplitude equations. The default value is 1e-6.

11.6.5.7 MAXITER: the maximum number of iterations

It sets the maximum allowed number iterations for the iterative solutions of amplitude equations. The default value is 100.

11.6.5.8 IO: parallel I/O scheme

There are five parallel I/O schemes implemented for all the models, which need to be wisely chosen for a particular problem and computer architecture.

- fortran : Fortran77 direct access,
- eaf : Exclusive Access File library,
- ga : Fully incore, Global Array virtual file,
- sf : Shared File library,
- replicated : Semi-replicated file on distributed file system with EAF library.
- dra : Distributed file on distributed file system with DRA library.
- ga_eaf : Semi-replicated file on distributed file system with EAF library. GA is used to speedup the file reconciliation.

The GA algorithm, which is default, stores all input (integrals and excitation amplitudes), output (residuals), and intermediate tensors in the shared memory area across all nodes by virtue of GA library. This fully incore algorithm replaces disk I/O by inter-process communications. This is a recommended algorithm whenever feasible. Note that the memory management through runtime orbital range tiling described above applies to local (unshared) memory of each node, which may be separately allocated from the shared memory space for GA. So when there is not enough shared memory space (either physically or due to software limitations, in particular, shmmax setting), the GA algorithm can crash due to an out-of-memory error. The replicated scheme is the currently the only disk-based algorithm for a genuinely distributed file system. This means that each node keeps an identical copy of input tensors and it holds non-identical overlapping segments of intermediate and output tensors in its local disk. Whenever data coherency is required, a file reconcilation process will take place to make the intermediate and output data identical throughout the nodes. This algorithm, while requiring redundant data space on local disk, performs reasonably efficiently in parallel. For sequential execution, this reduces to the EAF scheme. For a global file system, the SF scheme is recommended. This together with the Fortran77 direct access scheme does not usually exhibit scalability unless shared files on the global file system also share the same I/O buffer. For sequential executions, the SF, EAF, and replicated schemes are interchangeable, while the Fortran77 scheme is appreciably slower.

Two new I/O algorithms dra and ga_eaf combines GA and DRA or EAF based replicated algorithm. In the former, arrays that are not active (e.g., prior T amplitudes used in DIIS or EOM-CC trial vectors) in GA algorithm will be moved to DRA. In the latter, the intermediates that are formed by tensor contractions are initially stored in GA, thereby avoiding the need to accumulate the fragments of the intermediate scattered in EAFs in the original EAF algorithm. Once the intermediate is formed completely, then it will be replicated as EAFs.

The spin-free 4-index transformation algorithms are exclusively compatible with the GA I/O scheme, although out-of-core algorithms for the 4-index transformation are accessible using the 2emet options. See Alternative storage of two-electron integrals for details.

11.6.5.9 DIIS: the convergence acceleration

It sets the number iterations in which a DIIS extrapolation is performed to accelerate the convergence of excitation amplitudes. The default value is 5, which means in every five iteration, one DIIS extrapolation is performed (and in the rest of the iterations, Jacobi rotation is used). When zero or negative value is specified, the DIIS is turned off. It is not recommended to perform DIIS every iteration, whereas setting a large value for this parameter necessitates a large memory (disk) space to keep the excitation amplitudes of previous iterations.
Another tool for convergence acceleration is the level shift option (`lshift` keyword) that allows to increase small orbital energy differences used in calculating the up-dates for cluster amplitudes. Typical values for `lshift` oscillates between 0.3 and 0.5 for CC calculations for ground states of multi-configurational character. Otherwise, the value of `lshift` is by default set equal to 0.

11.6.5.10 FREEZE: the frozen core/virtual approximation

Some of the lowest-lying core orbitals and/or some of the highest-lying virtual orbitals may be excluded in the calculations by this keyword (this does not affect the ground state HF or DFT calculation). No orbitals are frozen by default. To exclude the atom-like core regions altogether, one may request

```
FREEZE atomic
```

To specify the number of lowest-lying occupied orbitals be excluded, one may use

```
FREEZE 10
```

which causes 10 lowest-lying occupied orbitals excluded. This is equivalent to writing

```
FREEZE core 10
```

To freeze the highest virtual orbitals, use the virtual keyword. For instance, to freeze the top 5 virtuals

```
FREEZE virtual 5
```

11.6.5.11 NROOTS: the number of excited states

One can specify the number of excited state roots to be determined. The default value is 1. It is advised that the users request several more roots than actually needed, since owing to the nature of the trial vector algorithm, some low-lying roots can be missed when they do not have sufficient overlap with the initial guess vectors.

11.6.5.12 TARGET and TARGETSYM: the target root and its symmetry

At the moment, the first and second geometrical derivatives of excitation energies that are needed in force, geometry, and frequency calculations are obtained by numerical differentiation. These keywords may be used to specify which excited state root is being used for the geometrical derivative calculation. For instance, when `TARGET 3` and `TARGETSYM a1g` are included in the input block, the total energy (ground state energy plus excitation energy) of the third lowest excited state

root (excluding the ground state) transforming as the irreducible representation a1g will be passed to the module which performs the derivative calculations. The default values of these keywords are 1 and none, respectively.

The keyword TARGETSYM is essential in excited state geometry optimization, since it is very common that the order of excited states changes due to the geometry changes in the course of optimization. Without specifying the TARGETSYM, the optimizer could (and would likely) be optimizing the geometry of an excited state that is different from the one the user had intended to optimize at the starting geometry. On the other hand, in the frequency calculations, TARGETSYM must be none, since the finite displacements given in the course of frequency calculations will lift the spatial symmetry of the equilibrium geometry. When these finite displacements can alter the order of excited states including the target state, the frequency calculation is not be feasible.

11.6.5.13 SYMMETRY: restricting the excited state symmetry

By adding this keyword to the input block, the user can request the module to seek just the roots of the specified irreducible representation as TARGETSYM. By default, this option is not set. TARGETSYM must be specified when SYMMETRY is invoked.

11.6.5.14 EOMSOL: alternative solver for the right EOMCCSD eigenvalue problem

The EOMSOL enables the user to switch between two algorithms for solving EOMCCSD eigenproblem. When EOMSOL is set equal to 1 (eomsol 1 directive in the tce group) the old solver is invoked. The advantage of this solver is a possibility of finding states of complicated configurational structure, for example doubly excited states. However, the dimension of the iterative space increases with each iteration and in effect this algorithm requires large memory allocations especially for large systems. In order to address this bottleneck, new algorithm (eomsol 2 directive in the tce group) was designed. In EOMSOL 2 algorithm all iterations are split into microcycles corresponding to diis microiterations (the use of diis parameter is discussed earlier). This algorithm enables the user to precisely estimate the memory usage in the EOMCCSD calculations, which is equal to diis*nroots*(size_x1+size_x2), where diis is the length of the DIIS cycle, nroots is the number of sought roots, size_x1 corresponds to the size of GA storing singly excited EOMCC almplitudes, and size_x2 is the size of GA with doubly excited EOMCC amplitudes. Generally, larger values of diis parameter lead to a faster convergence, however, this happens at the expense of larger memory requirements. It is recommended not to use in the EOMCCSD calculations with eomsol 2 diis parameter smaller than 5, which is its default value. The EOMSOL 2 algorithm uses the CIS vectors as initial guesses, and for this reason is suited mainly to track singly excited states. By default, the EOMSOL 1 option is called in the EOMCCSD calculations. It should be also stressed that all iterative EOMCC methods with higher than double excitations use EOMSOL 1 approach.

In some situations it is convenient to use separate convergence threshold for the CCSD and EOMCCSD solvers. This can be achieved by setting proper environmetal variables. In the following example

```
geometry/basis set specifications
tce
 thresh 1.0d-6
 ccsd
 nroots 2
end
set tce:thresheom 1.0d-4
task tce energy
```

the CCSD equations will be converged to the 1.0d-6 threshold while the EOMCCSD ones to 1.0d-4. This option shoul dbe used with the eomsol 2 option. In some situations finding several (n) roots to the EOMCCSD equations can be quite challenging. To by-pass this problem one can use the “n+1” model, i.e., we request another root to be converged. Usually, the presence the “buffer” root can imporve the iterative process for n roots of interest. However, the buffer root does not have to be converged to the same accuracy as n roots of interest. The follwing example, shows how to handle this process (we chose n=2, n+1=3):

```
geometry/basis set specifications
tce
 freeze core
 ccsd
 nroots 3
 thresh 1.0d-6
end
set tce:thresheom 1.0d-4
set tce:threshl 1.0d-3
task tce energy
```

In this example the CCSD equations are solved with the 1.0d-6 threshold, the first n (2) EOMCCSD roots are determined with the 10d-4 accuracy, while the buffer root is determined with relax conv. criterion 1.0d-3.

11.6.5.15 2EORB: alternative storage of two-electron integrals

In the 5.0 version a new option has been added in order to provide more economical way of storing two-electron integrals used in CC calculations based on the RHF and ROHF references. The `2EORB` keyword can be used for all CC methods except for those using an active-space (CCSDt) up to NWChem version 6.3. After that, further optimization restricted the use of `2EORB` to CCSD-based methods. Note that the four-index transformation is usually an insignificant amount of the wall time for methods involving iterative triples anyway. With 2EORB, all two-electron integrals are transformed and subsequently stored in a way which is compatible with assumed tiling scheme. The transformation from orbital to spinorbital form of the two-electron integrals is performed on-the-fly during execution of the CC module. This option, although slower, allows to significantly reduce the memory requirements needed by the first half of 4-index transformation and final file with fully transformed two-electron integrals. Savings in the memory requirements on the order of magnitude (and more) have been observed for large-scale open-shell calculations.

11.6.5.16 2EMET: alternative storage of two-electron integrals

Several new computation-intensive algorithms has been added with the purpose of improving scalability and overcoming local memory bottleneck of the 5.0 `2EORB` 4-index transformation. In order to give the user a full control over this part of the TCE code several keywords were designed to define the most vital parameters that determine the perfromance of 4-index transformation. All new keywords must be used with the `2EORB` keyword, and thus will not work beyond CCSD methods after NWChem 6.3 (see explanation for `2EORB` above). The 2emet keyword (default value 1 or `2emet 1`, refers to the older 4-index transformation), defines the algorithm to be used. By putting `2emet 2` the TCE code will execute the algoritm based on the two step procedure with two intermediate files. In some instances this algorithm is characterized by better timings compared to algorithms 3 and 4, although it is more memory demanding. In contrast to algorithms nr 1,3, and 4 this approach can make use of a disk to store intermediate files. For this purpose one should use the keyword `idiskx` (`idiskx 0` causes that all intermediate files are stored on global arrays, while `idiskx 1` tells the code to use a disk to store intermediates; default value of `idiskx` is equal 0). Algorithm nr 3 (`2emet 3`) uses only one intermediate file whereas algorithm nr 4 (`2emet 4`) is a version of algorithm 3 with the option of reducing the memory requirements. For example, by using the new keyword `split 4` we will reduce the size of only intermediate file by factor of 4 (the `split` keyword can be only used in the context of algorithm nr 4). All new algorithms (i.e. 2emet 2+) use the attilesize setting to define the size of the atomic tile. By default `attilesize` is set equal 30. For larger systems the use of larger values of `attilesize` is recommended (typically between 40-60).

Additional algorithms are numbered 5, 6 and 9. Other values of `2emet` are not supported and refer to methods which do not function properly. Algorithms 5 and 6 were written as out-of-core N^5 methods (`idiskx1`) and are the most efficient algorithms at the present time. The corresponding in-core variants (`idiskx 0`) are available but require excessive memory with respect to the methods discussed above, although they may be faster if sufficient memory is available (to get enough memory often requires excessive nodes, which decreases performance in the later stages of the calculation). The difference between 5 and 6 is that 5 writes to a single file (SF or GA) while 6 uses multiple files. For smaller calculations, particularly single-node jobs, 5 is faster than 6, but for more than a handful of processors, algorithm 6 should be used. The perforance discrepancy depends on the hardware used but in algorithm eliminates simulataneous disk access on parallel file systems or memory mutexes for the in-core case. For NFS filesystems attached to parallel clusters, no performance differences have been observed, but for Lustre and PVFS they are signficant. Using algorithm 5 for large parallel file systems will make the file system inaccessible to other users, invoking the wrath of system administrators.

Algorithm 9 is an out-of-core solution to the memory bottleneck of the 2-e integrals. In this approach, the intermediates of the 4-index transformation as well as the MO integrals are stored in an SF file. As before, this requires a shared file system. Because algorithm 9 is based upon algorithm 5, described above, it is not expected to scale. The primary purpose of algorithm 9 is to make the performance of the NWChem coupled-cluster codes competive with fast serial codes on workstations. It succeeds in this purpose when corresponding functionality is compared. A more scalable version of this algorithm is possible, but the utility is limited since large parallel computers do not permit the wall times necessary to use an out-of-core method, which is necessarily slower than the in-core variant. An extensible solution to these issues using complex heterogeneous I/O is in development. Restarting with algorithm 9 is not supported and attempting to use this feature with the present version may produce meaningless results.

New is the inclusion of multiple 2emet options for the spin-orbital transformations, which are the default when 2eorb is not set and are mandatory for UHF and KS references. The are currently three algorithms 1, 2 and 3 available. The numbering scheme does not correspond in any way to the numbering scheme for the 2eorb case, except that 2emet 1 corresponds to the default algorithm present in previous releases, which uses the user-defined I/O scheme. Algorithm 2 (2emet 2) writes an SF file for the half-transformed integrals, which is at least an order-of-magnitude larger than the fully-transformed integrals, but stores the fully-transformed integrals in core. Thus, once the 4-index transformation is complete, this algorithm will perform exactly as when algorithm 1 is used. Unfortuntely, the spin-orbital 2-e fully-transformed integrals are still quite large and an algorithm corresponding to 2eorb/2emet=9 is available with 2emet 3 . Algorithm 3 is also limited in its scalability, but it permits relatively large UHF-based calculations using single workstations for patient users.

In cases where the user has access to both shared and local filesystems for parallel calculations, the permanent_dir setting refers to the location of SF files. The file system for scratch_dir will not be used for any of the 4-index transformation algorithms which are compatible with io=ga .

Algorithms 13 and 14 are the N^5 variants of algorithms 3 and 4. They are the most efficient in core GA-based algorithms for the RHF and ROHF reference functions. Again, two parameters are needed to define the perfromance of these algorithms: tilesize and attilesize. By default attilesize is set equal to 40. In all runs tilesize is required to be less than attilesize (tilesize < attilesize).

New 4-index transformation for RHF/ROHF references (2emet 15) is available in NWChem 6.5. In contrast to algorithms 13 and 14 inter-processor communication is significantly reduced resulting in much better performance.

In the later part of this manual several examples illustrate the use of the newly introduced keywords.

An efficient loop-fused four-index transfromations for RHF and ROHF references can be enabled by the sequence 2eorb/2emet 16 .

11.6.5.17 CCSD(T)/CR-EOMCCSD(T) calculations with large tiles

In 6.5 version of NWChem we have enabled versions of the CCSD(T) and CR-EOMCCSD(T) codes, which by-pass the local memory limitation of previous implementations. For this purpose a sliced versions of the CCSD(T)/CR-EOMCCSD(T) codes have been developed (see K. Kowalski, S. Krishnamoorthy, R. Olson, V. Tipparaju, E. Apra, Supercomputing 2011, Seattle). In order to enable these versions it is enough to add

```
set tce:xmem 100
```

which defines maximum memory size (in MB) for the slice of 6-dimensional tensors (in the current example 100 MB; for more details see QA tests tce_ccsd_t_xmem and tce_cr_eomccsd_t_xmem).

11.6.5.18 DIPOLE: the ground- and excited-state dipole moments

When this is set, the ground-state CC calculation will enter another round of iterative step for the so-called Λ equation to obtain the one-particle density matrix and dipole moments. Likewise, for excited-states (EOM-CC), the transition moments and dipole moments will be computed when (and only when) this option is set. In the latter case, EOM-CC left hand side solutions will be sought incurring approximately three times the computational cost of excitation energies alone (note that

the EOM-CC effective Hamiltonian is not Hermitian and has distinct left and right eigenvectors).

11.6.5.19 (NO)FOCK: (not) recompute Fock matrix

The default is FOCK meaning that the Fock matrix will be reconstructed (as opposed to using the orbital energies as the diagonal part of Fock). This is essential in getting correct correlation energies with ROHF or DFT reference wave functions. However, currently, this module cannot reconstruct the Fock matrix when one-component relativistic effects are operative. So when a user wishes to run TCE's correlation methods with DK or other relativistic reference, NOFOCK must be set and orbital energies must be used for the Fock matrix.

11.6.5.20 DENSMAT

Generate Density Matrix that can be used in the DPLOT module as described in the Example section.

11.6.5.21 PRINT: the verbosity

This keyword changes the level of output verbosity. One may also request some particular items in the table below.

| Item | Print Level | Description | | ---------- | ---------- | ---------- | | "time" | vary | CPU and wall times | | "tile" | vary | Orbital range tiling information | | "t1" | debug | T_1 excitation amplitude dumping | | "t2" | debug | T_2 excitation amplitude dumping | | "t3" | debug | T_3 excitation amplitude dumping | | "t4" | debug | T_4 excitation amplitude dumping | | "general information" | default | General information | | "correlation information" | default | TCE information | | "mbpt2" | debug | Canonical HF MBPT2 test | | "get_block" | debug | I/O information | | "put_block" | debug | I/O information | | "add_block" | debug | I/O information | | "files" | debug | File information | | "offset" | debug | File offset information | | "ao1e" | debug | AO one-electron integral evaluation | | "ao2e" | debug | AO two-electron integral evaluation | | "mo1e" | debug | One-electron integral transformation | | "mo2e" | debug | Two-electron integral transformation | Printable items in the TCE modules and their default print levels

11.6.6 Sample input

The following is a sample input for a ROHF-UCCSD energy calculation of a water radical cation.

```
START h2o
TITLE "ROHF-UCCSD/cc-pVTZ H2O"
CHARGE 1
GEOMETRY
 O     0.00000000     0.00000000     0.12982363
 H     0.75933475     0.00000000    -0.46621158
 H    -0.75933475     0.00000000    -0.46621158
END
BASIS
 * library cc-pVTZ
END
SCF
 ROHF
 DOUBLET
 THRESH 1.0e-10
 TOL2E  1.0e-10
END
TCE
 CCSD
END
TASK TCE ENERGY
```

The same result can be obtained by the following input:

```
START h2o
TITLE "ROHF-UCCSD/cc-pVTZ H2O"
CHARGE 1
GEOMETRY
 O     0.00000000     0.00000000     0.12982363
 H     0.75933475     0.00000000    -0.46621158
 H    -0.75933475     0.00000000    -0.46621158
END
BASIS
 * library cc-pVTZ
END
SCF
 ROHF
 DOUBLET
 THRESH 1.0e-10
 TOL2E  1.0e-10
END
TASK UCCSD ENERGY
```

EOMCCSD calculations with EOMSOL 2 algorithm. In these calculations the diis value of 8 will be used both in the CCSD and EOMCCSD iterations.

```
TITLE "tce_eomccsd_eomsol2"
ECHO
START tce_eomccsd_eomsol2
GEOMETRY UNITS ANGSTROM
N           .034130    -.986909     .000000
N         -1.173397     .981920     .000000
C         -1.218805    -.408164     .000000
C          -.007302    1.702153     .000000
C          1.196200    1.107045     .000000
C          1.289085    -.345905     .000000
O          2.310232    -.996874     .000000
O         -2.257041   -1.026495     .000000
H           .049329   -1.997961     .000000
H         -2.070598    1.437050     .000000
H          -.125651    2.776484     .000000
H          2.111671    1.674079     .000000
END
BASIS
 * library 6-31G
END
SCF
 THRESH 1.0e-10
 TOL2E 1.0e-10
 SINGLET
 RHF
END
TCE
 FREEZE ATOMIC
 CREOMSD(T)
 EOMSOL 2
 DIIS 8
 TILESIZE 15
 THRESH 1.0d-5
 2EORB
 2EMET 13
 NROOTS 1
END
TASK TCE ENERGY
```

EOM-CCSDT calculation for excitation energies, excited-state dipole, and transition moments.

```
START tce_h2o_eomcc
GEOMETRY UNITS BOHR
 H    1.474611052297904   0.000000000000000   0.863401706825835
 O    0.000000000000000   0.000000000000000  -0.215850436155089
 H   -1.474611052297904   0.000000000000000   0.863401706825835
END
BASIS
 * library sto-3g
END
SCF
 SINGLET
 RHF
END
TCE
 CCSDT
 DIPOLE
 FREEZE CORE ATOMIC
 NROOTS 1
END
TASK TCE ENERGY
```

Active-space CCSDt/EOMCCSDt calculations (version I) of several excited states of the Be_3 molecule. Three highest-lying occupied α and β orbitals (active_oa and active_ob) and nine lowest-lying unoccupied α and β orbitals (active_va and active_vb) define the active space.

```
START TCE_ACTIVE_CCSDT
ECHO
GEOMETRY UNITS ANGSTROM
SYMMETRY C2V
 BE  0.00  0.00   0.00
 BE  0.00  1.137090 -1.96949
end
BASIS spherical
 # --- DEFINE YOUR BASIS SET ---
END
SCF
 THRESH 1.0e-10
 TOL2E 1.0e-10
 SINGLET
 RHF
END
TCE
 FREEZE ATOMIC
 CCSDTA
 TILESIZE 15
 THRESH 1.0d-5
 ACTIVE_OA 3
 ACTIVE_OB 3
 ACTIVE_VA 9
 ACTIVE_VB 9
 T3A_LVL 1
 NROOTS  2
END
TASK TCE ENERGY
```

Completely renormalized EOMCCSD(T) (CR-EOMCCSD(T)) calculations for the ozone molecule as described by the POL1 basis set. The CREOMSD(T) directive automatically initialize three-step procedure: (1) CCSD calculations; (2) EOMCCSD calculations; (3) non-iterative CR-EOMCCSD(T) corrections.

```
START TCE_CR_EOM_T_OZONE
ECHO
GEOMETRY UNITS BOHR
SYMMETRY C2V
 O   0.0000000000       0.0000000000       0.0000000000
 O   0.0000000000      -2.0473224350      -1.2595211660
 O   0.0000000000       2.0473224350      -1.2595211660
END
BASIS SPHERICAL
O    S
    10662.285000000      0.00079900
     1599.709700000      0.00615300
      364.725260000      0.03115700
      103.651790000      0.11559600
       33.905805000      0.30155200
O    S
       12.287469000      0.44487000
        4.756805000      0.24317200
O    S
        1.004271000      1.00000000
O    S
        0.300686000      1.00000000
O    S
        0.090030000      1.00000000
O    P
       34.856463000      0.01564800
        7.843131000      0.09819700
        2.306249000      0.30776800
        0.723164000      0.49247000
O    P
        0.214882000      1.00000000
O    P
        0.063850000      1.00000000
O    D
        2.306200000      0.20270000
        0.723200000      0.57910000
O    D
        0.214900000      0.78545000
        0.063900000      0.53387000
END
SCF
 THRESH 1.0e-10
 TOL2E 1.0e-10
 SINGLET
 RHF
END
TCE
 FREEZE ATOMIC
 CREOMSD(T)
 TILESIZE 20
 THRESH 1.0d-6
 NROOTS 2
END
TASK TCE ENERGY
```

The input for the active-space CR-EOMCCSD(T) calculations (the uracil molecule in the 6-31G* basis set). In this example, the model space is specified by defining the number of highest occupied orbitals (noact) and the number of lowest unoccupied orbitals (nuact) that will be considered as the active orbitals. In any type of the active-space CR-EOMCCSD(T) calculatoins based on the RHF and ROHF references more efficient versions of the orbital 4-index transformation can be invoked (i.e., `2emet 13` or `2emet 14`).

```
title "uracil-6-31-Gs-act"
echo
start uracil-6-31-Gs-act
memory stack 1000 mb heap 100 mb global 1000 mb noverify
geometry units angstrom
  N          .034130   -.986909    .000000
  N        -1.173397    .981920    .000000
  C        -1.218805   -.408164    .000000
  C         -.007302   1.702153    .000000
  C         1.196200   1.107045    .000000
  C         1.289085   -.345905    .000000
  O         2.310232   -.996874    .000000
  O        -2.257041  -1.026495    .000000
  H          .049329  -1.997961    .000000
  H        -2.070598   1.437050    .000000
  H         -.125651   2.776484    .000000
  H         2.111671   1.674079    .000000
end
basis cartesian
  * library 6-31G*
end
scf
  thresh 1.0e-10
  tol2e 1.0e-10
  singlet
  rhf
end
tce
  freeze atomic
  creom(t)ac
  oact 21
  uact 99
  tilesize 15
  thresh 1.0d-5
  2eorb
  2emet 13
  nroots 1
  symmetry
  targetsym a'
end
task tce energy
```

The active-space in the active-space CR-EOMCCSD(T) calculations can be alternatively specified by defining the energy window [emin_act,emax_act]. All orbitals with orbital energies falling into this widnow will considered as active (the active space in the following example is different from the one used in the previous example).

```
title "uracil-6-31-Gs-act"
echo
start uracil-6-31-Gs-act
memory stack 1000 mb heap 100 mb global 1000 mb noverify
geometry units angstrom
  N          .034130   -.986909    .000000
  N        -1.173397    .981920    .000000
  C        -1.218805   -.408164    .000000
  C         -.007302   1.702153    .000000
  C         1.196200   1.107045    .000000
  C         1.289085   -.345905    .000000
  O         2.310232   -.996874    .000000
  O        -2.257041  -1.026495    .000000
  H          .049329  -1.997961    .000000
  H        -2.070598   1.437050    .000000
  H         -.125651   2.776484    .000000
  H         2.111671   1.674079    .000000
end
basis cartesian
 * library 6-31G*
end
scf
 thresh 1.0e-10
 tol2e 1.0e-10
 singlet
 rhf
end
tce
 freeze atomic
 creom(t)ac
 emin_act -0.5
 emax_act 1.0
 tilesize 15
 thresh 1.0d-5
 2eorbe
 2emet 13
 nroots 1
 symmetry
 targetsym a'
end
task tce energy
```

The LR-CCSD(T) calculations for the glycine molecule in the aug-cc-pVTZ basis set. Option 2EORB is used in order to minimize memory requirements associated with the storage of two-electron integrals.

```
START TCE_LR_CCSD_T
ECHO
GEOMETRY UNITS BOHR
 O    -2.8770919486      1.5073755650      0.3989960497
 C    -0.9993929716      0.2223265108     -0.0939400216
 C     1.6330980507      1.1263991128     -0.7236778647
 O    -1.3167079358     -2.3304840070     -0.1955378962
 N     3.5887721300     -0.1900460352      0.6355723246
 H     1.7384347574      3.1922914768     -0.2011420479
 H     1.8051078216      0.9725472539     -2.8503867814
 H     3.3674278149     -2.0653924379      0.5211399625
 H     5.2887327108      0.3011058554     -0.0285088728
 H    -3.0501350657     -2.7557071585      0.2342441831
END
BASIS
 * library aug-cc-pVTZ
END
SCF
 THRESH 1.0e-10
 TOL2E 1.0e-10
 SINGLET
 RHF
END
TCE
 FREEZE ATOMIC
 2EORB
 TILESIZE 15
 LR-CCSD(T)
 THRESH 1.0d-7
END
TASK TCE ENERGY
```

The CCSD calculations for the triplet state of the C_{20} molecule. New algorithms for 4-index tranformation are used.

```
title "c20_cage"
echo
start c20_cage
memory stack 2320 mb heap 180 mb global 2000 mb noverify
geometry print xyz units bohr
  symmetry c2
  C      -0.761732  -1.112760   3.451966
  C       0.761732   1.112760   3.451966
  C       0.543308  -3.054565   2.168328
  C      -0.543308   3.054565   2.168328
  C       3.190553   0.632819   2.242986
  C      -3.190553  -0.632819   2.242986
  C       2.896910  -1.982251   1.260270
  C      -2.896910   1.982251   1.260270
  C      -0.951060  -3.770169   0.026589
  C       0.951060   3.770169   0.026589
  C       3.113776   2.128908   0.076756
  C      -3.113776  -2.128908   0.076756
  C       3.012003  -2.087494  -1.347695
  C      -3.012003   2.087494  -1.347695
  C       0.535910  -2.990532  -2.103427
  C      -0.535910   2.990532  -2.103427
  C       3.334106   0.574125  -2.322563
  C      -3.334106  -0.574125  -2.322563
  C      -0.764522  -1.081362  -3.453211
  C       0.764522   1.081362  -3.453211
end
basis spherical
 * library cc-pvtz
end
scf
  triplet
  rohf
  thresh 1.e-8
  maxiter 200
end
tce
  ccsd
  maxiter 60
  diis 5
  thresh 1.e-6
  2eorb
  2emet 3
  attilesize 40
  tilesize 30
  freeze atomic
end
task tce energy
```

11.6.7 TCE Response Properties

11.6.7.1 Introduction

Response properties can be calculated within the TCE. Ground-state dipole polarizabilities can be performed at the CCSD, CCSDT and CCSDTQ levels of theory. Neither CCSDT-LR nor CCSDTQ-LR are compiled by default. Like the rest of the TCE, properties can be calculated with RHF, UHF, ROHF and DFT reference wavefunctions.

Specific details for the implementation of CCSD-LR and CCSDT-LR can be found in the following papers:

- J. R. Hammond, M. Valiev, W. A. deJong and K. Kowalski, J. Phys. Chem. A, 111, 5492 (2007).
- J. R. Hammond, K. Kowalski and W. A. de Jong, J. Chem. Phys., 127, 144105 (2007).
- J. R. Hammond, W. A. de Jong and K. Kowalski , J. Chem. Phys., 128, 224102 (2008).

An appropriate background on coupled-cluster linear response (CC-LR) can be found in the references of those papers.

11.6.7.2 Performance

The coupled-cluster response codes were generated in the same manner as the rest of the TCE, thus all previous comments on performance apply here as well. The improved offsets available in the CCSD and EOM-CCSD codes is now also available in the CCSD-Λ and CCSD-LR codes. The bottleneck for CCSD-LR is the same as EOM-CCSD, likewise for CCSDT-LR and EOM-CCSDT. The CCSD-LR code has been tested on as many as 1024 processors for systems with more

than 2000 spin-orbitals, while the CCSDT-LR code has been run on as many as 1024 processors. The CCSDTQ-LR code is not particularly useful due to the extreme memory requirements of quadruples amplitudes, limited scalability and poor convergence in the CCSDTQ equations in general.

11.6.7.3 Input

The input commands for TCE response properties exclusively use set directives (seeSET) instead of TCE input block keywords. There are currently only three commands available:

```
set tce:lineresp <logical lineresp default: F>
set tce:afreq <double precision afreq(9) default: 0.0>
set tce:respaxis <logical respaxis(3) default: T T T>
```

The boolean variable lineresp invokes the linear response equations for the corresponding coupled-cluster method (only CCSD and CCSDT possess this feature) and evaluates the dipole polarizability. When lineresp is true, the Λ-equations will also be solved, so the dipole moment is also calculated. If no other options are set, the complete dipole polarizability tensor will be calculated at zero frequency (static). Up to nine real frequencies can be set; adding more should not crash the code but it will calculate meaningless quantities. If one desires to calculate more frequencies at one time, merely change the line `double precision afreq(9)` in `$NWCHEM_TOP/src/tce/include/tce.fh` appropriately and recompile.

The user can choose to calculate response amplitudes only for certain axis, either because of redundancy due to symmetry or because of memory limitations. The boolean vector of length three respaxis is used to determine whether or not a given set of response amplitudes are allocated, solved for, and used in the polarizability tensor evaluation. The logical variables represent the X, Y, Z axes, respectively. If respaxis is set to T F T, for example, the responses with respect to the X and Z dipoles will be calculated, and the four (three unique) tensor components will be evaluated. This feature is also useful for conserving memory. By calculating only one axis at a time, memory requirements can be reduced by 25% or more, depending on the number of DIIS vectors used. Reducing the number of DIIS vectors also reduces the memory requirements.

It is strongly advised that when calculating polarizabilities at high-frequencies, that user set the frequencies in increasing order, preferably starting with zero or other small value. This approach is computationally efficient (the initial guess for subsequent responses is the previously converged value) and mitigates starting from a zero vector for the response amplitudes.

11.6.7.4 Examples

This example runs in-core on a large workstation.

```
geometry units angstrom
 symmetry d2h
 C             0.000    1.390    0.000
 H             0.000    2.470    0.000
 C             1.204    0.695    0.000
 H             2.139    1.235    0.000
 C             0.000   -1.390    0.000
 H             0.000   -2.470    0.000
 C            -1.204   -0.695    0.000
 H            -2.139   -1.235    0.000
 C             1.204   -0.695    0.000
 H             2.139   -1.235    0.000
 C            -1.204    0.695    0.000
 H            -2.139    1.235    0.000
end
basis spherical
 * library aug-cc-pvdz
end
tce
 freeze atomic
 ccsd
 io ga
 2eorb
 tilesize 16
end
set tce:lineresp T
set tce:afreq 0.000 0.072
set tce:respaxis T T T
task tce energy
```

This is a relatively simple example for CCSDT-LR.

```
geometry units au
 symmetry c2v
 H 0      0       0
 F 0      0       1.7328795
end
basis spherical
 * library aug-cc-pvdz
end
tce
 ccsdt
 io ga
 2eorb
end
set tce:lineresp T
set tce:afreq 0.0 0.1 0.2 0.3 0.4
set tce:respaxis T F T
task tce energy
```

11.6.8 TCE Restart Capability

11.6.8.1 Overview

Check-pointing and restart are critical for computational chemistry applications of any scale, but particularly those done on supercomputers, or run for an extended period on workstations and clusters. The TCE supports parallel check-pointing and restart using the Shared Files (SF) library in the Global Arrays Tools. The SF library requires that the file system be accessible by every node, so reading and writing restart files can only be performed on a shared file system. For workstations, this condition is trivially met. Restart files must be persistent to be useful, so volatile file systems or those which are periodicly erased should not be used for check-pointing.

Restart is possible for all ground-state amplitudes (T, Λ and $T^{(1)}$ but not for excited-state amplitudes, as in an EOM-CC calculation. The latter functionality is under development.

Restart capability is useful in the following situations:

- The total run time is limited, as is the case for most supercomputing facilities.
- The system is volatile and jobs die randomly.
- When doing property calculations which require a large number of responses which cannot all be stored in-core simultaneously.

At the present time, restarting the amplitudes during a potential energy surface scan or numerical geometry optmization/frequency calculation is not advised due to the phase issue in the molecular orbital coefficients. If the phase changes, the amplitudes will no longer be a useful guess and may lead to nonsense results. Expert users may be able to use restart when the geometry varies using careful choices in the SCF input by using the rotate and lock options but this use of restart is not supported.

If SF encounters a failure during restart I/O, the job will fail. The capability to ignore a subset of failures, such as when saving the amplitudes prior to convergence, will be available in the future. This is useful on some large machines when the filesystem is being taxed by another job and may be appear unavailable at the moment a check-point write is attempted.

The performance of SF I/O for restart is excellent and the wall time for reading and writing integrals and amplitudes is negligible, even on a supercomputer (such systems have very fast parallel file systems in most cases). The only platform for which restart may cause I/O problems is BlueGene, due to ratio of compute to I/O nodes (64 on BlueGene/P).

11.6.8.2 Input

```
set tce:read_integrals <logical read_integrals default: F F F F F>
set tce:read_t <logical read_t default: F F F F>
set tce:read_l <logical read_l default: F F F F>
set tce:read_tr <logical read_tr default: F F F F>
set tce:save_integrals <logical save_integrals default: F F F F F>
set tce:save_t <logical save_t default: F F F F>
set tce:save_l <logical save_l default: F F F F>
set tce:save_tr <logical save_tr default: F F F F>
set tce:save_interval <integer save_interval default: 100000>
```

The boolean variables read_integrals and save_integrals control which integrals are read/saved. The first location is the 1-e integrals, the second is for the 2-e integrals, and the third is for dipole integrals. The fourth and fifth positions are reserved for quadrupole and octupole integrals but this functionality is not available. The read_t , read_l , read_tr , save_t , save_l and save_tr variables control the reading/saving of the T, Λ and $T^{(1)}$ (response) amplitudes. Restart control on the response amplitudes is implicitly controlled by the value of respaxis (see above). Requesting amplitudes that are beyond the scope of a given calculation, such as T_3 in a CCSD calculation, does not produce an error as these commands will never be processed.

Attempting to restart with a set of amplitudes without the corresponding integrals is ill-advised, due to the phase issue discussed above. For the same reason, one cannot save a subset of the integrals, so if it is even remotely possible that the dipole moment or polarizabilities will be desired for a given molecule, the dipole integrals should be saved as well. It is possible to save the dipole integrals without setting dipole in the TCE input block; setting save_integrals(3) true is sufficient for this to occur.

The save_interval variable controls the frequency with which amplitudes are saved. By default, the amplitudes are saved only when the iterative process has converged, meaning that if the iterations do not converge in less than the maximum, one must start the calculation again from scratch. The solution is to set save_interval to a smaller value, such as the number of DIIS cycles being used.

The user shall not change the tilesize when reading in saved amplitudes. The results of this are catastrophic and under no circumstance will this lead to physically meaningful results. Restart does not work for 2eorb and 2emet 9 ; no error will be produced but the results may be meaningless.

11.6.8.3 Examples

```
geometry units au
 symmetry c2v
 H 0      0       0
 F 0      0       1.7328795
end
basis spherical
 * library aug-cc-pvdz
end
tce
 ccsdt
 io ga
end
set tce:lineresp T
set tce:afreq 0.0 0.1 0.2 0.3 0.4
set tce:respaxis T F T
task tce energy
```

11.6.9 Maximizing performance

The following are recommended parameters for getting the best performance and efficiency for common methods on various hardware configurations. The optimal settings are far from extensible and it is extremely important that users take care in how they apply these recommendations. Testing a variety of settings on a simple example is recommended when optimal settings are desired. Nonetheless, a few guiding principles will improve the performance of TCE jobs markedly, including making otherwise impossible jobs possible.

11.6.9.1 Memory considerations

The default memory settings for NWChem are not optimal for TCE calculations. When 2 GB of memory is available per process, the following settings are close to optimal for CCSD jobs

```
memory stack 800 mb heap 100 mb global 1000 mb
```

for property jobs, which require more amplitudes to be stored, it is wise to favor the global allocation

```
memory stack 500 mb heap 100 mb global 1300 mb
```

If you get an error for ga_create during the iterative steps, reduce the number of DIIS vectors. If this error occurs during the four-index transformation (after d_v2 filesize appears) you need more GA space, a different 2emet, or more nodes.

The memory requirements for CCSD(T) are quite different because the triples are generated in local memory. The value of tilesize should not be larger than 30 in most cases and one should set something similar to the following

```
memory stack 1200 mb heap 100 mb global 600 mb
```

The local memory requires will be tilesize^N where N=4 for CCSD, N=6 for CCSD(T) and CCSDT, and N=8 for CCSDTQ. One should set tilesize to 16 or less for CCSDT and CCSDTQ, although symmetry will affect the local memory use significantly. The local memory usage of the CR-EOMCCSD(T) approach has recently been significantly reduced to the level of the CCSD(T) approach ($2*\text{tilesize}^6$).

11.6.9.2 Using OpenMP in TCE

TCE compute kernels are both floating-point and bandwidth intensive, hence are amenable to multithreading. However, not all TCE kernels support OpenMP, and therefore performance may be limited by Amdahl's law. Furthermore, Global Arrays communication is not yet thread-safe and must be called outside of threaded regions, potentially limiting performance. However, even partial OpenMP is likely to improve performance relative to idle cores, in the case where memory limitations or other considerations (see below for the case of Xeon Phi coprocessors) force the user to run NWChem on a subset of the available CPU cores.

Currently, OpenMP threading is available in the following kernels:

- BLAS (assuming the chosen library supports this).
- CCSD(T) and MRCCSD(T) triples kernels.

The development version of NWChem (post-6.6) supports OpenMP more kernels, including:

- 4- and 6-dimensional tensor transposes.
- Loop-driven CCSD tensor contractions.

In most cases, NWChem runs best on CPU-only systems without OpenMP threads. However, modest OpenMP has been found to improve performance of CCSD(T) jobs. We expect the benefits of OpenMP to be more visible with time, as NWChem achieves more complete coverage with OpenMP and as platforms evolve to have more and more cores per node.

11.6.9.3 How to run large CCSD/EOMCCSD calculations

When running large CCSD or EOMCCSD calculations for large systems (number of orbitals larger than 400) and using large number of cores it is recommended to switch to workflow based implementation of CCSD/EOMCCSD methods.

The original CCSD/EOMCCSD TCE implementations are aggregates of a large number of subroutines, which calculate either recursive intermediates or contributions to residual vector. The dimensionalities of the tensors involved in a given subroutine greatly impact the memory, computation, and communication characteristics of each subroutine, which can lead to pronounced problems with load balancing. For example, for the most computationally intensive part of the CCSD/EOMCCSD approaches associated with the inclusion of 4-particle integrals, the corresponding task pool (the number of tasks in a subroutine) can easily be 2 orders of magnitude larger than the task pool for subroutines calculating one-body intermediates. To address this problem and improve the scalability of the CCSD/EOMCCSD implementations, we exploited the dependencies exposed between the task pools into classes (C) characterized by a collective task pool. This was done in such a way as to ensure sufficient parallelism in each class while minimizing the total number of such classes. This procedure enabled us to reduce the number of synchronization steps from nearly 80, in the EOMCCSD case, down to 4. Optimized versions of the CCSD/EOMCCSD codes are enabled once the

```
set tce:nts T
```

directive is used in the input file. Compared to the original CCSD/EOMCCSD implementations the new approaches requires more global memory. The new CCSD/EOMCCSD implementations provides significant improvements in the parallel performance and average time per iteration.

References:

H.S. Hu, K. Bhaskaran-Nair, E. Apra, N. Govind, K. Kowalski, J. Phys. Chem. A 118, 9087 (2014).

K. Kowalski, S. Krishnamoorthy, R.M. Olson, V. Tipparaju, E. Apra, K. Kowalski, High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference 1 (2011).

11.6.9.4 SCF options

For parallel jobs on clusters with poor disk performance on the filesystem used for scratch_dir, it is a good idea to disable disk IO during the SCF stage of the calculation. This is done by adding semidirect memsize N filesize 0, where N is 80% of the stack memory divided by 8, as the value in this directive is the number of dwords, rather than bytes. With these settings, if the aggregate memory is sufficient to store the integrals, the SCF performance will be excellent, and it will be better than if direct is set in the SCF input block. If scratch_dir is set to a local disk, then one should use as much disk as is permissible, controlled by the value of filesize. On many high-performance computers, filling up the local scratch disk will crash the node, so one cannot be careless with these settings. In addition, on many such machines, the shared file system performance is better than that of the local disk (this is true for many NERSC systems).

11.6.9.5 Convergence criteria

It makes no sense to converge a calculation to a precision not relevant to experiment. However, the relationship between convergence criteria and calculated quantities is not fully known for some properties. For example, the effect of the convergence criteria on the polarizability is significant in some cases. In the case of CN, convergence of 10^{-11} is necessary to resolve the polarizability tensor components to 10^{-2}. However, for many systems 10^{-7} convergence is sufficient to get accurate results for all properties. It is important to calibrate the effect of convergence on property calculations, particularly for open-shell and post-CCSD methods, on a modest basis set before relaxing the convergence criteria too much.

11.6.9.6 IO schemes and integral transformation algorithms

The effect on memory use of using the 2eorb keyword is huge. However, this option can only be used with IO=GA and an RHF/ROHF reference. There are a number of choices for the integral transformation algorithm when using spin-free integrals. The fastest algorithm is 2EMET=5, but significant disk IO is required for this algorithm. One must set permanent_dir to a fast, shared file system for this algorithm to work. If disk performance is not good, one should use either 2EMET=3 or 2EMET=4 depending on how much memory is available. If one sees a ga_create error with 2EMET=3, then switch to algorithm 4 and add split 8 to the TCE input block.

11.6.10 Using coprocessor architectures

11.6.10.1 CCSD(T) and MRCCSD(T) implementations for Intel MIC architectures

This option is no longer available from version 7.0.0

NWChem 6.5 and 6.6 offer the possibility of using Intel Xeon Phi hardware to perform the most computationally intensive part of the CCSD(T) and MRCCSD(T) (only in NWChem 6.6) calculations (non-iterative triples corrections). The form of input is the same as used in standard TCE CCSD(T) and MRCCSD(T) runs. To enable new implementations please follow compilation directives described below.

Required for compilation: Intel Composer XE version 14.0.3 (or later versions)

Environmental variables required for compilation:

```
% setenv USE_OPENMP 1

% setenv USE_OFFLOAD 1
```

When using MKL and Intel Composer XE version 14 (or later versions), please use the following settings

```
% setenv BLASOPT   "-mkl -openmp   -lpthread -lm"
% setenv SCALAPACK "-mkl -openmp -lmkl_scalapack_ilp64 -lmkl_blacs_intelmpi_ilp64 -lpthread -lm"
```

The command require for compilation is

```
make FC=ifort
```

From our experience using the CCSD(T) and MRCCSD(T) TCE modules, we have determined that the optimal configuration is to use a single Global Arrays ranks for offloading work to each Xeon Phi card.

On the EMSL cascade system, each node is equipped with two coprocessors, and NWChem can allocate one GA ranks per coprocessor. In the job scripts, we recommend spawning just 6 GA ranks for each node, instead of 16 (number that would match the number of physical cores). Therefore, 2 out 6 GA ranks assigned to a particular compute node will offload to the coprocessors, while the remaining 6 cores while be used for traditional CPU processing duties. Since during offload the host core is idle, we can double the number of OpenMP threads for the host (`OMP_NUM_THREADS=4`) in order to fill the idle core with work from another GA rank (4 process with 4 threads each will total 16 threads on each node).

NWChem itself automatically detects the available coprocessors in the system and properly partitions them for optimal use, therefore no action is required other than specifying the number of processes on each node (using the appropriate mpirun/mpiexec options) and setting the value of `OMP_NUM_THREADS` as in the example above.

Environmental variables useful at run-time:

`OMP_NUM_THREADS` is needed for the thread-level parallelization on the Xeon CPU hosts

```
% setenv OMP_NUM_THREADS 4
```

MIC_USE_2MB_BUFFER greatly improve communication between host and Xeon Phi card

```
% setenv MIC_USE_2MB_BUFFER 16K
```

11.6.10.2 CCSD(T) method with CUDA

NWChem 6.3 offers a possibility of using GPU accelerators to perform the most computationally intensive part of the CCSD(T) calculations (non-iterative triples corrections). To enable this option one has to enable compilation options described below and add the `cuda n` directive to the tce block of input, where `n` refers to number of CUDA devices per node.

geometry/basis set specifications

```
tce
 io ga
 freeze atomic
 thresh 1.0d-6
 tilesize 15
 ccsd(t)
 cuda 1
end
```

In the example above the number of CUDA devises is set equal to 1, which means that user will use 1 GPU per node.

To enable the compilation of CUDA code one has to set the follwoing variables before the compilation of NWChem.

```
export TCE_CUDA=Y
export CUDA_LIBS="-L<Your Path to cuda>/lib64 -L<Your Path to cuda>/lib -lcudart"
export CUDA_FLAGS="-arch  <Your Cuda architecture>"
export CUDA_INCLUDE="-I. -I<Your Path to cuda>/include"
```

For example:

```
export TCE_CUDA=Y
export CUDA_LIBS="-L/usr/local/cuda-5.0/lib64 -L/usr/local/cuda-5.0/lib -lcudart"
export CUDA_FLAGS="-arch sm_20 "
export CUDA_INCLUDE="-I. -I/usr/local/cuda-5.0/include"
```

In addition the code needs to be compiled with the following make command

```
make FC=<fortran compiler> CUDA=nvcc
```

Before running production style calculations we strongly suggest the users to perform QA test from the /nwchem/QA/tests/tce_cuda directory. A full example of a TCE CUDA input file is given below:

```
start tce_cuda
echo
memory stack 1000 mb heap 100 mb global 500 mb verify
geometry units bohr
  O     0.00000000     0.00000000     0.22138519
  H     0.00000000    -1.43013023    -0.88554075
  H     0.00000000     1.43013023    -0.88554075
end
basis spherical
  H library cc-pVDZ
  O library cc-pVDZ
end
charge 0
scf
  thresh 1.0e-10
  tol2e 1.0e-10
  singlet
  rhf
end
tce
  ccsd(t)
  io ga
  cuda 1
  tilesize 18
end
task tce energy
```

11.7 MP2

11.7.1 Overview

There are (at least) three algorithms within NWChem that compute the Møller-Plesset (or many-body) perturbation theory second-order correction[1] to the Hartree-Fock energy (MP2). They vary in capability, the size of system that can be treated and use of other approximations

- Semi-direct – this is recommended for most large applications (up to about 2800 basis functions), especially on machines with significant disk I/O capability. Partially transformed integrals are stored on disk, multi-passing as necessary. RHF and UHF references may be treated including computation of analytic derivatives. This is selected by specifying mp2 on the task directive, e.g.

```
TASK MP2
```

- Fully-direct[2] – this is of utility if only limited I/O resources are available (up to about 2800 functions). Only RHF references and energies are available. This is selected by specifying `direct_mp2` on the task directive, e.g.

```
TASK DIRECT_MP2
```

- Resolution of the identity (RI) approximation MP2 (RI-MP2)[3] – this uses the RI approximation and is therefore only exact in the limit of a complete fitting basis. However, with some care, high accuracy may be obtained with relatively modest fitting basis sets. An RI-MP2 calculation can cost over 40 times less than the corresponding exact MP2 calculation. RHF and UHF references with only energies are available. This is selected by specifying rimp2 on the task directive, e.g.,

```
TASK RIMP2
```

All three MP2 tasks share the same input block.

```
MP2
  [FREEZE [[core] (atomic || <integer nfzc default 0>)] \
        [virtual <integer nfzv default 0>]]
  [TIGHT]
  [PRINT]
  [NOPRINT]
  [VECTORS <string filename default scf-output-vectors> \
    [swap [(alpha||beta)] <integer pair-list>] ]
  [RIAPPROX <string riapprox default V>]
  [FILE3C <string filename default $file_prefix$.mo3cint>
  [SCRATCHDISK <integer>]
END
```

11.7.2 FREEZE: Freezing orbitals

All MP2 modules support frozen core orbitals, however, only the direct MP2 and RI-MP2 modules support frozen virtual orbitals.

By default, no orbitals are frozen. The atomic keyword causes orbitals to be frozen according to the rules in the table below. Note that no orbitals are frozen on atoms on which the nuclear charge has been modified either by the user or due to the presence of an ECP. The actual input would be

```
freeze atomic
```

For example, in a calculation on $Si(OH)_2$, by default the lowest seven orbitals would be frozen (the oxygen 1*s*, and the silicon 1*s*, 2*s* and 2*p*).

11.7.2.1 Number of orbitals considered “core” in the “freeze by atoms” algorithm

Period	Elements	Core Orbitals	Number of Core
0	H - He	-	0
1	Li - Ne	*1s*	1
2	Na - Ar	*1s2s2p*	5
3	K - Kr	*1s2s2p3s3p*	9
4	Rb - Xe	*1s2s2p3s3p4s3d4p*	18
5	Cs - Rn	*1s2s2p3s3p4s3d4p5s4d5p*	27
6	Fr - Lr	*1s2s2p3s3p4s3d4p5s4d5p6s4f5d6p*	43

Caution: The rule for freezing orbitals “by atoms” are rather unsophisticated since the number of orbitals to be frozen is computed from the table above by summing the number of core orbitals in each atom present. Therefore, the corresponding number of lowest-energy orbitals are frozen. If for some reason the actual core orbitals are not the lowest lying, then correct results will not be obtained. It is likely that special attention should be paid to systems including third- and higher- period atoms.

The user may also specify the number of orbitals to be frozen by atom. Following the $Si(OH)_2$ example, the user could specify

```
freeze atomic O 1 Si 3
```

In this case only the lowest four orbitals would be frozen. If the user does not specify the orbitals by atom, the rules default to values reported in the Table above.

Caution: The system does not check for a valid number of orbitals per atom. If the user specifies to freeze more orbitals then are available for the atom, the system will not catch the error. The user must specify a logical number of orbitals to be frozen for the atom.

The FREEZE directive may also be used to specify the number of core orbitals to freeze. For instance, to freeze the first 10 orbitals

```
freeze 10
```

or equivalently, using the optional keyword core

```
freeze core 10
```

Again, note that if the 10 orbitals to be frozen do not correspond to the first 10 orbitals, then the swap keyword of the VECTORS directive must be used to order the input orbitals correctly (MO vectors).

To freeze the highest virtual orbitals, use the virtual keyword. For instance, to freeze the top 5 virtuals

```
freeze virtual 5
```

Again, note that this only works for the direct-MP2 and RI-MP2 energy codes.

11.7.3 TIGHT: Increased precision

The TIGHT directive can be used to increase the precision in the MP2 energy and gradients.

By default the MP2 gradient package should compute energies accurate to better than a micro-Hartree, and gradients accurate to about five decimal places (atomic units). However, if there is significant linear dependence in the basis set the precision might not be this good. Also, for computing very accurate geometries or numerical frequencies, greater precision may be desirable.

This option increases the precision to which both the SCF (from 10^{6} to 10^{-8} and CPHF 10^{-4} to \$$10^{-6}$ are solved, and also tightens thresholds for computation of the AO and MO integrals (from 10^{-9} to 10^{-11} within the MP2 code.

11.7.4 SCRATCHDISK: Limiting I/O usage

This directive - used only in the semi-direct algorithm - allows to limit the per process disk usage. Mandatory argument for this keyword is the maximum number of MBytes. For example, the following input line

```
scratchdisk 512
```

puts an upper limit of 512 MBytes to the semi-direct MP2 usage of disk (again, on a per process base).

11.7.5 PRINT and NOPRINT

The standard print control options are recognized. The list of recognized names are given in the table below.

Item	Print Level	Description
RI-MP2		
“2/3 ints”	debug	Partial 3-center integrals
“3c ints”	debug	MO 3-center integrals
“4c ints b”	debug	“B” matrix with approx. 4c integrals
“4c ints”	debug	Approximate 4-center integrals
“amplitudes”	debug	“B” matrix with denominators
“basis”	high	
“fit xf”	debug	Transformation for fitting basis
“geombas”	debug	Detailed basis map info
“geometry”	high	
“information”	low	General information about calc.
“integral i/o”	high	File size information
“mo ints”	debug	
“pair energies”	debug	(working only in direct_mp2)
“partial pair energies”	debug	Pair energy matrix each time it is updated
“progress reports”	default	Report completion of time-consuming steps
“reference”	high	Details about reference wavefunction
“warnings”	low	Non-fatal warnings

11.7.6 VECTORS: MO vectors

All of the (supported) MP2 modules require use of converged canonical SCF (RHF or UHF) orbitals for correct results. The vectors are by default obtained from the preceding SCF calculation, but it is possible to specify a different source using the VECTORS directive. For instance, to obtain vectors from the file /tmp/h2o.movecs, use the directive

```
vectors /tmp/h2o.movecs
```

As noted above (FREEZE) if the SCF orbitals are not in the correct order, it is necessary to permute the input orbitals using the swap keyword of the VECTORS directive. For instance, if it is desired to freeze a total six orbitals corresponding to the SCF orbitals 1-5, and 7, it is necessary to swap orbital 7 into the 6th position. This is accomplished by

```
vectors swap 6 7
```

The swap capability is examined in more detail inInput/output of MO vectors.

11.7.7 RI-MP2 fitting basis

The RI-MP2 method requires a fitting basis, which must be specified with the name “ri-mp2 basis” (see Basis). For instance,

```
basis "ri-mp2 basis"
  O s; 10000.0 1
  O s;  1000.0 1
  O s;   100.0 1
  ...
end
```

Alternatively, using a standard capability of basis sets (Basis) another named basis may be associated with the fitting basis. For instance, the following input specifies a basis with the name “small fitting basis” and then defines this to be the “ri-mp2 basis”.

```
basis "small fitting basis"
  H s; 10    1
  H s;  3    1
  H s;  1    1
  H s;  0.1  1
  H s;  0.01 1
end


set "ri-mp2 basis" "small fitting basis"
```

11.7.8 FILE3C: RI-MP2 3-center integral filename

The default name for the file used to store the transformed 3-center integrals is “file_prefix.mo3cint” in the scratch directory. This may be overridden using the FILE3C directive. For instance, to specify the file /scratch/h2o.3c, use this directive

```
file3c /scratch/h2o.3c
```

11.7.9 RIAPPROX: RI-MP2 Approximation

The type of RI approximation used in the RI-MP2 calculation is controlled by means of the RIAPPROX directive. The two possible values are V and SVS (case sensitive), which correspond to the approximations with the same names described by Vahtras et al.[4]. The default is V.

11.7.10 Advanced options for RI-MP2

These options, which functioned at the time of writing, are not currently supported.

11.7.10.1 Control of linear dependence

Construction of the RI fit requires the inversion of a matrix of fitting basis integrals which is carried out via diagonalization. If the fitting basis includes near linear dependencies, there will be small eigenvalues which can ultimately lead to non-physical RI-MP2 correlation energies. Eigenvectors of the fitting matrix are discarded if the corresponding eigenvalue is less than $_{\text{min eval}}$ which defaults to 10^{-8}. This parameter may be changed by setting the a parameter in the database. For instance, to set it to 10^{-10}

```
set "mp2:fit min eval" 1e-10
```

11.7.10.2 Reference Spin Mapping for RI-MP2 Calculations

The user has the option of specifying that the RI-MP2 calculations are to be done with variations of the SCF reference wavefunction. This is accomplished with a SET directive of the form,

```
set "mp2:reference spin mapping" <integer array default 0>
```

Each element specified for array is the SCF spin case to be used for the corresponding spin case of the correlated calculation. The number of elements set determines the overall type of correlated calculation to be performed. The default is to use the unadulterated SCF reference wavefunction.

For example, to perform a spin-unrestricted calculation (two elements) using the alpha spin orbitals (spin case 1) from the reference for both of the correlated reference spin cases, the SET directive would be as follows,

```
set "mp2:reference spin mapping" 1 1
```

The SCF calculation to produce the reference wavefunction could be either RHF or UHF in this case.

The SET directive for a similar case, but this time using the beta-spin SCF orbitals for both correlated spin cases, is as follows,

```
set "mp2:reference spin mapping" 2 2
```

The SCF reference calculation must be UHF in this case.

The SET directive for a spin-restricted calculation (one element) from the beta-spin SCF orbitals using this option is as follows,

```
set "mp2:reference spin mapping" 2
```

The SET directive for a spin-unrestricted calculation with the spins flipped from the original SCF reference wavefunction is as follows,

```
set "mp2:reference spin mapping" 2 1
```

11.7.10.3 Batch Sizes for the RI-MP2 Calculation

The user can control the size of each batch in the transformation and energy evaluation in the MP2 calculation, and consequently the memory requirements and number of passes required. This is done using two SET directives of the following form,

```
set "mp2:transformation batch size" <integer size default -1>
set "mp2:energy batch size" <integer isize jsize default -1 -1>
```

The default is for the code to determine the batch size based on the available memory. Should there be problems with the program-determined batch sizes, these variables allow the user to override them. The program will always use the smaller of the user's value of these entries and the internally computed batch size.

The transformation batch size computed in the code is the number of occupied orbitals in the *(occ vir|fit)* three-center integrals to be produced at a time. If this entry is less than the number of occupied orbitals in the system, the transformation will require multiple passes through the two-electron integrals. The memory requirements of this stage are two global arrays of dimension *batch size x vir x fit* with the "fit" dimension distributed across all processors (on shell-block boundaries). The compromise here is memory space versus multiple integral evaluations.

The energy evaluation batch sizes are computed in the code from the number of occupied orbitals in the two sets of three-center integrals to be multiplied together to produce a matrix of approximate four-center integrals. Two blocks of integrals of dimension *(batch isize x vir)* and *(batch jsize x vir)* by fit are read in from disk and multiplied together to produce *batch isize batch jsize vir^2* approximate integrals. The compromise here is performance of the distributed matrix multiplication (which requires large matrices) versus memory space.

11.7.10.4 Energy Memory Allocation Mode: RI-MP2 Calculation

The user must choose a strategy for the memory allocation in the energy evaluation phase of the RI-MP2 calculation, either by minimizing the amount of I/O, or minimizing the amount of computation. This can be accomplished using a SET directive of the form,

```
set "mp2:energy mem minimize" <string mem_opt default I>
```

A value of I entered for the string `mem_opt` means that a strategy to minimize I/O will be employed. A value of C tells the code to use a strategy that minimizes computation.

When the option to minimize I/O is selected, the block sizes are made as large as possible so that the total number of passes through the integral files is as small as possible. When the option to minimize computation is selected, the blocks are chosen as close to square as possible so that permutational symmetry in the energy evaluation can be used most effectively.

11.7.10.5 Local Memory Usage in Three-Center Transformation

For most applications, the code will be able to size the blocks without help from the user. Therefore, it is unlikely that users will have any reason to specify values for these entries except when doing very particular performance measurements.

The size of `xf3ci:AO 1 batch size` is the most important of the three, in terms of the effect on performance.

Local memory usage in the first two steps of the transformation is controlled in the RI-MP2 calculation using the following SET directives,

```
set "xf3ci:AO 1 batch size" <integer max>
set "xf3ci:AO 2 batch size" <integer max>
set "xf3ci:fit batch size" <integer max>
```

The size of the local arrays determines the sizes of the two matrix multiplications. These entries set limits on the size of blocks to be used in each index. The listing above is in order of importance of the parameters to performance, with xf3ci:AO 1 batch size being most important.

Note that these entries are only upper bounds and that the program will size the blocks according to what it determines as the best usage of the available local memory. The absolute maximum for a block size is the number of functions in the AO basis, or the number of fitting basis functions on a node. The absolute minimum value for block size is the size of the largest shell in the appropriate basis. Batch size entries specified for max that are larger than these limits are automatically reset to an appropriate value.

11.7.11 One-electron properties and natural orbitals

If an MP2 energy gradient is computed, all contributions are available to form the MP2 linear-response density. This is the density that when contracted with any spin-free, one-electron operator yields the associated property defined as the derivative of the energy. Thus, the reported MP2 dipole moment is the derivative of the energy w.r.t. an external electric field and is not the expectation value of the operator over the wavefunction. It has been shown that evaluating the MP2 density through a derivative provides more accurate results, presumably because this matches the way experiments probe the electron density more closely[raghavachari1981][567].

Only dipole moments are printed by the MP2 gradient code, but natural orbitals are produced and stored in the permanent directory with a file extension of “.mp2nos”. These may be fed into the property package to compute more general properties as in the following example.

```
start h2o
geometry
  O      2.15950      0.88132      0.00000
  H      3.12950      0.88132      0.00000
  H      1.83617      0.89369     -0.91444
end

basis spherical
 * library aug-cc-pVDZ
end

mp2
 freeze atomic
end

task mp2 gradient

property
 vectors  h2o.mp2nos
 mulliken
end

task mp2 property
```

Note that the MP2 linear response density matrix is not necessarily positive definite so it is not unusual to see a few small negative natural orbital occupation numbers. Significant negative occupation numbers have been argued to be a sign that the system might be near degenerate[8].

11.7.12 SCS-MP2: Spin-Component Scaled MP2

Each MP2 output contains the calculation of the SCS-MP2 correlation energies as suggested by S.Grimme[9]

The SCS keyword is only required for gradients calculations:

```
MP2
  [SCS]
END
```

Scaling factors for the two components (parallel and opposite spin) can be defined by using the keywords FSS (same spin factor) and FOS (opposite spin factor):

```
mp2
  scs
  fss   1.13
  fos   0.56
end
```

Default values are FSS=0.333333333, FOS=1.2 for MP2, and FSS=1.13, FOS=1.27 for CCSD.

11.7.13 References

1. Møller, Chr.; Plesset, M. S. Note on an Approximation Treatment for Many-Electron Systems. *Physical Review* **1934**, *46* (7), 618–622. https://doi.org/10.1103/PhysRev.46.618
2. Wong, A. T.; Harrison, R. J.; Rendell, A. P. Parallel Direct Four-Index Transformations. *Theoretica Chimica Acta* **1996**, *93* (6), 317–331. https://doi.org/10.1007/BF01129213
3. Bernholdt, D. E.; Harrison, R. J. Large-Scale Correlated Electronic Structure Calculations: The RI-MP2 Method on Parallel Computers. *Chemical Physics Letters* **1996**, *250* (5-6), 477–484. https://doi.org/10.1016/0009-2614(96)00054-1
4. Vahtras, O.; Almlöf, J.; Feyereisen, M. W. Integral Approximations for LCAO-SCF Calculations. *Chemical Physics Letters* **1993**, *213* (5-6), 514–518. https://doi.org/10.1016/0009-2614(93)89151-7
5. Diercksen, G. H. F.; Roos, B. O.; Sadlej, A. J. Legitimate Calculation of First-Order Molecular Properties in the Case of Limited CI Functions. Dipole Moments. *Chemical Physics* **1981**, *59* (1-2), 29–39. https://doi.org/10.1016/0301-0104(81)80082-1
6. Rice, J. E.; Amos, R. D. On the Efficient Evaluation of Analytic Energy Gradients. *Chemical Physics Letters* **1985**, *122* (6), 585–590. https://doi.org/10.1016/0009-2614(85)87275-4
7. Wiberg, K. B.; Hadad, C. M.; LePage, T. J.; Breneman, C. M.; Frisch, M. J. Analysis of the Effect of Electron Correlation on Charge Density

Distributions. *The Journal of Physical Chemistry* **1992**, *96* (2), 671–679. https://doi.org/10.1021/j100181a030

8. Gordon, M. S.; Schmidt, M. W.; Chaban, G. M.; Glaesemann, K. R.; Stevens, W. J.; Gonzalez, C. A Natural Orbital Diagnostic for Multiconfigurational Character in Correlated Wave Functions. *The Journal of Chemical Physics* **1999**, *110* (9), 4199–4207. https://doi.org/10.1063/1.478301
9. Grimme, S. Improved Second-Order Møller-Plesset Perturbation Theory by Separate Scaling of Parallel- and Antiparallel-Spin Pair Correlation Energies. *The Journal of Chemical Physics* **2003**, *118* (20), 9095–9102. https://doi.org/10.1063/1.1569242

11.8 Coupled Cluster Calculations

11.8.1 Overview

The NWChem coupled cluster energy module is primarily the work of Alistair Rendell and Rika Kobayashi[1,2], with contributions from Bert de Jong, David Bernholdt and Edoardo Aprà[3].

The coupled cluster code can perform calculations with full iterative treatment of single and double excitations and non-iterative inclusion of triple excitation effects. It is presently limited to closed-shell (RHF) references.

Note that symmetry is not used within most of the CCSD(T) code. This can have a profound impact on performance since the speed-up from symmetry is roughly the square of the number of irreducible representations. In the absence of symmetry, the performance of this code is competitive with other programs.

The operation of the coupled cluster code is controlled by the input block

```
CCSD
  [MAXITER <integer maxiter default 20>]
  [THRESH  <real thresh default 1e-6>]
  [TOL2E <real tol2e default min(1e-12 , 0.01**`thresh`*`)>]
  [DIISBAS  <integer diisbas default 5>]
  [FREEZE [[core] (atomic || <integer nfzc default 0>)] \
       [virtual <integer nfzv default 0>]]
  [NODISK]
  [IPRT  <integer IPRT default 0>]
  [PRINT ...]
  [NOPRINT ...]
END
```

Note that the keyword CCSD is used for the input block regardless of the actual level of theory desired (specified with the TASK directive). The following directives are recognized within the CCSD group.

11.8.2 MAXITER – Maximum number of iterations

The maximum number of iterations is set to 20 by default. This should be quite enough for most calculations, although particularly troublesome cases may require more.

```
MAXITER  <integer maxiter default 20>
```

11.8.3 THRESH – Convergence threshold

Controls the convergence threshold for the iterative part of the calculation. Both the RMS error in the amplitudes and the change in energy must be less than thresh.

```
THRESH  <real thresh default 1e-6>
```

11.8.4 TOL2E – integral screening threshold

```
TOL2E <real tol2e default min(1e-12, 0.01*thresh)>
```

The variable tol2e is used in determining the integral screening threshold for the evaluation of the energy and related

quantities.

CAUTION! At the present time, the tol2e parameter only affects the three- and four-virtual contributions, and the triples, all of which are done “on the fly”. The transformations used for the other parts of the code currently have a hard-wired threshold of 10^{-12}. The default for tol2e is set to match this, and since user input can only make the threshold smaller, setting this parameter can only make calculations take longer.

11.8.5 DIISBAS – DIIS subspace dimension

Specifies the maximum size of the subspace used in DIIS convergence acceleration. Note that DIIS requires the amplitudes and errors be stored for each iteration in the subspace. Obviously this can significantly increase memory requirements, and could force the user to reduce DIISBAS for large calculations.

Measures to alleviate this problem, including more compact storage of the quantities involved, and the possibility of disk storage are being considered, but have not yet been implemented.

```
DIISBAS  <integer diisbas default 5>
```

11.8.6 FREEZE – Freezing orbitals

```
[FREEZE [[core] (atomic || <integer nfzc default 0>)] \
    [virtual <integer nfzv default 0>]]
```

This directive is identical to that used in the MP2 module.

11.8.7 NODISK – On-the-fly computation of integrals

The CCSD modules by default computes once and stores on disk the integrals. To avoid this kind of I/O operations, specify the keyword NODISK

11.8.8 IPRT – Debug printing

This directive controls the level of output from the code, mostly to facilitate debugging and the like. The larger the value, the more output printed. From looking at the source code, the interesting values seem to be IPRT > 5, 10, and 50.

```
IPRT  <integer IPRT default 0>
```

11.8.9 PRINT and NOPRINT

The coupled cluster module supports the standard NWChem print control keywords, although very little in the code is actually hooked into this mechanism yet.

Item	Print Level	Description
“reference”	high	Wavefunction information
“guess pair energies”	debug	MP2 pair energies
“byproduct energies”	default	Intermediate energies
“term debugging switches”	debug	Switches for individual terms

11.8.10 Methods (Tasks) Recognized

Currently available methods are

- CCSD - Full iterative inclusion of single and double excitations
- CCSD+T(CCSD) - The fourth order triples contribution computed with converged singles and doubles amplitudes
- CCSD(T) - The linearized triples approximation due to Raghavachari.

The calculation is invoked using the TASK directive, so to perform a CCSD+T(CCSD) calculation, for example, the input file should include the directive

```
TASK CCSD+T(CCSD)
```

Lower-level results which come as by-products (such as MP3/MP4) of the requested calculation are generally also printed in the output file and stored on the run-time database, but the method specified in the TASK directive is considered the primary result.

11.8.11 Debugging and Development Aids

The information in this section is intended for use by experts (both with the methodology and with the code), primarily for debugging and development work. Messing with stuff in listed in this section will probably make your calculation quantitatively wrong! Consider yourself warned!

11.8.11.1 Switching On and Off Terms

The /DEBUG/ common block contains a number of arrays which control the calculation of particular terms in the program. These are 15-element integer arrays (although from the code only a few elements actually effect anything) which can be set from the input deck. See the code for details of how the arrays are interpreted.

Printing of this data at run-time is controlled by the “term debugging switches” print option. The values are checked against the defaults at run-time and a warning is printed to draw attention to the fact that the calculation does not correspond precisely to the requested method.

```
DOA  <integer array default 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2>
DOB  <integer array default 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2>
DOG  <integer array default 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1>
DOH  <integer array default 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1>
DOJK <integer array default 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2>
DOS  <integer array default 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1>
DOD  <integer array default 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1>
```

11.8.12 Alternative Implementations of Triples

There are four customized versions of the CCSD(T) triples driver that may improve performance on some architectures. These are not the default implementation and are not tested regularly. The burden is on the user to evaluate their correctness in comparison to the default triples driver. The triples driver only affects how the (T) energy contribution is evaluated; the CCSD code is the same in all cases.

All of the non-standard triples drivers are activated using RTDB set directives, which are specified outside of the CCSD input block.

11.8.12.1 Nonblocking

The nonblocking variant of the triples driver uses nonblocking Global Arrays get operations. It may improve communication overlap at large node code, provided that nonblocking communication makes asynchronous progress.

```
set ccsd:use_trpdrv_nb T
```

11.8.12.2 OpenMP

As of November 2016, the development version of semidirect CCSD(T) uses OpenMP extensively. The OpenMP variant of the triples driver includes OpenMP threaded kernels and attempts to run multiple DGEMM calls simultaneously. The CCSD iteration uses OpenMP threading in kernels with a relatively small number of parallel regions. It also uses nonblocking Global Arrays get operations.

```
set ccsd:use_ccsd_omp T
set ccsd:use_trpdrv_omp T
```

If one runs with only the (T) portion of the code using threads, the CCSD code will run slower when using fewer cores. Thus, it may be prudent to run the CCSD portion with a larger number of processes and then run a second job for (T) that restarts the computation on a smaller number of processes and a larger number of threads.

Preliminary evaluation of this implementation indicates that a small number of threads (2 to 4) is optimal, with the assumption that single-threaded execution can utilize all of the cores. It is expected that nodes with a large number of cores may not be able to support process-only parallelism due to memory-capacity constraints, in which case the OpenMP implementation allows the user to make use of more cores than otherwise possible.

Because of the extensive refactoring of the code to maximize OpenMP performance and the intrinsic non-associativity of floating-point arithmetic, the OpenMP variant may not produce the exact same answer as the default one. If there is concern about the numerical fidelity of results, a more stringent numerical threshold for the CCSD equations may be required.

11.8.12.3 Offload

The offload variant of the triples driver supports Intel Xeon Phi coprocessors (Knights Corner family), in addition to the aforementioned OpenMP and nonblocking features. This implementation has not been tested extensively and a recommendation concerning the right number of processes and threads is not available.

```
set ccsd:use_trpdrv_offload T
```

11.8.13 References

1. Rendell, A.P., Lee, T.J., Komornicki, A., and Wilson, S. (1992) “Evaluation of the contribution from triply excited intermediates to the fourth-order perturbation theory energy on Intel distributed memory supercomputers”, *Theor. Chem. Acc.*, **84**, 271-287, doi: 10.1007/BF01113267
2. Kobayashi, R. and Rendell, A.P. (1997) “A direct coupled cluster algorithm for massively parallel computers”, *Chem. Phys. Lett.*, **265**, 1-11, doi: 10.1016/S0009-2614(96)01387-5
3. Aprà, E., Harrison, R.J., de Jong, W.A., Rendell, A.P., Tipparaju, V. and Xantheas, S.S. (2009) “Liquid Water: Obtaining the Right Answer for the Right Reasons”, *Proc. SC‘09*, doi: 10.1145/1654059.1654127

11.9 MCSCF

11.9.1 Overview

The NWChem multiconfiguration SCF (MCSCF) module can currently perform complete active space SCF (CASSCF) calculations with at most 20 active orbitals and about 500 basis functions.

```
MCSCF
  STATE <string state>
  ACTIVE <integer nactive>
  ACTELEC <integer nactelec>
  MULTIPLICITY <integer multiplicity>
  [SYMMETRY <integer symmetry default 1>]
  [VECTORS [[input] <string input_file default file_prefix.movecs>]
        [swap <integer vec1 vec2> ...] \
        [output <string output_file default input_file>] \
        [lock]
  [HESSIAN (exact||onel)]
  [MAXITER <integer maxiter default 20>]
  [THRESH  <real thresh default 1.0e-4>]
  [TOL2E <real tol2e default 1.0e-9>]
  [LEVEL <real shift default 0.1d0>]
END
```

Note that the ACTIVE , ACTELEC , and MULTIPLICITY directives are required. The symmetry and multiplicity may alternatively be entered using the STATE directive.

11.9.2 ACTIVE: Number of active orbitals

The number of orbitals in the CASSCF active space must be specified using the ACTIVE directive.

E.g.,

```
active 10
```

The input molecular orbitals (see the vectors directive in MCSCF Vectors and SCF Vectors) must be arranged in order

1. doubly occupied orbitals,
2. active orbitals, and
3. unoccupied orbitals.

11.9.3 ACTELEC: Number of active electrons

The number of electrons in the CASSCF active space must be specified using the ACTELEC directive. An error is reported if the number of active electrons and the multiplicity are inconsistent.

The number of closed shells is determined by subtracting the number of active electrons from the total number of electrons (which in turn is derived from the sum of the nuclear charges minus the total system charge).

11.9.4 MULTIPLICITY

The spin multiplicity must be specified and is enforced by projection of the determinant wavefunction.

E.g., to obtain a triplet state

```
multiplicity 3
```

11.9.5 SYMMETRY: Spatial symmetry of the wavefunction

This species the irreducible representation of the wavefunction as an integer in the range 1–8 using the same numbering of representations as output by the SCF program. Note that only Abelian point groups are supported.

E.g., to specify a B_1 state when using the C_{2v} group

```
symmetry 3
```

11.9.6 STATE: Symmetry and multiplicity

The electronic state (spatial symmetry and multiplicity) may alternatively be specified using the conventional notation for an electronic state, such as $^3B_2$ for a triplet state of B_2 symmetry. This would be accomplished with the input

```
state 3b2
```

which is equivalent to

```
symmetry 4
multiplicity 3
```

11.9.7 VECTORS: Input/output of MO vectors

Calculations are best started from RHF/ROHF molecular orbitals (seeSCF), and by default vectors are taken from the previous MCSCF or SCF calculation. To specify another input file use the VECTORS directive. Vectors are by default output to the input file, and may be redirected using the output keyword. The swap keyword of the VECTORS directive may be used to reorder orbitals to obtain the correct active space.

The LOCK keyword allows the user to specify that the ordering of orbitals will be locked to that of the initial vectors, insofar as possible. The default is to order by ascending orbital energies within each orbital space. One application where locking might be desirable is a calculation where it is necessary to preserve the ordering of a previous geometry, despite flipping of the orbital energies. For such a case, the LOCK directive can be used to prevent the SCF calculation from changing the ordering, even if the orbital energies change.

Output orbitals of a converged MCSCF calculation are canonicalized as follows:

- Doubly occupied and unoccupied orbitals diagonalize the corresponding blocks of an effective Fock operator. Note that in the case of degenerate orbital energies this does not fully determine the orbtials.
- Active-space orbitals are chosen as natural orbitals by diagonalization of the active space 1-particle density matrix. Note that in the case of degenerate occupations that this does not fully determine the orbitals.

11.9.8 HESSIAN: Select preconditioner

The MCSCF will use a one-electron approximation to the orbital-orbital Hessian until some degree of convergence is obtained, whereupon it will attempt to use the exact orbital-orbital Hessian which makes the micro iterations more expensive but potentially reduces the total number of macro iterations. Either choice may be forced throughout the calculation by specifying the appropriate keyword on the HESSIAN directive.

E.g., to specify the one-electron approximation throughout

```
hessian onel
```

11.9.9 LEVEL: Level shift for convergence

The Hessian used in the MCSCF optimization is by default level shifted by 0.1 until the orbital gradient norm falls below 0.01, at which point the level shift is reduced to zero. The initial value of 0.1 may be changed using the LEVEL directive. Increasing the level shift may make convergence more stable in some instances.

E.g., to set the initial level shift to 0.5

```
level 0.5
```

11.9.10 PRINT and NOPRINT

Specific output items can be selectively enabled or disabled using theprint control mechanism with the available print options listed in the table below.

Option	Class	Synopsis
ci energy	default	CI energy eigenvalue
fock energy	default	Energy derived from Fock matrices
gradient norm	default	Gradient norm
movecs	default	Converged occupied MO vectors
trace energy	high	Trace Energy
converge info	high	Convergence data and monitoring
precondition	high	Orbital preconditioner iterations
microci	high	CI iterations in line search
canonical	high	Canonicalization information
new movecs	debug	MO vectors at each macro-iteration
ci guess	debug	Initial guess CI vector
density matrix	debug	One- and Two-particle density matrices

11.10 GW

11.10.1 Overview

Electron attachment and detachment energies can be accurately described by many-body perturbation theory (MBPT) methods. In particular, the *GW* approximation (GWA) to the self-energy is a MBPT method that has seen recent interest in its application to molecules due to a promising cost/accuracy ratio.

The *GW* module implemented in NWChem takes aDFT mean-field approximation to the Green's function,G_0, in order to solve the quasiparticle equation at the one-shot G_0W_0 or at various *levels* of the eigenvalue self-consistent*GW* approach (ev*GW*). Since the mean-field orbitals are kept fixed in all these approaches, the results depend on the actual starting point G_0 (hence, they depend on the exchange-correlation functional chosen for the underlying DFT calculation). For example, it has been known that a large fraction of exact exchange is needed for the accurate prediction of core-level binding energies at the one-shot G_0W_0 level.

For further theoretical insights and details about the actual implementation in NWChem, please refer to the paper by Mejia-Rodriguez et al[1].

GW input is provided using the compound directive

```
GW
 ...
END
```

The actual *GW* calculation will be performed when the input module encounters the TASK directive.

Note that DFT must be specified as the underlying *QM* theory before GW. The charge, geometry, and DFT options are all specified as normal.

In addition to an atomic orbital basis set, the *GW* module **requires** an *auxiliary basis set* to be provided in order to fit the four-center electron repulsion integrals. The auxliary basis set can have either the cd basis or ri basis names (see also DFT). Three combinations can be obtained:

- If a ri basis is given without a cd basis, the ground-state DFT will be performed without density fitting, and the *GW* task will use the ri basis to fit the integrals.
- If a cd basis is given without a ri basis, **both** DFT and *GW* tasks will be performed using the cd basis to fit the integrals.
- If both cd basis and ri basis are present, the cd basis will be used for the DFT task, while the ri basis will be used for the *GW* task.

11.10.2 GW Input directive

There are sub-directives which allow for customized *GW* calculations. The most general *GW* input block directive will look like:

```
GW
  RPA
  CORE
  EVGW [<integer eviter default 4>]
  EVGW0 [<integer eviter default 4>]
  FIRST <integer first_orbital default 1>
  METHOD [ [analytic] || [cdgw <integer grid_points default 200>] ]
  ETA <real infinitesimal default 0.001>
  SOLVER [ [newton <integer maxiter default 10> ] || [graph] ]
  STATES [ [ alpha || beta ] [occ <integer number default 1>] [vir <integer default 0>] ]
  CONVERGENCE <real threshold default 0.005> [<string units default ev>]
END
```

The following sections describe these keywords.

11.10.2.1 RPA

The keyword RPA triggers the computation of the RPA correlation energy. This adds a little overhead to the *CD-GW* approach.

11.10.2.2 CORE and FIRST

The CORE keyword forces to start counting the STATES from the FIRST molecular orbital **upwards**.

The FIRST keyword has no meaning without CORE specified.

11.10.2.3 EVGW and EVGW0

The EVGW keyword trigger the partial self-consistnet ev*GW* approach, where both the Green's function *G* and the screened Coulomb *W* are updated by using the quasiparticle energies from the previous step in their construction.

Similarly, the EVGW0 triggers the evGW_0 approach, where only the Green's function *G* is updated with the quasiparticle energies of the previous iterations. W_0 is kept fixed.

Both partial self-consistent cycles run for eviter number of cycles.

The use of EVGW or EVGW0 will trigger the use of a scissor-shift operator for all states not updated in the ev*GW* cycle.

11.10.2.4 METHOD

Two different techniques to obtain the diagonal self-energy matrix elements are implemented in NWChem.

The analytic method builds and diagonalizes the full Casida RPA matrix in order to obtain the screened Coulomb matrix elements. The Casida RPA matrix grows very rapidly in size ($N_{occ} \times N_{vir}$) and ultimately yields a N^6 scaling due to the diagonalization step. It is therefore recommended to link the ELPA and turn on its use by setting

```
SET dft:scaleig e
```

The cdgw method uses the Contour-Deformation technique in order to avoid the N^6 diagonalization step. The diagonal self-energy matrix elements Σ_{nn} are obtained via a numerical integration on the imaginary axis and the integrals over closed contours on the first and third quadrants of the complex plane. The grid_points value controls the density of the modified Gauss-Legendre grid used in the numerical integration over the imaginary axis.

Both analytic and cdgw methods are suitable for *core* **and** *valence* calculations.

11.10.2.5 ETA

The magnitude of the imaginary infinitesimal can be controlled using the keyword ETA. The default value of 0.001 should work rather well for *valence* calculations, but CORE calculations might need a larger value, sometimes between 0.005 or even 0.01.

11.10.2.6 SOLVER

Two methods to solver the quasiparticle equations are implemented in NWChem.

The newton method uses a modified Newton approach to find the fixed-point of the quasiparticle equations. The Newton method tries to bracket the solution and switches to a golden section method whenever the Newton step goes beyond the bracketing values.

The graph method uses a frequency grid in order to bracket the solution between two consecutive grid points. The number of grid points is controlled heuristically depending on the METHOD and on the presence, or not, of nearby states in a cluster of energy (see below).

Regardless of the solver, the energies of the states are always classified in clusters with a maximum extension of 1.5 eV. For a given cluster of energies, the newton method will start with the state closer to the Fermi level and use its solution as guess for the rest of the states in the cluster. The graph method will look for the solution of all the states in a given cluster at once with a frequency grid with range large enough to encompass all the cluster ± 0.2 eV.

11.10.2.7 STATES

The keyword STATES controls for which particular state the *GW* quasiparticle equations are to be solved. The keyword might appear twice, one for the *alpha* spin channel and one for the *beta* channel. The *beta* channel keyword is meaningless for restricted closed-shell DFT calculations (MULT 1 without ODFT in the DFT input block).

The number of occupied states will be counted starting from the state closest to the Fermi level (HOMO) unless the keyword CORE is present. The virtual states will **always** be counted from the state closest to the Fermi level upwards.

A **-1** following either occ or vir stands for all states in the respective space.

11.10.2.8 CONVERGENCE

The convergence threshold of the quasiparticle equations can be controlled with the keyword CONVERGENCE and might be given either in eV or Hartree au.

11.10.3 Sample Input File

- A *GW* calculation requesting the core-level binding energies of all 1*s* states (6 Fluorines and 6 Carbons) using the CD-GW method.

```
title "CDGW C6F6 core"
start
echo

memory 2000 mb

geometry
 C     -0.21589696     1.38358991     0.00000000
 C     -1.30618181     0.50480033     0.00000000
 C     -1.09023026    -0.87871037     0.00000000
 C      0.21590562    -1.38360671     0.00000000
 C      1.30610372    -0.50476737     0.00000000
 C      1.09020243     0.87883094     0.00000000
 F     -0.42025331     2.69273557     0.00000000
 F     -2.54211642     0.98238922     0.00000000
 F     -2.12174279    -1.71033945     0.00000000
 F      0.42026196    -2.69275237     0.00000000
 F      2.54203111    -0.98237286     0.00000000
 F      2.12188428     1.71024875     0.00000000
end

basis "ao basis" spherical
 * library cc-pvdz
end

basis "cd basis" spherical
 * library cc-pvdz-ri
end

dft
 xc xpbe96 0.55 hfexch 0.45 cpbe96 1.0
 direct
end

gw
 core
 eta 0.01
 method cdgw
 solver newton 15
 states alpha occ 12
end

task dft gw
```

- A *valence GW* calculation to obtain the vertical ionization potential and the vertical electron affinity of the water molecule using the analytic method.

```
start

geometry
O     -0.000545      1.517541      0.000000
H      0.094538      0.553640      0.000000
H      0.901237      1.847958      0.000000
end

basis "ao basis" spherical
 h library def2-svp
 o library def2-svp
end

basis "cd basis" spherical
 h library def2-universal-jkfit
 o library def2-universal-jkfit
end

dft
 mult 1
 xc pbe96
 grid fine
 direct
end

gw
 states alpha occ 1 vir 1
end

task dft gw
```

- An evGW_0 calculation with 10 iterations using the analytic method. All occupied energies, but only 10 virtual ones, are updated using *GW*. The rest of the virtual states are shifted using the so-called scissor operator.

```
start

geometry
O     -0.000545      1.517541      0.000000
H      0.094538      0.553640      0.000000
H      0.901237      1.847958      0.000000
end

basis "ao basis" spherical
 h library def2-svp
 o library def2-svp
end

basis "cd basis" spherical
 h library def2-universal-jkfit
 o library def2-universal-jkfit
end

dft
 mult 1
 xc pbe96
 grid fine
 direct
end

gw
 evgw0 10
 states alpha occ -1 vir 10
end

task dft gw
```

11.10.4 References

1. Mejia-Rodriguez, D.; Kunitsa, A.; Aprà, E.; Govind, N. Scalable Molecular GW Calculations: Valence and Core Spectra. *Journal of Chemical Theory and Computation* **2021**, *17* (12), 7504–7517. https://doi.org/10.1021/acs.jctc.1c00738

11.11 XTB

11.11.1 Overview

XTB method [1]. Full documentation available at the XTB website.
The NWChem implementation makes use of the TBlite library.

```
XTB
  [ACC <real acc default 1.0>]
  [UHF <integer uhf default 0>]
  [METHOD gfn1 || gfn2 default gfn2]
  [GUESS]
  [VERBOSITY]
  [PRINT]
  [NOPRINT]
  [NSPIN <integer nspin default 1>]
  [BROYDEN <real broyden default 0.4>]
END
```

11.11.2 ACC: Accuracy

For example, if the user wants to increase the accuracy:

```
acc 1d-4
```

11.11.3 UHF: number of unpaired electrons

```
uhf 3
```

11.11.4 NSPIN:

A value of `nspin` equal to two triggers an open-shell calculation (while the default value of 1 triggers a closed-shell calculation).

```
nspin 2
```

11.11.5 References

1. Bannwarth, C.; Caldeweyher, E.; Ehlert, S.; Hansen, A.; Pracht, P.; Seibert, J.; Spicher, S.; Grimme, S. Extended Tight-Binding Quantum Chemistry Methods. *WIREs Computational Molecular Science* **2020**, *11* (2). https://doi.org/10.1002/wcms.1493

12 Quantum Molecular Dynamics

12.1 Pseudopotential plane-wave density functional theory (NWPW)

- Pseudopotential plane-wave density functional theory (NWPW)
  - Overview
  - PSPW Tasks: Gamma Point Calculations
    - PAW Potentials
      - PAW Implementation Notes
    - Exchange-Correlation Potentials
      - DFT + U Corrections

        - Langreth style vdw and vdw van der Wall functionals
        - Grimme Dispersion Corrections
        - Using Exchange-Correlation Potentials Available in the DFT Module
        - Exact Exchange
        - Self-Interaction Corrections
    - Wannier
    - Mulliken Analysis
    - Density of States
    - Projected Density of States
    - Point Charge Analysis
    - PSPW_DPLOT: Generate Gaussian Cube Files
- Band Tasks: Multiple k-point Calculations
    - Brillouin Zone
        - Band Structure Paths
        - Special Points of Different Space Groups (Conventional Cells)
    - Screened Exchange
    - Density of States and Projected Density of States
    - Two-Component Wavefunctions (Spin-Orbit ZORA)
    - BAND_DPLOT: Generate Gaussian Cube Files
- Car-Parrinello
    - Adding Geometry Constraints to a Car-Parrinello Simulation
    - Car-Parrinello Output Datafiles
        - XYZ motion file
        - ION_MOTION motion file
        - EMOTION motion file
        - HMOTION motion file
        - EIGMOTION motion file
        - OMOTION motion file
- Born-Oppenheimer Molecular Dynamics
- i-PI Socket Communication
- Metropolis Monte-Carlo
- Free Energy Simulations
    - MetaDynamics
        - Input
    - TAMD - Temperature Accelerated Molecular Dynamics
        - Input
    - Collective Variables
        - Bond Distance Collective Variable
        - Angle Collective Variable
        - Coordination Collective Variable

        - N-Plane Collective Variable
        - User defined Collective Variable
- Extended X-Ray Absorption Fine Structure (EXAFS) - Integration with FEFF6L
- Frozen Phonon Calculations
- Steepest Descent
- Simulation Cell
- Unit Cell Optimization
- SMEAR - Fractional Occupation of the Molecular Orbitals
- Spin Penalty Functions
- AIMD/MM (QM/MM)
- PSP_GENERATOR
    - ATOMIC_FILLING Block
    - CUTOFF
    - SEMICORE_RADIUS
- PAW Tasks: Legacy Implementation
- Pseudopotential and PAW basis Libraries
- NWPW RTDB Entries and Miscellaneous DataFiles
    - Ion Positions
    - Ion Velocities
    - Wavefunction Datafile
    - Velocity Wavefunction Datafile
    - Formatted Pseudopotential Datafile
    - One-Dimensional Pseudopotential Datafile
- Car-Parrinello Scheme for Ab Initio Molecular Dynamics
    - Verlet Algorithm for Integration
    - Constant Temperature Simulations: Nose-Hoover Thermostats
- NWPW Tutorial 1: S2 dimer examples with PSPW
    - Total energy of S2 dimer with LDA approximation
    - Structural optimization of S2 dimer with LDA approximation
    - Frequency calculation of S2 dimer with LDA approximation
    - Ab initio molecular dynamics simulation (Car-Parrinello) of S2 dimer using the LDA approximation
    - Ab initio molecular dynamics simulation (Born-Oppenheimer) of S<sub2 dimer using the LDA approximation
- NWPW Tutorial 2: Using PSPW Car-Parrinello Simulated Annealing Simulations to Optimize Structures
    - Simulated Annealing Using Constant Energy Simulation
    - Simulated Annealing Using Constant Temperature Simulation
- NWPW Tutorial 3: using isodesmic reaction energies to estimate gas-phase thermodynamics
- NWPW Tutorial 4: AIMD/MM simulation of CCl\(_4\) + 64 H\(_2\)O
- NWPW Tutorial 5: Optimizing the Unit Cell and Geometry of Diamond
    - Optimizing the Unit Cell and Geometry for an 8 Atom Supercell of Diamond with PSPW
    - Optimizing the Unit Cell for an 8 Atom Supercell of Diamond with BAND

    - Using BAND to Optimize the Unit Cell for a 2 Atom Primitive Cell of Diamond
    - Using BAND to Calculate the Band Structures of Diamond
    - Using BAND to Calculate the Density of States of Diamond
    - Calculate the Phonon Spectrum of Diamond
- NWPW Tutorial 6: optimizing the unit cell of nickel with fractional occupation
- NWPW Tutorial 7: Optimizing the unit cells with symmetry: Diamond with Fd-3m symmetry and Brucite with P-3m1 symmetry
- NWPW Tutorial 8: NVT Metropolis Monte-Carlo Simulations
- NWPW Tutorial 9: NPT Metropolis Monte-Carlo Simulations
- NWPW Tutorial 9: Free Energy Simulations
- PAW Tutorial
    - Optimizing a water molecule
    - Optimizing a unit cell and geometry for Silicon-Carbide
    - Running a Car-Parrinello Simulation
- NWPW Capabilities and Limitations
- Questions and Difficulties

12.1.1 Overview

The NWChem plane-wave (NWPW) module uses pseudopotentials and plane-wave basis sets to perform Density Functional Theory calculations (simple introduction pw-lecture.pdf). This module complements the capabilities of the more traditional Gaussian function based approaches by having an accuracy at least as good for many applications, yet is still fast enough to treat systems containing hundreds of atoms. Another significant advantage is its ability to simulate dynamics on a ground state potential surface directly at run-time using the Car-Parrinello algorithm. This method's efficiency and accuracy make it a desirable first principles method of simulation in the study of complex molecular, liquid, and solid state systems. Applications for this first principles method include the calculation of free energies, search for global minima, explicit simulation of solvated molecules, and simulations of complex vibrational modes that cannot be described within the harmonic approximation.

The NWPW module is a collection of three modules.

- PSPW - (PSeudopotential Plane-Wave) A gamma point code for calculating molecules, liquids, crystals, and surfaces.
- Band - A band structure code for calculating crystals and surfaces with small band gaps (e.g. semi-conductors and metals).
- PAW - a (gamma point) projector augmented plane-wave code for calculating molecules, crystals, and surfaces (*This module will be deprecated in the future releases since PAW potentials have been added to PSPW*)

The PSPW, Band, and PAW modules can be used to compute the energy and optimize the geometry. Both the PSPW and Band modules can also be used to find saddle points, and compute numerical second derivatives. In addition the PSPW module can also be used to perform Car-Parrinello molecular dynamics. Section PSPW Tasks describes the tasks contained within the PSPW module, section Band Tasks describes the tasks contained within the Band module, section PAW Tasks describes the tasks contained within the PAW module, and sectionPseudopotential and PAW basis Libraries describes the pseudopotential library included with NWChem. The datafiles used by the PSPW module are described in section NWPW RTDB Entries and DataFiles. Car-Parrinello output data files are described in sectionCar-Parrinello Output Datafiles, and the minimization and Car-Parrinello algorithms are described in sectionCar-Parrinello Scheme for Ab Initio Molecular Dynamics. Examples of how to setup and run a PSPW geometry optimization, a Car-Parrinello simulation, a band structure minimization, and a PAW geometry optimization are presented at the end. Finally in section NWPW Capabilities and Limitations the capabilities and limitations of the NWPW module are discussed.

As of NWChem 6.6 to use PAW potentials the user is recommended to use the implementation contained in the PSPW module (see Sections). PAW potentials are also being integrated into the BAND module. Unfortunately, the porting to BAND was not completed for the NWChem 6.6 release.

If you are a first time user of this module it is recommended that you skip the next five sections and proceed directly to the tutorials.

12.1.2 PSPW Tasks: Gamma Point Calculations

All input to the PSPW Tasks is contained within the compound PSPW block,

```
PSPW
  ...
END
```

To perform an actual calculation a TASK PSPW directive is used (Section Task).

```
TASK PSPW
```

In addition to the directives listed inTask, i.e.

```
TASK PSPW energy
TASK PSPW gradient
TASK PSPW optimize
TASK PSPW saddle
TASK PSPW freqencies
TASK PSPW vib
```

there are additional directives that are specific to the PSPW module, which are:

```
TASK PSPW [Car-Parrinello            ||
          Born-Oppenheimer           ||
          Metropolis                 ||
          pspw_et                    ||
          noit_energy                ||
          stress                     ||
          pspw_dplot                 ||
          wannier                    ||
          expand_cell                ||
          exafs                      ||
          ionize                     ||
          lcao                       ||
          rdf                        ||
          aimd_properties            ||
          translate                  ||
          psp_generator              ||
          steepest_descent           ||
          psp_formatter              ||
          wavefunction_initializer   ||
          v_wavefunction_initializer ||
          wavefunction_expander      ]
```

Once a user has specified a geometry, the PSPW module can be invoked with no input directives (defaults invoked throughout). However, the user will probably always specify the simulation cell used in the computation, since the default simulation cell is not well suited for most systems. There are sub-directives which allow for customized application; those currently provided as options for the PSPW module are:

```
NWPW
  SIMULATION_CELL           ... (see section [Simulation Cell](#simulation-cell)) END
  CELL_NAME <string cell_name default 'cell_default'>
  VECTORS [[input (<string input_wavefunctions default file_prefix.movecs>) ||
          [output(<string output_wavefunctions default file_prefix.movecs>)]]
  XC (Vosko      || LDA          || PBE96        || revPBE       || PBEsol          ||
     LDA-SIC    || LDA-SIC/2    || LDA-0.4SIC    || LDA-SIC/4    || LDA-0.2SIC    ||
     PBE96-SIC  || PBE96-SIC/2  || PBE96-0.4SIC  || PBE96-SIC/4  || PBE96-0.2SIC  ||
     revPBE-SIC || revPBE-SIC/2 || revPBE-0.4SIC || revPBE-SIC/4 || revPBE-0.2SIC ||
     PBE96-Grimme2  || PBE96-Grimme3  || PBE96-Grimme4  || BLYP-Grimme2   || BLYP-Grimme3    || BLYP-Grimme4 ||
     revPBE-Grimme2 || revPBE-Grimme3 || revPBE-Grimme4 || PBEsol-Grimme2 || PBEsol-Grimme3 || PBEsol-Grimme4 ||
     PBE0-Grimme2    || PBE0-Grimme3    || PBE0-Grimme4   || B3LYP-Grimme2  || B3LYP-Grimme3  || B3LYP-Grimme4 ||
     revPBE0-Grimme2 || revPBE0-Grimme3 || revPBE0-Grimme4 ||
     PBE0       || revPBE0      || HSE  || HF || default Vosko)
  XC new ...(see section [Using Exchange-Correlation Potentials Available in the DFT Module](#Using_Exchange-Correlation_Potentials_Available_in_the_DFT_Module))
  DFT||ODFT||RESTRICTED||UNRESTRICTED
  MULT <integer mult default 1>
  CG
  LMBFGS
  SCF [Anderson|| simple || Broyden]
     [CG || RMM-DIIS]
     [density || potential]
     [ALPHA real alpha default 0.25]
     [Kerker real ekerk nodefault]
     [ITERATIONS integer inner_iterations default 5]
     [OUTER_ITERATIONS integer outer_iterations default 0]
  LOOP <integer inner_iteration outer_iteration default 10 100>
  TOLERANCES <real tole tolc default 1.0e-7 1.0e-7>
  FAKE_MASS <real fake_mass default 400000.0>
  TIME_STEP <real time_step default 5.8>
  EWALD_NCUT <integer ncut default 1>
  EWALD_RCUT <real rcut default (see input description)>
  CUTOFF  <real cutoff>
  ENERGY_CUTOFF <real ecut default (see input description)>
  WAVEFUNCTION_CUTOFF <real wcut default (see input description)>
  ALLOW_TRANSLATION
  TRANSLATION (ON || OFF)
  ROTATION (ON || OFF)
  MULLIKEN [OFF]
  EFIELD

  BO_STEPS <integer bo_inner_iteration bo_outer_iteration default 10 100>
  MC_STEPS <integer mc_inner_iteration mc_outer_iteration default 10 100>
  BO_TIME_STEP  <real bo_time_step default 5.0>
  BO_ALGORITHM [verlet|| velocity-verlet || leap-frog]
  BO_FAKE_MASS <real bo_fake_mass default 500.0>

  SOCKET (UNIX || IPI_CLIENT) <string socketname default (see input description)>

  MAPPING <integer mapping default 1>
  NP_DIMENSIONS <integer npi npj default -1 -1>
  CAR-PARRINELLO            ... (see section [Car-Parrinello](#car-parrinello-scheme-for-ab-initio-molecular-dynamics)) END
  STEEPEST_DESCENT          ... (see section [Steepest Descent](#STEEPEST_DESCENT)) END
  DPLOT                   ... (see section [DPLOT](#DPLOT)) END
  WANNIER                   ... (see section [Wannier](#Wannier)) END
  PSP_GENERATOR             ... (see section [PSP Generator](#PSP_GENERATOR))) END

  WAVEFUNCTION_INITIALIZER   ... (see section [Wavefunction Initializer](NWPW_RETIRED.md#WAVEFUNCTION_INITIALIZER) - retired) END
  V_WAVEFUNCTION_INITIATIZER ... (see section [Wavefunction Velocity Initializer (NWPW_RETIRED#V_WAVEFUNCTION_INITIALIZER) - retired) END
  WAVEFUNCTION_EXPANDER      ... (see section [Wavefunction Expander](NWPW_RETIRED.md#WAVEFUNCTION_EXPANDER) - retired) END
  INPUT_WAVEFUNCTION_FILENAME <string input_wavefunctions default file_prefix.movecs>
 OUTPUT_WAVEFUNCTION_FILENAME <string output_wavefunctions default file_prefix.movecs>
END
```

The following list describes the keywords contained in the PSPW input block.

- cell_name - name of the simulation_cell named cell_name . See section Simulation Cell.
- input_wavefunctions - name of the file containing one-electron orbitals
- output_wavefunctions - name of the file that will contain the one-electron orbitals at the end of the run.
- fake_mass - value for the electronic fake mass\(\mu\) This parameter is not presently used in a conjugate gradient simulation.
- time_step - value for the time step\(\Delta t\). This parameter is not presently used in a conjugate gradient simulation.
- inner_iteration - number of iterations between the printing out of energies and tolerances
- outer_iteration - number of outer iterations
- tole - value for the energy tolerance.

- tolc - value for the one-electron orbital tolerance.
- cutoff - value for the cutoff energy used to define the wavefunction. In addition using the CUTOFF keyword automatically sets the cutoff energy for the density to be twice the wavefunction cutoff.
- ecut - value for the cutoff energy used to define the density. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- wcut - value for the cutoff energy used to define the one-electron orbitals. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- ncut - value for the number of unit cells to sum over (in each direction) for the real space part of the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.
- rcut - value for the cutoff radius used in the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.

Default set to be $\frac{MIN(\left| \vec{a_i} \right|)}{\pi}, i=1,2,3$.

- (Vosko || PBE96 || revPBE || …) - Choose between Vosko et al's LDA parameterization or the orginal and revised Perdew, Burke, and Ernzerhof GGA functional. In addition, several hybrid options.
- MULT - optional keyword which if specified allows the user to define the spin multiplicity of the system
- MULLIKEN - optional keyword which if specified causes a Mulliken analysis to be performed at the end of the simulation.
- EFIELD - optional keyword which if specified causes an atomic electric field analysis to be performed at the end of the simulation.
- ALLOW_TRANSLATION - By default the the center of mass forces are projected out of the computed forces. This optional keyword if specified allows the center of mass forces to not be zero.
- TRANSLATION - By default the the center of mass forces are projected out of the computed forces. TRANSLATION ON allows the center of mass forces to not be zero.
- ROTATION - By default the overall rotation is not projected out of the computed forces. ROTATION OFF projects out the overal rotation of the molecule.
- CG - optional keyword which sets the minimizer to 1
- LMBFGS - optional keyword which sets the minimizer to 2
- SCF - optional keyword which sets the minimizer to be a band by band minimizer. Several options are available for setting the density or potential mixing, and the type of Kohn-Sham minimizer.
- mapping - for a value of 1 slab FFT is used, for a value of 2 a 2d-hilbert FFT is used.

A variety of prototype minimizers can be used to minimize the energy. To use these other optimizers the following SET directive needs to be specified:

```
set nwpw:mimimizer 1 # Default - Grassman conjugate gradient minimizer is used to minimize the energy.
set nwpw:mimimizer 2 # Grassman LMBFGS minimimzer is used to minimize the energy.
set nwpw:minimizer 4 # Stiefel conjugate gradient minimizer is used to minimize the energy.
set nwpw:minimizer 5 # Band-by-band (potential) minimizer is used to minimize the energy.
set nwpw:minimizer 6 # Projected Grassman LMBFGS minimizer is used to minimize the energy.
set nwpw:minimizer 7 # Stiefel LMBFGS minimizer is used to minimize the energy.
set nwpw:minimizer 8 # Band-by-band (density) minimizer is used to minimize the energy.
```

Limited testing suggests that the Grassman LMBFGS minimizer is about twice as fast as the conjugate gradient minimizer. However, there are several known cases where this optimizer fails, so it is currently not a default option, and should be used with caution.

In addition the following SET directives can be specified:

```
set nwpw:lcao_skip .false. # Initial wavefunctions generated using an LCAO guess.
set nwpw:lcao_skip .true. # Default - Initial wavefunctions generated using a random plane-wave guess.
set nwpw:lcao_print .false. # Default - Output not produced during the generation of the LCAO guess.
set nwpw:lcao_print .true. # Output produced during the generation of the LCAO guess.
set nwpw:lcao_iterations 2 #specifies the number of LCAO iterations.
```

12.1.2.1 PAW Potentials

The PSPW code can now handle PAW potentials. To use them the pseudopotentials input block is used to redirect the code to use the paw potentials located in the default paw potential library ($NWCHEM_TOP/src/nwpw/libraryp/paw_default). For example, to redirect the code to use PAW potentials for carbon and hydrogen, the following input would be used.

```
nwpw
  pseudopotentials
    C library paw_default
    H library paw_default
  end
end
```

Most of the capabilities of PSPW will work with PAW potentials including geometry optimization, Car-Parrinello ab initio molecular dynamics, Born-Oppenheimer ab initio molecular dynamics, Metropolis Monte-Carlo, and AIMD/MM. Unfortunately, some of the functionality is missing at this point in time such as Mulliken analysis, and analytic stresses. However these small number of missing capabilities should become available over the next couple of months in the development tree of NWChem.

Even though analytic stresses are not currently available with PAW potentials unit cell optimization can still be carried out using numerical stresses. The following SET directives can be used to tell the code to calculate stresses numerically.

```
set includestress  .true.              #this option tells driver to optimize the unit cell
set includelattice .true.              #this option tells driver to optimize cell using a,b,c,alpha,beta,gamma
set nwpw:frozen_lattice:thresh 999.0   #large number guarentees the lattice gridding does not adjust during optimization
set nwpw:cif_filename  pspw_corundum
set nwpw:stress_numerical .true.
set nwpw:lstress_numerical .true.
```

12.1.2.1.1 PAW Implementation Notes

The main idea in the PAW method(Blochl 1994) is to project out the high-frequency components of the wavefunction in the atomic sphere region. Effectively this splits the original wavefunction into two parts:

$$\psi_n(\mathbf{r}) = \tilde{\psi}_n(\mathbf{r}) + \sum_I \psi_n^I(\mathbf{r})$$

The first part $\tilde{\psi}_n(\mathbf{r})$ is smooth and can be represented using a plane wave basis set of practical size. The second term is localized with the atomic spheres and is represented on radial grids centered on the atoms as

$$\psi_n^I(\mathbf{r}) = \sum_{\alpha} (\varphi_{\alpha}^I(\mathbf{r})-\tilde{\varphi}_{\alpha}^I(\mathbf{r}))c_{n\alpha}^I$$

where the coefficients $c_{n\alpha}^I$ are given by

$$ c_{n\alpha}^I = <\tilde{p}_{\alpha}^I | \tilde{\psi}_n>$$

This decomposition can be expressed using an invertible linear transformation,T, is defined which relates the stiff one-electron wavefunctions ψ_n to a set of smooth one-electron wavefunctions$\tilde{\psi}_n$

$$\tilde{\psi}_n = T \psi_n$$
$$\psi_n = T^{-1} \tilde{\psi}_n$$

which can be represented by fairly small plane-wave basis. The transformation$T$ is defined using a local PAW basis, which consists of atomic orbitals, $\varphi{\alpha}^I(\mathbf{r})$, smooth atomic orbitals, $\tilde{\varphi}_\alpha^I(\mathbf{r})$ which coincide with the atomic orbitals outside a defined atomic sphere, and projector functions, $p_{\alpha}^I(\mathbf{r})$. Where I is the atomic index and α is the orbital index. The projector functions are constructed such that they are localized within the defined atomic sphere and in addition are orthonormal to the atomic orbitals. Blöchl defined the invertible linear

transformations by

$$T = 1 + \sum_I \sum_\alpha (|\tilde{\varphi}_\alpha^I> - |\varphi_\alpha^I >) < p_\alpha^I|$$
$$\tilde{T} = 1 + \sum_I \sum_{\alpha} (|\varphi_{\alpha}^I> - |\tilde{\varphi}_{\alpha}^I >)<\tilde{\varphi}_{\alpha}^I|$$
$$|\tilde{p}_{\alpha}^I> = \sum_{\beta} [<\tilde{p}^I | \varphi^I >]_{\alpha\beta}^{-1} | p_{\beta}^I>$$

The main effect of the PAW transformation is that the fast variations of the valence wave function in the atomic sphere region are projected out using local basis set, thereby producing a smoothly varying wavefunction that may be expanded in a plane wave basis set of a manageable size.

The expression for the total energy in PAW method can be separated into the following 15 terms.

$$E_{PAW} = \tilde{E}_{kinetic-pw} + \tilde{E}_{vlocal-pw} + \tilde{E}_{Coulomb-pw} + \tilde{E}_{xc-pw} + E_{ion-ion}$$
$$ + E_{cmp-cmp} + E_{cmp-pw} + E_{valence-core} + E_{kinetic-core} + E_{ion-core}$$

The first five terms are essentially the same as for a standard pseudopotential plane-wave program, minus the non-local pseudopotential, where

$$\tilde{E}_{kinetic-pw} = \sum_i \sum_{\mathbf{G}} \frac{|\mathbf{G}|^2}{2} \tilde{\psi}_i^{*}(\mathbf{G}) \tilde{\psi}_i(\mathbf{G})$$
$$\tilde{E}_{vlocal-pw} = \sum_{\mathbf{G}} \tilde{\rho}(\mathbf{G}) V_{local}(\mathbf{G})$$
$$\tilde{E}_{Coulomb-pw} = \frac{\Omega}{2} \sum_{\mathbf{G}} \frac{4\pi}{|\mathbf{G}|} \tilde{\rho}^{*}(\mathbf{G}) \tilde{\rho}(\mathbf{G})$$
$$\tilde{E}_{xc-pw} = \frac{\Omega}{N_1 N_2 N_3} \sum_{\mathbf{r}} \tilde{\rho}(\mathbf{r}) \epsilon_{xc}(\tilde{\rho}(\mathbf{r}))$$
$$E_{ion-ion} = \frac{1}{2\Omega} \sum_{\mathbf{G}} \frac{4\pi}{|\mathbf{G}|^2} \exp(\frac{|\mathbf{G}|^2}{4\epsilon}) \sum_{I,J} Z_I \exp (-i \mathbf{G} \cdot \mathbf{R}_I) Z_J \exp (-i \mathbf{G} \cdot \mathbf{R}_J) +$$
$$\frac{1}{2}\sum_{\mathbf{a}} \sum_{I,J \in |\mathbf{R}_I-\mathbf{R}_J+\mathbf{a}|} Z_I Z_J \frac{erf(\epsilon |\mathbf{R}_I-\mathbf{R}_J+\mathbf{a}|)}{|\mathbf{R}_I-\mathbf{R}_J+\mathbf{a}|} - \frac{\epsilon}{\pi}\sum_I Z_I^2 +$$
$$ - \frac{\pi}{2\epsilon^2\Omega} \left(\sum_{I} Z_{I} \right)^2 $$

The local potential in the $\tilde{E}_{vlocal-pw}$ term is the Fourier transform of

$$V_{local}(\mathbf{r}) = -\sum_I Z_I \frac{\frac{|\mathbf{r}-\mathbf{R}_I|}{\sigma_I}}{|\mathbf{r}-\mathbf{R}_I|} + v_{ps}^I(|\mathbf{|\mathbf{r}-\mathbf{R}_I|})$$

It turns out that for many atoms σ_I needs to be fairly small. This results in $V_{local} (\mathbf{r})$ being stiff. However, since in the integral above this function is multiplied by a smooth density $\tilde{\rho}(\mathbf{G})$ the expansion of $V_{local}(\mathbf{G})$ only needs to be the same as the smooth density. The auxiliary pseudoptential $v_{ps}^I (|\mathbf{r}-\mathbf{R}_I |)$ is defined to be localized within the atomic sphere and is introduced to remove ghost states due to local basis set incompleteness.

The next four terms are atomic based and they essentially take into account the difference between the true valence wavefunctions and the pseudowavefunctions.

$$ E_{kinetic-atom}= \sum_I \sum_i \sum_{\alpha\beta} <\tilde{\psi}_i|\tilde{p}_{\alpha}^I < (t_{atom}^I)_{\alpha\beta} <\tilde{p}_{\beta}^I|\tilde{\psi}_i $$
$$ E_{local-atom}=\sum_I \sum_i \sum_{\alpha\beta} <\tilde{\psi}_i|\tilde{p}_{\alpha}^I> (u_{atom}^I)_{\alpha\beta} <\tilde{p}_{\beta}^I|\tilde{\psi}_i>$$
$$ E_{xc-atom}=\sum_I \sum_{\theta\phi} w_{\theta\phi} \int_0^{r_{cut}^I} r^2 (\rho^I(r,\theta,\phi) \epsilon_{xc}(\rho^I(r,\theta,\phi)) -\tilde{\rho}^I(r,\theta,\phi) \epsilon_{xc}(\tilde{\rho}^I(r,\theta,\phi))) dr $$
$$ E_{hartree-atom}= \sum_I W_{atom}^I =\frac{1}{2} \sum_I \sum_i \sum_{\alpha\beta} <\tilde{\psi}_i|\tilde{p}_{\alpha}> <\tilde{p}_{\beta}^I|\tilde{\psi}_i> $$
$$ \sum_j \sum_{\mu\nu} <\tilde{\psi}_j|\tilde{p}_{\mu}^I> <\tilde{p}_{\nu}^I|\tilde{\psi}_j> \sum_{lm} \tau_{l_\alpha m_\alpha, l_\beta m_\beta}^{lm} \tau_{l_\mu m_\mu, l_\nu m_\nu}^{lm} (V_{Heff}^I)_{\alpha\beta\mu\nu}^{l} $$

The next three terms are the terms containing the compensation charge densities.

$$E_{cmp\text{-}vloc}= \sum_{\mathbf{G}} [\rho_{cmp} (\mathbf{G}) \tilde{V}_{local}(\mathbf{G})+\tilde{\rho}_{cmp}(\mathbf{G}) (V_{local}(\mathbf{G})-\tilde{V}_{local}(\mathbf{G}))] + \\ \int (\rho_{cmp}(\mathbf{r})-\tilde{\rho}_{cmp}(\mathbf{r}))(V_{local}(r)-\tilde{V}_{local}(\mathbf{r}))d\mathbf{r}$$

$$E_{cmp\text{-}cmp}= \Omega \sum_{\mathbf{G} \neq 0} \frac{4\pi}{|\mathbf{G}|^2} [\rho_{cmp}(\mathbf{G}) \tilde{\rho}_{cmp}(\mathbf{G})- \frac{1}{2} \tilde{\rho}_{cmp}(\mathbf{G})\tilde{\rho}_{cmp}(\mathbf{G})] + \\ \frac{1}{2} \iint \frac{(\rho_{cmp}(\mathbf{r})-\tilde{\rho}_{cmp}(\mathbf{r}))(\rho_{cmp}(\mathbf{r}')-\tilde{\rho}_{cmp}(\mathbf{r}'))} {|\mathbf{r}-\mathbf{r}'|} d\mathbf{r} d\mathbf{r}'$$

$$E_{cmp\text{-}pw}= \Omega \sum_{\mathbf{G} \neq 0} \frac{4\pi}{|\mathbf{G}|^2} \rho_{cmp}(\mathbf{G}) \tilde{\rho}(\mathbf{G})$$

In the first two formulas the first terms are computed using plane-waves and the second terms are computed using Gaussian two center integrals. The smooth local potential in the $E_{cmp\text{-}vloc}$ term is the Fourier transform of

$$\tilde{V}_{local}(\mathbf{r})=- \sum_I Z_I \frac{ erf(\frac{ |\mathbf{r}-\mathbf{R}_I | }{\tilde{\sigma}_I })} {|r-R_I |}$$

The stiff and smooth compensation charge densities in the above formula are

$$\rho_{cmp}(r)=\sum_I\sum_{lm} Q_{lm}^I g_{lm}^(\sigma_I) (r-R_I)$$
$$\tilde{\rho}_{cmp}(r)=\sum_I \sum_{lm} Q_{lm}^I g_{lm}^(\tilde{\sigma}_I) (r-R_I)$$

where

$$Q_{lm}^I = \sum_i \sum_{\alpha\beta} <\tilde{\psi}_i|\tilde{p}_\alpha^I > <\tilde{p}_\beta^I|\tilde{\psi}_i > \tau_{l_\alpha m_\alpha, l_\beta m_\beta}^{lm} (q_{comp}^I)_{\alpha\beta}^I$$

The decay parameter σ_I is defined the same as above, and$\tilde{\sigma}\)_I$ is defined to be smooth enough in order that $\tilde{\rho}_{cmp}(r)$ and $\tilde{V}\)_{local}(r)$ can readily be expanded in terms of plane-waves.

The final three terms are the energies that contain the core densities

$$E_{valence\text{-}core}=\sum_{i}\sum_{I}\sum_{\alpha \beta}<\tilde{\psi}_{i} | \tilde{p}_{\alpha}^{I} > (V_{valence\text{-}core})_{\alpha \beta}^{I} <\tilde{p}_\beta^{I} | \tilde{\psi}_{i} >$$
$$E_{kinetic\text{-}core}=\sum_c \int_0^\infty \left[(\varphi_{n_c l_c}^I (r))^{'} (\varphi_{n_c l_c}^{I} (r))^{'} + l_c (l_c+1) \frac{\varphi_{n_c l_c}^{I} (r) \varphi_{n_c l_c}^{I} (r)}{r^{2}} \right] dr$$
$$E_{ion\text{-}core}= \sum_I \frac{1}{2} \iint \frac{\rho_c^{I} (r) \rho_c^{I} (r')}{|r-r'|} drdr^{'} - \int \frac{\rho_c^{I} (r)}{|r|} (Z_I+Z_I^{core}) dr$$

The matrix elements contained in the above formula are

$$(t_{atom}^{I})_{\alpha \beta}= {\delta_{m_\alpha m_\beta} \delta_{l_\alpha l_\beta }}2 \int_0^{r_{cut}^I} \left[(\varphi_{n_\alpha l_\alpha}^{I} (r))' (\varphi_{n_\beta l_\beta}^{I} (r))' - (\tilde{\varphi}_{n_\alpha l_\alpha}^{I} (r))' (\tilde{\varphi}_{n_\beta l_\beta}^{I} (r))' + l_\alpha (l_\alpha+1) \frac{ \varphi_{n_\alpha l_\alpha}^{I}(r) \varphi_{n_\beta l_\beta}^{I}(r) - \tilde{\varphi}_{n_\alpha l_\alpha}^{I}(r) \tilde{\varphi}_{n_\beta l_\beta}^I }{r^2 } \right] dr$$
$$(u_{atom}^I)_{\alpha \beta} =\frac{Z_I}{4\pi} (V_{comp}^I)_{\alpha \beta}^{l=0} + \frac{2Z_I}{(\sqrt{2\pi} \sigma_I } (q_comp^I)_{\alpha \beta}^(l=0) +\delta_{m_\alpha m_\beta } \delta_{l_\alpha l_\beta } \int_0^{r_{cut}^I} [\varphi_{n_\alpha l_\alpha}^I (r) \varphi_(n_\beta l_\beta)^I (r) \frac{-Z_I}{r} + \tilde{\varphi}_{n_\alpha l_\alpha}^I (r) \tilde{\varphi}_{n_\beta l_\beta}^I (r) (-v_{ps}^I (r))] dr$$

$$(V_{Heff}^I)_{\alpha \beta \mu \nu}^I = (V_H^I)_{\alpha\beta\mu\nu}^I - 2(V_{comp}^I)_{\alpha\beta}^I (q_{comp}^I)_{\mu\nu}^I -(v_g^I)^I (q_{comp}^I)_{\alpha\beta}^I (q_{comp}^I)_{\mu\nu}^I$$
$$(V_H^I)_{\alpha\beta\mu\nu}^I = \frac{4\pi}{2l+1} \int_0^{r_{cut}^I} \int_0^{r_{cut}^I} \left(\frac{r_{<}^I}{r_{>}^{l+1} } \right)(\varphi_{n_\alpha l_\alpha } (r) \varphi_{n_\beta l_\beta } (r) \varphi_{n_\mu l_\mu } (r) \varphi_{n_\nu l_\nu } (r) - \tilde{\varphi}_{n_\alpha l_\alpha } (r) \tilde{\varphi}_(n_\beta l_\beta) (r) \tilde{\varphi}_{n_\mu l_\mu } (r) \tilde{\varphi}_{n_\nu l_\nu } (r)) drdr'$$
$$(V_{comp}^I)_{\alpha\beta}^I = \frac{4\pi}{2l+1} \int_0^{r_{cut}^I} \int_0^{r_{cut}^I} \tilde{\varphi}_{n_\alpha l_\alpha } (r) \tilde{\varphi}_{n_\beta l_\beta } (r) \left(\frac{r_{<}^I}{r_{>}^{l+1} } \right) g_I^I (r') r'^2 drdr'$$
$$(q_{comp}^I)_{\alpha\beta}^I = \int_0^\infty r^I \left(\varphi_{n_\alpha l_\alpha } (r) \varphi_{n_\beta l_\beta } (r)- \tilde{\varphi}_{n_\alpha l_\alpha } (r) \tilde{\varphi}_{n_\beta l_\beta } (r) \right) dr$$

\[(v_g^l)^l =\frac{4\sqrt{2\pi}}{ (2l+1)(2l+1)!! \sigma_l^{2l+1} }\]
\[\tau_{l_\alpha m_\alpha,l_\beta m_\beta}^{lm}= \int_0^{2\pi} \int_0^{\pi} T_{lm} (\theta,\phi) T_{l_\alpha m_\alpha } (\theta,\phi) T_{l_\beta m_\beta } (\theta,\phi) \sin(\theta) d\theta d\phi\]

12.1.2.2 Exchange-Correlation Potentials

12.1.2.2.1 DFT + U Corrections

TO DO

```
nwpw
  uterm d 0.13634 0.0036749 1
end
```

12.1.2.2.2 Langreth style vdw and vdw van der Wall functionals

These potenials that are used to augment standard exchange-correlation potentials area calculated from a double integral over a nonlocal interaction kernel, \(\phi(\mathbf{r},\mathbf{r}^{'})\)

\[E_{vdw} = \int \rho(\mathbf{r}) \phi(\mathbf{r},\mathbf{r}^{'}) \rho(\mathbf{r}^{'}) d\mathbf{r} d\mathbf{r}^{'} \]

that is evaluated using the fast Fourier transformation method of Roman-Perez and Soler.

G. Roman-Perez and J. M. Soler, Phys. Rev. Lett. 103, 096102 (2009).

Langreth vdw and vdw2 van der Wall functionals are currently available for the BEEF, PBE96, revPBE, PBEsol, BLYP, PBE0, revPBE0, HSE, and B3LYP exchange-correlation functionals. To use them the following keywords BEEF-vdw, BEEF-vdw2, PBE96-vdw, PBE96-vdw2, BLYP-vdw, BLYP-vdw2, revPBE-vdw, revPBE-vdw, PBEsol-vdw PBEsol-vdw2, PBE0-vdw, PBE0-vdw2, revPBE0-vdw, revPBE0-vdw2, HSE-vdw, HSE-vdw2, B3LYP-vdw, and B3LYP-vdw2 can be used in the XC input directive, e.g.

```
nwpw
  xc beef-vdw
end
```

```
nwpw
  xc beef-vdw2
end
```

the vdw and vdw2 functionals are defined in

(vdw) Dion M, Rydberg H, Schröder E, Langreth DC, Lundqvist BI. Van der Waals density functional for general geometries. Physical review letters. 2004 Jun 16;92(24):246401.

(vdw2) K. Lee, E. D. Murray, L. Kong, B. I. Lundqvist, and D. C. Langreth, Phys. Rev. B 82, 081101 (2010).

12.1.2.2.3 Grimme Dispersion Corrections

Grimme dispersion corrections are currently available for the PBE96, revPBE, PBEsol, BLYP, PBE0, revPBE0, HSE, and B3LYP exchange-correlation functionals. To use them the following keywords PBE96-Grimme2, PBE96-Grimme3, PBE96-Grimme4, BLYP-Grimme2, BLYP-Grimme3, BLYP-Grimme4, revPBE-Grimme2, revPBE-Grimme3, revPBE-Grimme4, PBEsol-Grimme2, PBEsol-Grimme3, PBEsol-Grimme4, PBE0-Grimme2, PBE0-Grimme3, PBE0-Grimme4, revPBE0-Grimme2, revPBE0-Grimme3, revPBE0-Grimme4, HSE-Grimme2, HSE-Grimme3, HSE-Grimme4, B3LYP-Grimme2, B3LYP-Grimme3, and B3LYP-Grimme4 can be used in the XC input directive, e.g.

```
nwpw
  xc pbe96-grimme2
end
```

In these functionals Grimme2, Grimme3 and Grimme4 are defined in the following papers by S. Grimme.

Grimme2 - Commonly known as DFT-D2, S. Grimme, J. Comput. Chem., 27 (2006), 1787-1799.

Grimme3 - Commonly known as DFT-D3, S. Grimme, J. Antony, S. Ehrlich and H. Krieg A consistent and accurate ab initio parameterization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu, J. Chem. Phys, 132 (2010), 154104

Grimme4 - Commonly known as DFT-D3 with BJ damping. This correction augments the Grimme3 correction by including BJ-damping, S. Grimme, J. Antony, S. Ehrlich and H. Krieg A consistent and accurate ab initio parameterization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu, J. Chem. Phys, 132 (2010), 154104. S. Grimme, S. Ehrlich and L. Goerigk, J. Comput. Chem, 32 (2011), 1456-1465. This correction augments the Grimme3 correction by including BJ-damping.

If these functionals are used in a publication please include in your citations the references to Grimme's work.

12.1.2.2.4 Using Exchange-Correlation Potentials Available in the DFT Module

(*Warning - To use this capability in NWChem 6.6 the user must explicitly include the nwxc module in the NWCHEM_MODULES list when compiling. Unfortunately, there was too much uncertainty in how the nwxc computed higher-order derivatives used by some of the functionality in nwdft module to include it in a release for all the functionality in NWChem. We are planning to have a debug release in winter 2016 to take fix this problem. This capability is still included by default in NWChem 6.5*)

The user has the option of using many of the exchange-correlation potentials available in DFT Module (see Section XC and DECOMP – Exchange-Correlation Potentials).

```
XC [[acm] [b3lyp] [beckehandh] [pbe0] [bhlyp]\
   [becke97]  [becke97-1] [becke97-2] [becke97-3] [becke98] [hcth] [hcth120] [hcth147] \
   [hcth407] [becke97gga1] [hcth407p] \
   [optx] [hcthp14] [mpw91] [mpw1k] [xft97] [cft97] [ft97] [op] [bop] [pbeop]\
   [HFexch <real prefactor default 1.0>] \
   [becke88 [nonlocal] <real prefactor default 1.0>] \
   [xperdew91 [nonlocal] <real prefactor default 1.0>] \
   [xpbe96 [nonlocal] <real prefactor default 1.0>] \
   [gill96 [nonlocal] <real prefactor default 1.0>] \
   [lyp <real prefactor default 1.0>] \
   [perdew81 <real prefactor default 1.0>] \
   [perdew86 [nonlocal] <real prefactor default 1.0>] \
   [perdew91 [nonlocal] <real prefactor default 1.0>] \
   [cpbe96 [nonlocal] <real prefactor default 1.0>] \
   [pw91lda <real prefactor default 1.0>] \
   [slater <real prefactor default 1.0>] \
   [vwn_1 <real prefactor default 1.0>] \
   [vwn_2 <real prefactor default 1.0>] \
   [vwn_3 <real prefactor default 1.0>] \
   [vwn_4 <real prefactor default 1.0>] \
   [vwn_5 <real prefactor default 1.0>] \
   [vwn_1_rpa <real prefactor default 1.0>]]
```

These functional can be invoked by prepending the "new" directive before the exchange correlation potetntials in the input directive, XC new slater vwn_5.

That is, this statement in the input file

```
nwpw
 XC new slater vwn_5
end
task pspw energy
```

Using this input the user has ability to include only the local or nonlocal contributions of a given functional. The user can also specify a multiplicative prefactor (the variable `prefactor` in the input) for the local/nonlocal component or total (for more details see Section XC and DECOMP – Exchange-Correlation Potentials). An example of this might be,

```
XC new becke88 nonlocal 0.72
```

The user should be aware that the Becke88 local component is simply the Slater exchange and should be input as such.

Any combination of the supported exchange functional options can be used. For example the popular Gaussian B3

exchange could be specified as:

```
XC new slater 0.8 becke88 nonlocal 0.72 HFexch 0.2
```

Any combination of the supported correlation functional options can be used. For example B3LYP could be specified as:

```
XC new vwn_1_rpa 0.19 lyp 0.81 HFexch 0.20 slater 0.80 becke88 nonlocal 0.72
```

and X3LYP as:

```
xc new vwn_1_rpa 0.129 lyp 0.871 hfexch 0.218 slater 0.782 \
becke88 nonlocal 0.542  xperdew91 nonlocal 0.167
```

12.1.2.2.5 Exact Exchange

12.1.2.2.6 Self-Interaction Corrections

The SET directive is used to specify the molecular orbitals contribute to the self-interaction-correction (SIC) term.

```
set pspw:SIC_orbitals <integer list_of_molecular_orbital_numbers>
```

This defines only the molecular orbitals in the list as SIC active. All other molecular orbitals will not contribute to the SIC term. For example the following directive specifies that the molecular orbitals numbered 1,5,6,7,8, and 15 are SIC active.

```
set pspw:SIC_orbitals 1 5:8 15
```

or equivalently

```
set pspw:SIC_orbitals 1 5 6 7 8 15
```

The following directive turns on self-consistent SIC.

```
set pspw:SIC_relax .false. # Default - Perturbative SIC calculation
set pspw:SIC_relax .true. # Self-consistent SIC calculation
```

Two types of solvers can be used and they are specified using the following SET directive

```
set pspw:SIC_solver_type 1 # Default - cutoff coulomb kernel
set pspw:SIC_solver_type 2 # Free-space boundary condition kernel
```

The parameters for the cutoff coulomb kernel are defined by the following SET directives:

```
set pspw:SIC_screening_radius <real rcut>
set pspw:SIC_screening_power <real rpower>
```

12.1.2.3 Wannier

The pspw wannier task is generate maximally localized (Wannier) molecular orbitals. The algorithm proposed by Silvestrelli et al is use to generate the Wannier orbitals.

Input to the Wannier task is contained within the Wannier sub-block.

```
NWPW
...
 Wannier
   ...
 END
...
END
```

To run a Wannier calculation the following directive is used:

```
TASK PSPW Wannier
```

Listed below is the format of a Wannier sub-block.

```
NWPW
...
 Wannier
  OLD_WAVEFUNCTION_FILENAME <string input_wavefunctions default input_movecs>
  NEW_WAVEFUNCTION_FILENAME <string output_wavefunctions default input_movecs>
 END
...
END
```

The following list describes the input for the Wannier sub-block.

- input_wavefunctions - name of pspw wavefunction file.
- output_wavefunctions - name of pspw wavefunction file that will contain the Wannier orbitals.

12.1.2.4 Mulliken Analysis

12.1.2.5 Density of States

The “dos” option is used to turn on a density of states analysis. This option can be specified without additional parameters, i.e.

```
nwpw
  dos
end
```

in which case default values are used, or it can be specified with additional parameters, e.g.

```
nwpw
   dos 0.002 700 -0.80000 0.8000
end
```

The parameters are

```
nwpw
 dos  [<alpha> <npoints> <emin> <emax>]
end
```

where

- alpha value for the broadening the eigenvalues, default 0.05/27.2116 au
- npoints number of plotting points in dos files, default 500
- emin minimum energy in dos plots, default min(eigenvalues)-0.1 au
- emax maximimum energy in dos plots, default max(eigenvalues)+0.1 au

The units for dos parameters are in atomic units. Note that if virtual states are specified in the pspw calculation then the virtual density of states will also be generated in addition to the filled density of states.

The following files are generated and written to the permanent_dir for restricted calculations

- file_prefix.smear_dos_both - total density of states
- file_prefix.smear_fdos_both - density of states of filled states
- file_prefix.smear_vdos_both - density of states of virtual states

For unrestricted calculations

- file_prefix.smear_dos_alpha - total density of states for up electrons
- file_prefix.smear_dos_beta - total density of states for down electrons

- file_prefix.smear_fdos_alpha - density of states for filled up electrons
- file_prefix.smear_fdos_beta - density of states for filled down electrons
- file_prefix.smear_vdos_alpha - density of states for virtual up electrons
- file_prefix.smear_vdos_beta - density of states for virtual down electrons

The nwpw:dos:actlist variable is used to specify the atoms used to determine weights for dos generation. If the variable is not set then all the atoms are used, e.g.

```
set nwpw:dos:actlist 1 2 3 4
```

12.1.2.6 Projected Density of States

For projected density of states the “Mulliken” keyword needs to be set, e.g.

```
nwpw
  Mulliken
  dos
end
```

The following additional files are generated and written to the permanent_dir for restricted calculations

- file_prefix.mulliken_dos_both_s - total s projected density of restricted states
- file_prefix.mulliken_fdos_both_s - s projected density of states of filled restricted states
- file_prefix.mulliken_vdos_both_s - s projected density of states of virtual restricted states
- file_prefix.mulliken_dos_both_p - total p projected density of states
- file_prefix.mulliken_fdos_both_p - p projected density of states of filled states
- file_prefix.mulliken_vdos_both_p - p projected density of states of virtual states

…

- file_prefix.mulliken_dos_both_all - total of projected density of filled and virtual restricted states
- file_prefix.mulliken_fdos_both_all - total of projected density of filled restricted states
- file_prefix.mulliken_vdos_both_all - total of projected density of states of virtual restricted states

Similarly for unrestricted calculations

- file_prefix.mulliken_dos_alpha_s - total s projected density of up states
- file_prefix.mulliken_fdos_alpha_s - s projected density of states of filled up states
- file_prefix.mulliken_vdos_alpha_s - s projected density of states of virtual up states
- file_prefix.mulliken_dos_alpha_p - total p projected density of up states
- file_prefix.mulliken_fdos_alpha_p - p projected density of states of filled up states
- file_prefix.mulliken_vdos_alpha_p - p projected density of states of virtual up states

…

- file_prefix.mulliken_dos_alpha_all - total of projected density of filled up states
- file_prefix.mulliken_fdos_alpha_all - total of projected density of filled up states
- file_prefix.mulliken_vdos_alpha_all - total of projected density of states of virtual up states

…

- file_prefix.mulliken_dos_beta_s - total s projected density of down states

- file_prefix.mulliken_fdos_beta_s - s projected density of states of filled down states
- file_prefix.mulliken_vdos_beta_s - s projected density of states of virtual down states
- file_prefix.mulliken_dos_beta_p - total p projected density of down states
- file_prefix.mulliken_fdos_beta_p - p projected density of states of filled down states
- file_prefix.mulliken_vdos_beta_p - p projected density of states of virtual down states

…

- file_prefix.mulliken_dos_beta_all - total of projected density of filled down states
- file_prefix.mulliken_fdos_beta_all - total of projected density of filled down states
- file_prefix.mulliken_vdos_beta_all - total of projected density of states of virtual down states

12.1.2.7 Point Charge Analysis

The MULLIKEN option can be used to generate derived atomic point charges from a plane-wave density. This analysis is based on a strategy suggested in the work of P.E. Blochl, J. Chem. Phys. vol. 103, page 7422 (1995). In this strategy the low-frequency components a plane-wave density are fit to a linear combination of atom centered Gaussian functions.

The following SET directives are used to define the fitting.

```
set nwpw_APC:Gc <real Gc_cutoff> # specifies the maximum frequency component of the density to be used in the fitting in units of au.
set nwpw_APC:nga <integer number_gauss> # specifies the the number of Gaussian functions per atom.
set nwpw_APC:gamma <real gamma_list> # specifies the decay lengths of each atom centered Gaussian.
```

We suggest using the following parameters.

```
set nwpw_APC:Gc 2.5
set nwpw_APC:nga 3
set nwpw_APC:gamma 0.6 0.9 1.35
```

12.1.2.8 PSPW_DPLOT: Generate Gaussian Cube Files

The pspw dplot task is used to generate plots of various types of electron densities (or orbitals) of a molecule. The electron density is calculated on the specified set of grid points from a PSPW calculation. The output file generated is in the Gaussian Cube format. Input to the DPLOT task is contained within the DPLOT sub-block.

```
NWPW
...
 DPLOT
   ...
 END
...
END
```

To run a DPLOT calculation the following directive is used:

```
TASK PSPW PSPW_DPLOT
```

Listed below is the format of a DPLOT sub-block.

```
NWPW
...
 DPLOT
  VECTORS <string input_wavefunctions default input_movecs>
  DENSITY [total||diff||alpha||beta||laplacian||potential default total]
        <string density_name no default>
  ELF [restricted|alpha|beta] <string elf_name no default>
  ORBITAL <integer orbital_number no default> <string orbital_name no default>
  [LIMITXYZ [units <string Units default au>]
  <real X_From> <real X_To> <integer No_Of_Spacings_X>
  <real Y_From> <real Y_To> <integer No_Of_Spacings_Y>
  <real Z_From> <real Z_To> <integer No_Of_Spacings_Z>]
  NCELL <integer nx default 0> <integer ny default 0> <integer nz default 0>
  POSITION_TOLERANCE <real rtol default 0.001>
 END
...
END
```

The following list describes the input for the DPLOT sub-block.

```
VECTORS <string input_wavefunctions default input_movecs>
```

This sub-directive specifies the name of the molecular orbital file. If the second file is optionally given the density is computed as the difference between the corresponding electron densities. The vector files have to match.

```
DENSITY [total||difference||alpha||beta||laplacian||potential default total]
      <string density_name no default>
```

This sub-directive specifies, what kind of density is to be plotted. The known names for total, difference, alpha, beta, laplacian, and potential.

```
ELF [restricted|alpha|beta] <string elf_name no default>
```

This sub-directive specifies that an electron localization function (ELF) is to be plotted.

```
ORBITAL <integer orbital_number no default> <string orbital_name no default>
```

This sub-directive specifies the molecular orbital number that is to be plotted.

```
LIMITXYZ [units <string Units default angstroms>]
<real X_From> <real X_To> <integer No_Of_Spacings_X>
<real Y_From> <real Y_To> <integer No_Of_Spacings_Y>
<real Z_From> <real Z_To> <integer No_Of_Spacings_Z>
```

By default the grid spacing and the limits of the cell to be plotted are defined by the input wavefunctions. Alternatively the user can use the LIMITXYZ sub-directive to specify other limits. The grid is generated using No_Of_Spacings + 1 points along each direction. The known names for Units are angstroms, au and bohr.

12.1.3 Band Tasks: Multiple k-point Calculations

All input to the Band Tasks is contained within the compound NWPW block,

```
NWPW
 ...
END
```

To perform an actual calculation a Task Band directive is used (Section Task).

```
Task Band
```

Once a user has specified a geometry, the Band module can be invoked with no input directives (defaults invoked throughout). There are sub-directives which allow for customized application; those currently provided as options for the Band module are:

```
NWPW
 CELL_NAME <string cell_name default cell_default>
 ZONE_NAME <string zone_name default zone_default>
 INPUT_WAVEFUNCTION_FILENAME <string input_wavefunctions default input_movecs>
 OUTPUT_WAVEFUNCTION_FILENAME <string output_wavefunctions default input_movecs>
 FAKE_MASS <real fake_mass default 400000.0>
 TIME_STEP <real time_step default 5.8>
 LOOP <integer inner_iteration outer_iteration default 10 100>
 TOLERANCES <real tole tolc default 1.0e-7 1.0e-7>
 CUTOFF <real cutoff>
 ENERGY_CUTOFF <real ecut default (see input description)>
 WAVEFUNCTION_CUTOFF <real wcut default (see input description)>
 EWALD_NCUT <integer ncut default 1>]
 EWALD_RCUT <real rcut default (see input description)>

 XC (Vosko     || LDA         || PBE96       || revPBE      || PBEsol         ||  `
            || HSE  || default Vosko)                                              `
  #Note that HSE is the only hybrid functional implemented in BAND

 DFT||ODFT||RESTRICTED||UNRESTRICTED
 MULT <integer mult default 1>
 CG
 LMBFGS
 SCF [Anderson|| simple || Broyden]
 [CG || RMM-DIIS] [density || potential]
 [ALPHA real alpha default 0.25]
 [ITERATIONS integer inner_iterations default 5]
 [OUTER_ITERATIONS integer outer_iterations default 0]

 SIMULATION_CELL  [units <string units default bohrs>]
   ... (see input description)
 END
 BRILLOUIN_ZONE
   ... (see input description)
 END
 MONKHORST-PACK <real n1 n2 n3 default 1 1 1>
 BAND_DPLOT
   ... (see input description)
 END
 MAPPING <integer mapping default 1>
 SMEAR <sigma default 0.001>
 [TEMPERATURE <temperature>]
 [FERMI || GAUSSIAN || MARZARI-VANDERBILT default FERMI]
 [ORBITALS <integer orbitals default 4>]
END
```

The following list describes these keywords.

- cell_name - name of the simulation_cell named cell_name . See Simulation Cell.
- input_wavefunctions - name of the file containing one-electron orbitals
- output_wavefunctions - name that will point to file containing the one-electron orbitals at the end of the run.
- fake_mass - value for the electronic fake mass\(\mu\). This parameter is not presently used in a conjugate gradient simulation
- time_step - value for the time step\(\Delta t\). This parameter is not presently used in a conjugate gradient simulation.
- inner_iteration - number of iterations between the printing out of energies and tolerances
- outer_iteration - number of outer iterations
- tole - value for the energy tolerance.
- tolc - value for the one-electron orbital tolerance.
- cutoff - value for the cutoff energy used to define the wavefunction. In addition using the CUTOFF keyword automatically sets the cutoff energy for the density to be twice the wavefunction cutoff.
- ecut - value for the cutoff energy used to define the density. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- wcut - value for the cutoff energy used to define the one-electron orbitals. Default is set to be the maximum value that will fix within the simulation_cell cell_name .
- ncut - value for the number of unit cells to sum over (in each direction) for the real space part of the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.

- rcut - value for the cutoff radius used in the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.

Default set to be \(\frac{MIN(\left| \vec{a_i} \right|)}{\pi}, i=1,2,3\).

- (Vosko || PBE96 || revPBE) - Choose between Vosko et al's LDA parameterization or the orginal and revised Perdew, Burke, and Ernzerhof GGA functional.
- SIMULATION_CELL (see section -sec:pspw_cell-)
- BRILLOUIN_ZONE (see section -sec:band_brillouin_zone-)
- MONKHORST-PACK - Alternatively, the MONKHORST-PACK keyword can be used to enter a MONKHORST-PACK sampling of the Brillouin zone.
- smear - value for smearing broadending
- temperature - same as smear but in units of K.
- CG - optional keyword which sets the minimizer to 1
- LMBFGS - optional keyword which sets the minimizer to 2
- SCF - optional keyword which sets the minimizer to be a band by band minimizer. Several options are available for setting the density or potential mixing, and the type of Kohn-Sham minimizer.

12.1.3.1 Brillouin Zone

To supply the special points of the Brillouin zone, the user defines a brillouin_zone sub-block within the NWPW block. Listed below is the format of a brillouin_zone sub-block.

```
NWPW
...
 BRILLOUIN_ZONE
   ZONE_NAME <string name default zone_default>
   (KVECTOR <real k1 k2 k3 no default> <real weight default (see input description)>
    ...)
 END
...
END
```

The user enters the special points and weights of the Brillouin zone. The following list describes the input in detail.

- name - user-supplied name for the simulation block.
- k1 k2 k3 - user-supplied values for a special point in the Brillouin zone.
- weight - user-supplied weight. Default is to set the weight so that the sum of all the weights for the entered special points adds up to unity.

12.1.3.1.1 Band Structure Paths

SC: gamma, m, r, x

FCC: gamma, k, l, u, w, x

BCC: gamma, h, n, p

Rhombohedral: not currently implemented

Hexagonal: gamma, a, h, k, l, m

Simple Tetragonal: gamma, a, m, r, x, z

Simple Orthorhombic: gamma, r, s, t, u, x, y, z

Body-Centered Tetragonal: gamma, m, n, p, x

12.1.3.1.2 Special Points of Different Space Groups (Conventional Cells)

(1) P1

(2) P-1

(3)

12.1.3.2 Screened Exchange

12.1.3.3 Density of States and Projected Density of States

The “dos” option is used to calculate density of states using broadening of the eigenvalues . This option can be specified without additional parameters, i.e.

```
nwpw
  dos
end
```

in which case default values are used, or it can be specified with additional parameters, e.g.

```
nwpw
  dos 0.002 700 -0.80000 0.8000
end
```

The parameters are

```
nwpw
 dos  [<alpha> <npoints> <emin> <emax>]
end
```

where

- alpha - value for the broadening the eigenvalues, default 0.05/27.2116 au
- npoints - number of plotting points in dos files, default 500
- emin - minimum energy in dos plots, default min(eigenvalues)-0.1 au
- emax - maximimum energy in dos plots, default max(eigenvalues)+0.1 au

The units for dos parameters are in atomic units. Note that if virtual states are specified in the pspw calculation then the virtual density of states will also be generated in addition to the filled density of states.

The following files are generated and written to the permanent_dir for restricted calculations

- file_prefix.smear_dos_both - total density of states
- file_prefix.smear_fdos_both - density of states of filled states
- file_prefix.smear_vdos_both - density of states of virtual states

For unrestricted calculations

- file_prefix.smear_dos_alpha - total density of states for up electrons
- file_prefix.smear_dos_beta - total density of states for down electrons
- file_prefix.smear_fdos_alpha - density of states for filled up electrons
- file_prefix.smear_fdos_beta - density of states for filled down electrons
- file_prefix.smear_vdos_alpha - density of states for virtual up electrons
- file_prefix.smear_vdos_beta - density of states for virtual down electrons

The nwpw:dos:actlist variable is used to specify the atoms used to determine weights for dos generation. If the variable is not set then all the atoms are used, e.g.

```
set nwpw:dos:actlist 1 2 3 4
```

For projected density of states the “Mulliken” keyword needs to be set, e.g.

```
nwpw
  Mulliken
  dos
end
```

The following additional files are generated and written to the permanent_dir for restricted calculations

- file_prefix.mulliken_dos_both_s - total s projected density of restricted states
- file_prefix.mulliken_fdos_both_s - s projected density of states of filled restricted states
- file_prefix.mulliken_vdos_both_s - s projected density of states of virtual restricted states
- file_prefix.mulliken_dos_both_p - total p projected density of states
- file_prefix.mulliken_fdos_both_p - p projected density of states of filled states
- file_prefix.mulliken_vdos_both_p - p projected density of states of virtual states

…

- file_prefix.mulliken_dos_both_all - total of projected density of filled and virtual restricted states
- file_prefix.mulliken_fdos_both_all - total of projected density of filled restricted states
- file_prefix.mulliken_vdos_both_all - total of projected density of states of virtual restricted states

Similarly for unrestricted calculations

- file_prefix.mulliken_dos_alpha_s - total s projected density of up states
- file_prefix.mulliken_fdos_alpha_s - s projected density of states of filled up states
- file_prefix.mulliken_vdos_alpha_s - s projected density of states of virtual up states
- file_prefix.mulliken_dos_alpha_p - total p projected density of up states
- file_prefix.mulliken_fdos_alpha_p - p projected density of states of filled up states
- file_prefix.mulliken_vdos_alpha_p - p projected density of states of virtual up states

…

- file_prefix.mulliken_dos_alpha_all - total of projected density of filled up states
- file_prefix.mulliken_fdos_alpha_all - total of projected density of filled up states
- file_prefix.mulliken_vdos_alpha_all - total of projected density of states of virtual up states

…

- file_prefix.mulliken_dos_beta_s - total s projected density of down states
- file_prefix.mulliken_fdos_beta_s - s projected density of states of filled down states
- file_prefix.mulliken_vdos_beta_s - s projected density of states of virtual down states
- file_prefix.mulliken_dos_beta_p - total p projected density of down states
- file_prefix.mulliken_fdos_beta_p - p projected density of states of filled down states
- file_prefix.mulliken_vdos_beta_p - p projected density of states of virtual down states

…

- file_prefix.mulliken_dos_beta_all - total of projected density of filled down states
- file_prefix.mulliken_fdos_beta_all - total of projected density of filled down states
- file_prefix.mulliken_vdos_beta_all - total of projected density of states of virtual down states

12.1.3.4 Two-Component Wavefunctions (Spin-Orbit ZORA)

12.1.3.5 BAND_DPLOT: Generate Gaussian Cube Files

The BAND BAND_DPLOT task is used to generate plots of various types of electron densities (or orbitals) of a crystal. The electron density is calculated on the specified set of grid points from a Band calculation. The output file generated is in the Gaussian Cube format. Input to the BAND_DPLOT task is contained within the BAND_DPLOT sub-block.

```
NWPW
...
 BAND_DPLOT
   ...
 END
...
END
```

To run a BAND_DPLOT calculation the following directive is used:

```
TASK BAND BAND_DPLOT
```

Listed below is the format of a BAND_DPLOT sub-block.

```
NWPW
...
 BAND_DPLOT
   VECTORS <string input_wavefunctions default input_movecs>
   DENSITY [total||difference||alpha||beta||laplacian||potential default total] <string density_name no default>
   ELF [restricted|alpha|beta] <string elf_name no default>
   ORBITAL (density || real || complex default density)
         <integer orbital_number no default>
         <integer brillion_number default 1>
         <string orbital_name no default>
   [LIMITXYZ [units <string Units default angstroms>]
    <real X_From> <real X_To> <integer No_Of_Spacings_X>
    <real Y_From> <real Y_To> <integer No_Of_Spacings_Y>
    <real Z_From> <real Z_To> <integer No_Of_Spacings_Z>]
 END
...
END
```

The following list describes the input for the BAND_DPLOT sub-block.

```
VECTORS <string input_wavefunctions default input_movecs>
```

This sub-directive specifies the name of the molecular orbital file. If the second file is optionally given the density is computed as the difference between the corresponding electron densities. The vector files have to match.

```
DENSITY [total||difference||alpha||beta||laplacian||potential default total] <string density_name no default>
```

This sub-directive specifies, what kind of density is to be plotted. The known names for total, difference, alpha, beta, laplacian, and potential.

```
ELF [restricted|alpha|beta] <string elf_name no default>
```

This sub-directive specifies that an electron localization function (ELF) is to be plotted.

```
ORBITAL (density || real || complex default density) <integer orbital_number no default><integer brillion_number default 1> <string orbital_name no default>
```

This sub-directive specifies the molecular orbital number that is to be plotted.

```
LIMITXYZ [units <string Units default angstroms>]
<real X_From> <real X_To> <integer No_Of_Spacings_X>
<real Y_From> <real Y_To> <integer No_Of_Spacings_Y>
<real Z_From> <real Z_To> <integer No_Of_Spacings_Z>
```

By default the grid spacing and the limits of the cell to be plotted are defined by the input wavefunctions. Alternatively the user can use the LIMITXYZ sub-directive to specify other limits. The grid is generated using No_Of_Spacings + 1 points along each direction. The known names for Units are angstroms, au and bohr.

12.1.4 Car-Parrinello

The Car-Parrinello task is used to perform ab initio molecular dynamics using the scheme developed by Car and Parrinello. In this unified ab initio molecular dynamics scheme the motion of the ion cores is coupled to a fictitious motion for the Kohn-Sham orbitals of density functional theory. Constant energy or constant temperature simulations can be performed. A detailed description of this method is described in section Car-Parrinello Scheme for Ab Initio Molecular Dynamics.

Input to the Car-Parrinello simulation is contained within the Car-Parrinello sub-block.

```
NWPW
...
 Car-Parrinello
   ...
 END
...
END
```

To run a Car-Parrinello calculation the following directives are used:

```
TASK PSPW Car-Parrinello
 TASK BAND Car-Parrinello
 TASK PAW Car-Parrinello
```

The Car-Parrinello sub-block contains a great deal of input, including pointers to data, as well as parameter input. Listed below is the format of a Car-Parrinello sub-block.

```
NWPW
...
 Car-Parrinello
   CELL_NAME <string cell_name default 'cell_default'>
   INPUT_WAVEFUNCTION_FILENAME <string input_wavefunctions default file_prefix.movecs>
   OUTPUT_WAVEFUNCTION_FILENAME <string output_wavefunctions default file_prefix.movecs>
   INPUT_V_WAVEFUNCTION_FILENAME <string input_v_wavefunctions default file_prefix.vmovecs>
   OUTPUT_V_WAVEFUNCTION_FILENAME <string output_v_wavefunctions default file_prefix.vmovecs>
   FAKE_MASS <real fake_mass default default 1000.0>
   TIME_STEP <real time_step default 5.0>
   LOOP <integer inner_iteration outer_iteration default 10 1>
   SCALING <real scale_c scale_r default 1.0 1.0>
   ENERGY_CUTOFF <real ecut default (see input description)>
   WAVEFUNCTION_CUTOFF <real wcut default (see input description)>
   EWALD_NCUT <integer ncut default 1>
   EWALD_RCUT <real rcut default (see input description)>
    XC (Vosko      || LDA          || PBE96        || revPBE       || HF           ||
        LDA-SIC    || LDA-SIC/2    || LDA-0.4SIC   || LDA-SIC/4    || LDA-0.2SIC   ||
        PBE96-SIC  || PBE96-SIC/2  || PBE96-0.4SIC  || PBE96-SIC/4  || PBE96-0.2SIC  ||
        revPBE-SIC || revPBE-SIC/2 || revPBE-0.4SIC || revPBE-SIC/4 || revPBE-0.2SIC ||
        PBE0       || revPBE0      || default Vosko)
   [Nose-Hoover <real Period_electron real Temperature_electron
                 real Period_ion      real Temperature_ion
                 integer Chainlength_electron integer Chainlength_ion default 100.0 298.15 100.0 298.15 1 1>]
   [TEMPERATURE <real Temperature_ion      real Period_ion
                 real Temperature_electron real Period_electron
                 integer Chainlength_ion integer Chainlength_electron default 298.15 1200 298.15 1200.0 1 1>]
   [SA_decay <real sa_scale_c sa_scale_r default 1.0 1.0>]
   XYZ_FILENAME <string xyz_filename default file_prefix.xyz>
   ION_MOTION_FILENAME <string ion_motion_filename default file_prefix.ion_motion
   EMOTION_FILENAME <string emotion_filename default file_prefix.emotion>
   HMOTION_FILENAME <string hmotion_filename nodefault>
   OMOTION_FILENAME <string omotion_filename nodefault>
   EIGMOTION_FILENAME <string eigmotion_filename nodefault>
 END
...
END
```

The following list describes the input for the Car-Parrinello sub-block.

- cell_name - name of the the simulation_cell named cell_name . See section Simulation Cell.
- input_wavefunctions - name of the file containing one-electron orbitals
- output_wavefunctions - name of the file that will contain the one-electron orbitals at the end of the run.
- input_v_wavefunctions - name of the file containing one-electron orbital velocities.
- output_v_wavefunctions - name of the file that will contain the one-electron orbital velocities at the end of the run.
- fake_mass - value for the electronic fake mass (\(\mu\)).
- time_step - value for the Verlet integration time step (\(Delta t\)).
- inner_iteration - number of iterations between the printing out of energies.
- outer_iteration - number of outer iterations
- scale_c - value for the initial velocity scaling of the one-electron orbital velocities.
- scale_r - value for the initial velocity scaling of the ion velocities.
- ecut - value for the cutoff energy used to define the density. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- wcut - value for the cutoff energy used to define the one-electron orbitals. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- ncut - value for the number of unit cells to sum over (in each direction) for the real space part of the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.
- rcut - value for the cutoff radius used in the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.

Default set to be \(\frac{MIN(\left| \vec{a_i} \right|)}{\pi}, i=1,2,3\).

- (Vosko || PBE96 || revPBE || …) - Choose between Vosko et al's LDA parameterization or the orginal and revised Perdew, Burke, and Ernzerhof GGA functional. In addition, several hybrid options.
- Nose-Hoover or Temperature - optional subblock which if specified causes the simulation to perform Nose-Hoover dynamics. If this subblock is not specified the simulation performs constant energy dynamics. See section -sec:pspw_nose- for a description of the parameters. Note that the Temperature subblock is just a reordering of the Nose-Hoover subblock.
  - Period_electron \(\equiv P_{electron}\) - estimated period for fictitious electron thermostat.
  - Temperature_electron \(\equiv T_{electron}\) - temperature for fictitious electron motion
  - Period_ion \(\equiv P_{ion}\) - estimated period for ionic thermostat
  - Temperature_ion \(\equiv T_{ion}\) - temperature for ion motion
  - Chainlength_electron - number of electron thermostat chains
  - Chainlength_ion - number of ion thermostat chains
- SA_decay - optional subblock which if specified causes the simulation to run a simulated annealing simulation. For simulated annealing to work the Nose-Hoover subblock needs to be specified. The initial temperature are taken from the Nose-Hoover subblock. See section -sec:pspw_nose- for a description of the parameters.
  - sa_scale_c \(\equiv \tau_{electron}\) - decay rate in atomic units for electronic temperature.
  - sa_scale_r \(\equiv \tau_{ionic}\) - decay rate in atomic units for the ionic temperature.
- xyz_filename - name of the XYZ motion file generated
- emotion_filename - name of the emotion motion file. See section EMOTION motion file for a description of the datafile.
- hmotion_filenameh - name of the hmotion motion file. See section HMOTION motion file for a description of the datafile.
- eigmotion_filename - name of the eigmotion motion file. See section EIGMOTION motion file for a description of the datafile.

- ion_motion_filename - name of the ion_motion motion file. See section ION_MOTION motion file for a description of the datafile.
- MULLIKEN - optional keyword which if specified causes an omotion motion file to be created.
- omotion_filename - name of the omotion motion file. See section OMOTION motion file for a description of the datafile.

When a DPLOT sub-block is specified the following SET directive can be used to output dplot data during a PSPW Car-Parrinello simulation:

```
set pspw_dplot:iteration_list <integer list_of_iteration_numbers>
```

The Gaussian cube files specified in the DPLOT sub-block are appended with the specified iteration number.

For example, the following directive specifies that at the 3,10,11,12,13,14,15, and 50 iterations Gaussian cube files are to be produced.

```
set pspw_dplot:iteration_list 3,10:15,50
```

12.1.4.1 Adding Geometry Constraints to a Car-Parrinello Simulation

The Car-Parrinello module allows users to freeze the cartesian coordinates in a simulation (Note - the Car-Parrinello code recognizes Cartesian constraints, but it does not recognize internal coordinate constraints). The +SET+ directive (Section Applying constraints in geometry optimizations) is used to freeze atoms, by specifying a directive of the form:

```
set geometry:actlist <integer list_of_center_numbers>
```

This defines only the centers in the list as active. All other centers will have zero force assigned to them, and will remain frozen at their starting coordinates during a Car-Parrinello simulation.

For example, the following directive specifies that atoms numbered 1, 5, 6, 7, 8, and 15 are active and all other atoms are frozen:

```
set geometry:actlist 1 5:8 15
```

or equivalently,

```
set geometry:actlist 1 5 6 7 8 15
```

If this option is not specified by entering a +SET+ directive, the default behavior in the code is to treat all atoms as active. To revert to this default behavior after the option to define frozen atoms has been invoked, the +UNSET+ directive must be used (since the database is persistent, see Section NWChem Architecture). The form of the +UNSET+ directive is as follows:

```
unset geometry:actlist
```

In addition, the Car-Parrinello module allows users to freeze bond lengths via a Shake algorithm. The following +SET+ directive shows how to do this.

```
set nwpw:shake_constraint "2 6 L 6.9334"
```

This input fixes the bond length between atoms 2 and 6 to be 6.9334 bohrs. Note that this input only recognizes bohrs.

When using constraints it is usually necessary to turn off center of mass shifting. This can be done by the following +SET+ directive.

```
set nwpw:com_shift .false.
```

12.1.4.2 Car-Parrinello Output Datafiles

12.1.4.2.1 XYZ motion file

Data file that stores ion positions and velocities as a function of time in XYZ format.

```
[line 1: ] n_ion
[line 2: ] do ii=1,n_ion
[line 2+ii: ] atom_name(ii), x(ii),y(ii),z(ii),vx(ii),vy(ii),vz(ii)
end do
[line n_ion+3 ] n_nion
 do ii=1,n_ion
[line n_ion+3+ii: ] atom_name(ii), x(ii),y(ii),z(ii), vx(ii),vy(ii),vz(ii)
end do
[line 2*n_ion+4: ] ....
```

12.1.4.2.2 ION_MOTION motion file

Datafile that stores ion positions and velocities as a function of time

```
[line 1: ] it_out, n_ion, omega, a1.x,a1.y,a1.z, a2.x,a2,y,a2.z, a3.x,a3.y,a3.z
[line 2: ] time
do ii=1,n_ion
[line 2+ii: ] ii, atom_symbol(ii),atom_name(ii), x(ii),y(ii),z(ii), vx(ii),vy(ii),vz(ii)
end do
[line n_ion+3 ] time
do do ii=1,n_ion
[line n_ion+3+ii: ] ii, atom_symbol(ii),atom_name(ii), x(ii),y(ii),z(ii), vx(ii),vy(ii),vz(ii)
end do
[line 2*n_ion+4: ] ....
```

12.1.4.2.3 EMOTION motion file

Datafile that store energies as a function of time.

```
[line 1: ] time, E1,E2,E3,E4,E5,E6,E7,E8,(E9,E10, if Nose-Hoover),eave,evar,have,hvar,ion_Temp
[line 2: ] ...
```

where

```
E1 = total energy
E2 = potential energy
E3 = ficticious kinetic energy
E4 = ionic kinetic energy
E5 = orbital energy
E6 = hartree energy
E7 = exchange-correlation energy
E8 = ionic energy
eave = average potential energy
evar = variance of potential energy
have = average total energy
hvar = variance of total energy
ion_Temp = temperature
```

12.1.4.2.4 HMOTION motion file

Datafile that stores the rotation matrix as a function of time.

```
[line 1: ] time
[line 2: ] ms,ne(ms),ne(ms)
do i=1,ne(ms)
[line 2+i: ] (hml(i,j), j=1,ne(ms)
end do
[line 3+ne(ms): ] time
[line 4+ne(ms): ] ....
```

12.1.4.2.5 EIGMOTION motion file

Datafile that stores the eigenvalues for the one-electron orbitals as a function of time.

```
[line 1: ] time, (eig(i), i=1,number_orbitals)
[line 2: ] ...
```

12.1.4.2.6 OMOTION motion file

Datafile that stores a reduced representation of the one-electron orbitals. To be used with a molecular orbital viewer that will be ported to NWChem in the near future.

12.1.5 Born-Oppenheimer Molecular Dynamics

```
NWPW
...
  BO_STEPS <integer bo_inner_iteration bo_outer_iteration default 10 100>
  BO_TIME_STEP  <real bo_time_step default 5.0>
  BO_ALGORITHM [verlet|| velocity-verlet || leap-frog]
  BO_FAKE_MASS <real bo_fake_mass default 500.0>
END
```

12.1.6 i-PI Socket Communication

```
NWPW
  SOCKET (UNIX || IPI_CLIENT) <string socketname default (see input description)>
END
```

The NWPW module provides native communication via the i-PI socket protocol. The behavior is identical to the i-PI socket communication provided by the DRIVER module. The NWPW implementation of the SOCKET directive is better optimized for plane-wave calculations.

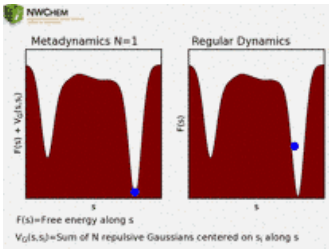
For proper behavior, the TASK directive should be set to GRADIENT , e.g. TASK PSPW GRADIENT or TASK BAND GRADIENT .

12.1.7 Metropolis Monte-Carlo

```
NWPW
...
  MC_STEPS <integer mc_inner_iteration mc_outer_iteration default 10 100>
END
```

12.1.8 Free Energy Simulations

12.1.8.1 MetaDynamics



Metadynamics234 is a powerful, non-equilibrium molecular dynamics method which accelerates the sampling of the multidimensional free energy surfaces of chemical reactions. This is achieved by adding an external time-dependent bias potential that is a function of user defined collective variables, $\mathbf{s}$. The bias potential discourages the system from sampling previously visited values of $\mathbf{s}$ (i.e., encourages the system to explore new values of $\mathbf{s}$). During the simulation the bias potential accumulates in low energy wells which then allows the system to cross energy barriers much more quickly than would occur in standard dynamics. The collective variable $\mathbf{s}$ is a generic function of the system coordinates,$\mathbf{R}$ (e.g. bond distance, bond angle, coordination numbers, etc.) that is capable of describing the chemical reaction of interest. $\mathbf{s}\left(\mathbf{R}\right)$ can be regarded as a reaction coordinate if it can distinguish between the reactant, transition, and products states, and also capture the kinetics of the reaction.

The biasing is achieved by “flooding” the energy landscape with repulsive Gaussian “hills” centered on the current location of $\mathbf{s}\left(\mathbf{R}\right)$ at a constant time intervalΔt. If the height of the Gaussians is constant in time

then we have standard metadynamics; if the heights vary (slowly decreased) over time then we have well-tempered metadynamics. In between the addition of Gaussians, the system is propagated by normal (but out of equilibrium) dynamics. Suppose that the dimension of the collective space is d, i.e. $\mathbf{s}\left(\mathbf{R}\right)=\bigl[s_1\left(\mathbf{R}\right),s_2\left(\mathbf{R}\right),\ldots,s_d\left(\mathbf{R}\right)\bigr]$ and that prior to any time t during the simulation, $N+1$ Gaussians centered on $\mathbf{S}^{t_g}$ are deposited along the trajectory of $\mathbf{s}\left(\mathbf{R}\right)$ at times $t_g = 0, \Delta t, 2\Delta t, \ldots ,N\Delta t$. Then, the value of the bias potential, V, at an arbitrary point, $\mathbf{s}\left(\mathbf{R}\right)=\bigl[s_1\left(\mathbf{R}\right),s_2\left(\mathbf{R}\right),\ldots,s_d\left(\mathbf{R}\right)\bigr]$, along the trajectory of $\mathbf{s}\left(\mathbf{R}\right)$ at time t is given by

$$V_{meta}\left(\pmb{s},t\right) = \sum_{t_g=0}^{t_g<} W(t) \exp\left(-\sum_{i=1}^{d}\frac{({s_i}-{s_i}^{t_g})^2}{2\sigma_i^2}\right)$$

where $W(t)=W_0 \exp\left(-\frac{V_{meta}\left(\mathbf{s},t-\Delta t\right)}{k_B T_{tempered}}\right)$ is the time-dependent Gaussian height. $\sigma_i \,(i=1,2,\ldots,d)$ and W_0 are width and initial height respectively of Gaussians, and $T_{tempered}$ is the tempered metadynamics temperature. $T_{tempered}=0$ corresponds to standard molecular dynamics because $W(t)=0$ and therfore there is no bias. $T_{tempered}=\infty$ corresponds to standard metadynamics since in this case $W(t)=W_0$=constant. A positive, finite value of $T_{tempered}$ (eg. $T_{tempered}$ >=1500 K) corresponds to *well-tempered* metadynamics in which $0 < W(t)<= W_0$.

For sufficiently large t, the history potential $V_{meta}\left(\mathbf{s},t\right)$ will nearly flatten the free energy surface, $F(\mathbf{s})$, along $\mathbf{S}$ and an unbiased estimator of F(s) is given by

$$F(\mathbf{s}) = -\left(1+\frac{T}{T_{tempered}}\right)\lim_{t \to \infty} V_{meta}(\mathbf{s},t)$$

12.1.8.1.1 Input

Input to a metadynamics simulation is contained within the METADYNAMICS sub-block. Listed below is the the format of a METADYNAMICS sub-block,

```
NWPW
 METADYNAMICS
   [
   BOND <integer atom1_index no default> <integer atom2_index no default>
       [W <real w default 0.00005>]
       [SIGMA <real sigma default 0.1>]
       [RANGE <real a b default (see input description)>]
       [NRANGE <integer nrange default 501>]
   ...]
   [
   ANGLE <integer atom1_index no default> <integer atom2_index no default> <integer atom3_index no default>
       [W <real w default 0.00005>]
       [SIGMA <real sigma default 0.1>]
       [RANGE <real a b default 0]
       [NRANGE <integer nrange default 501>]
   ...]
   [
   COORD_NUMBER [INDEX1 <integer_list atom1_indexes no default>][INDEX2 <integer_list atom2_indexes no default>]
       [SPRIK]
       [N <real n default 6.0>]
       [M <real m default 12.0>]
       [R0 <real r0 default 3.0>]

       [W <real w default 0.00005>]
       [SIGMA <real sigma default 0.1>]
       [RANGE <real a b no default>]
       [NRANGE <integer nrange default 501>]
   ...]
   [
   N-PLANE  <integer atom1_index no default> <integer_list atom_indexes no default>
       [W <real w default 0.00005>]
       [SIGMA <real sigma default 0.1>]
       [RANGE <real a b no default>]
       [NRANGE <integer nrange default 501>]
       [NVECTOR <real(3) nx ny nz>]
   ...]
   [UPDATE <integer meta_update default 1>]
   [PRINT_SHIFT <integer print_shift default 0>]
   [TEMPERED <real tempered_temperature no default>]
   [BOUNDARY <real w_boundary sigma_boundary no default>]
  END
 END
```

Multiple collective variables can be defined at the same time, e.g.

```
NWPW
 METADYNAMICS
   BOND 1 8  W 0.0005 SIGMA 0.1
   BOND 1 15 W 0.0005 SIGMA 0.1
  END
END
```

will produce a two-dimensional potential energy surface.

12.1.8.2 TAMD - Temperature Accelerated Molecular Dynamics

12.1.8.2.1 Input

12.1.8.3 Collective Variables

12.1.8.3.1 Bond Distance Collective Variable

This describes the bond distance between any pair of atoms$i$ and j:

$$s\left(r_{ij}\right) = \left \vert \mathbf{r}_{i}-\mathbf{r}_j\right\vert = r_{ij}$$

12.1.8.3.2 Angle Collective Variable

This describes the bond angle formed at$i$ by the triplet $<ijk>$

$$s\left(r_{ij},r_{ik}\right) = \frac{\pmb{r}_{ij}\cdot\pmb{r}_{ik}}{r_{ij}r_{ik}}$$

12.1.8.3.3 Coordination Collective Variable

The coordination number collective variable is defined as

$$s\left(r_{ij},r_{0}\right) =\sum_{i,j}\xi_{ij}$$

where the summation over i and j runs over two types of atoms,ξ_{ij} is the *weighting function*, and r_{0} is the cut-off distance. In the standard procedure for computing the coordination number, ξ_{ij} =1 if $r_{ij} < r_0$, otherwise ξ_{ij} =0, implying that ξ_{ij} is not continuous when $r_{ij}=r_{0}$. To ensure a smooth and continuous definition of the coordination number, we adopt two variants of the weighting function. The first variant is

$$\xi_{ij} = \frac{1-\left(r_{ij}/r_{0}\right)^n}{1-\left(r_{ij}/r_{0}\right)^m}$$

where n and m are integers (m > n) chosen such that$\xi_{ij}\approx 1$ when $r_{ij} < r_0$ and $\xi_{ij}\rightarrow 0$ when r_{ij} is much larger than r_{0}. For example, the parameters of the O-H coordination in water is well described by r_{0} =1.6 Å, $n=6$ and $m=18$. In practice, n and m must tuned for a given r_{0} to ensure that ξ_{ij} is smooth and satisfies the above mentioned properties, particularly the large r_{ij}

The second form of the weighting function, which is due to Sprik, is

$$\xi_{ij}=\frac{1}{1 + \exp\left[n\left(r_{ij}-r_{0}\right)\right]}$$

In this definition ξ_{ij} is analogous to the Fermi function and its width is controlled by the parameter$\frac{1}{n}$. Large and small values of n respectively correspond to sharp and soft transitions at$r_{ij} = r_{0}$. Furthermore ξ_{ij} should approach 1 and 0 when $r_{ij} < 0$ and respectively. In practice n =6-10 Å $^{-1}$. For example, a good set of parameters of the O-H coordination in water is r_{0} =1.4 Å and n =10 Å $^{-1}$.

12.1.8.3.4 N-Plane Collective Variable

The N-Plane collective variable is useful for probing the adsorption of adatom/admolecules to a surface. It is defined as the *average distance* of the adatom/admolecule from a given layer in the slab along the surface normal:

$$s = Z_{ads}-\frac{1}{N_{plane}}\sum_{i=1}^{N_{plane}}Z_i$$

where Z_{ads} denotes the position of the adatom/admolecule/impurity along the surface normal (here, we assume the surface normal to be the z-axis) and the summation over i runs over N_{plane} atoms at Z_i which form the layer. The layer could be on the face or in the interior of the slab.

12.1.8.3.5 User defined Collective Variable

12.1.9 Extended X-Ray Absorption Fine Structure (EXAFS) - Integration with FEFF6L

12.1.10 Frozen Phonon Calculations

12.1.11 Steepest Descent

The functionality of this task is now performed automatically by the PSPW and BAND. For backward compatibility, we provide a description of the input to this task.

The steepest_descent task is used to optimize the one-electron orbitals with respect to the total energy. In addition it can also be used to optimize geometries. This method is meant to be used for coarse optimization of the one-electron orbitals.

Input to the steepest_descent simulation is contained within the steepest_descent sub-block.

```
NWPW
...
 STEEPEST_DESCENT
   ...
 END
...
END
```

To run a steepest_descent calculation the following directive is used:

```
TASK PSPW steepest_descent
TASK BAND steepest_descent
```

The steepest_descent sub-block contains a great deal of input, including pointers to data, as well as parameter input. Listed below is the format of a STEEPEST_DESCENT sub-block.

```
NWPW
...
 STEEPEST_DESCENT
   CELL_NAME <string cell_name>
   [GEOMETRY_OPTIMIZE]
   INPUT_WAVEFUNCTION_FILENAME <string input_wavefunctions default file_prefix.movecs>
   OUTPUT_WAVEFUNCTION_FILENAME <string output_wavefunctions default file_prefix.movecs>
   FAKE_MASS <real fake_mass default 400000.0>
   TIME_STEP <real time_step default 5.8>
   LOOP <integer inner_iteration outer_iteration default 10 1>
   TOLERANCES <real tole tolc tolr default 1.0d-9 1.0d-9 1.0d-4>
   ENERGY_CUTOFF <real ecut default (see input desciption)>
   WAVEFUNCTION_CUTOFF <real wcut default (see input description)>
   EWALD_NCUT <integer ncut default 1>
   EWALD_RCUT <real rcut default (see input description)>
   XC (Vosko      || LDA          || PBE96        || revPBE       || HF           ||
      LDA-SIC    || LDA-SIC/2    || LDA-0.4SIC    || LDA-SIC/4    || LDA-0.2SIC    ||
      PBE96-SIC  || PBE96-SIC/2  || PBE96-0.4SIC  || PBE96-SIC/4  || PBE96-0.2SIC  ||
      revPBE-SIC || revPBE-SIC/2 || revPBE-0.4SIC || revPBE-SIC/4 || revPBE-0.2SIC ||
      PBE0       || revPBE0      || default Vosko)
   [MULLIKEN]
 END
...
END
```

The following list describes the input for the STEEPEST_DESCENT sub-block.

- cell_name - name of the simulation_cell named cell_name . See Simulation Cell.
- GEOMETRY_OPTIMIZE - optional keyword which if specified turns on geometry optimization.
- input_wavefunctions - name of the file containing one-electron orbitals
- output_wavefunctions - name of the file tha will contain the one-electron orbitals at the end of the run.
- fake_mass - value for the electronic fake mass\(\mu\)
- time_step - value for the time step\(Delta t\).
- inner_iteration - number of iterations between the printing out of energies and tolerances
- outer_iteration - number of outer iterations
- tole - value for the energy tolerance.
- tolc - value for the one-electron orbital tolerance.
- tolr - value for the ion position tolerance.
- ecut - value for the cutoff energy used to define the density. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- wcut - value for the cutoff energy used to define the one-electron orbitals. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- ncut - value for the number of unit cells to sum over (in each direction) for the real space part of the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.
- rcut - value for the cutoff radius used in the Ewald summation. Note Ewald summation is only used if the simulation_cell is periodic.

Default set to be \(\frac{MIN(\left| \vec{a_i} \right|)}{\pi}, i=1,2,3\).

- (Vosko || PBE96 || revPBE || …) - Choose between Vosko et al's LDA parameterization or the orginal and revised Perdew, Burke, and Ernzerhof GGA functional. In addition, several hybrid options (hybrid options are not available in BAND).
- MULLIKEN - optional keyword which if specified causes a Mulliken analysis to be performed at the end of the simulation.

12.1.12 Simulation Cell

The simulation cell parameters are entered by defining a simulation_cell sub-block within the PSPW block. Listed below is the format of a simulation_cell sub-block.

```
NWPW
...
 SIMULATION_CELL  [units <string units default bohrs>]
   CELL_NAME <string name default 'cell_default'>
   BOUNDARY_CONDITIONS (periodic || aperiodic default periodic)
   LATTICE_VECTORS
     <real a1.x a1.y a1.z default 20.0 0.0 0.0>
     <real a2.x a2.y a2.z default 0.0 20.0 0.0>
     <real a3.x a3.y a3.z default 0.0 0.0 20.0>
   NGRID <integer na1 na2 na3 default 32 32 32>
  END
...
END
```

Basically, the user needs to enter the dimensions, gridding and boundary conditions of the simulation cell. The following list describes the input in detail.

- name - user-supplied name for the simulation block.

- periodic - keyword specifying that the simulation cell has periodic boundary conditions.
- aperiodic - keyword specifying that the simulation cell has free-space boundary conditions.
- a1.x a1.y a1.z - user-supplied values for the first lattice vector
- a2.x a2.y a2.z - user-supplied values for the second lattice vector
- a3.x a3.y a3.z - user-supplied values for the third lattice vector
- na1 na2 na3 - user-supplied values for discretization along lattice vector directions.

Alternatively, instead of explicitly entering lattice vectors, users can enter the unit cell using the standard cell parameters, a, b, c, \(alpha\), \(\beta\), and \(\gamma\), by using the LATTICE block. The format for input is as follows:

```
NWPW
...
 SIMULATION_CELL [units <string units default bohrs>]
   ...
   LATTICE
     [lat_a <real a default 20.0>]
     [lat_b <real b default 20.0>]
     [lat_c <real c default 20.0>]
     [alpha <real alpha default 90.0>]
     [beta <real beta default 90.0>]
     [gamma <real gamma default 90.0>]
   END
   ...
 END
...
END
```

The user can also enter the lattice vectors of standard unit cells using the keywords SC, FCC, BCC, for simple cubic, face-centered cubic, and body-centered cubic respectively. Listed below is an example of the format of this type of input.

```
NWPW
...
 SIMULATION_CELL [units <string units default bohrs>]
   SC 20.0
   ....
 END
...
END
```

Finally, the lattice vectors from the unit cell can also be defined using the fractional coordinate input in the GEOMETRY input (see section Geometry Lattice Parameters). Listed below is an example of the format of this type of input for an 8 atom silicon carbide unit cell.

```
geometry units au
 system crystal
   lat_a 8.277
   lat_b 8.277
   lat_c 8.277
   alpha 90.0
   beta 90.0
   gamma 90.0
 end
 Si -0.50000 -0.50000 -0.50000
 Si  0.00000  0.00000 -0.50000
 Si  0.00000 -0.50000  0.00000
 Si -0.50000  0.00000  0.00000
 C  -0.25000 -0.25000 -0.25000
 C   0.25000  0.25000 -0.25000
 C   0.25000 -0.25000  0.25000
 C  -0.25000  0.25000  0.25000
end
```

Warning - Currently only the “system crystal” option is recognized by NWPW. The “system slab” and “system polymer” options will be supported in the future.

12.1.13 Unit Cell Optimization

The PSPW module using the DRIVER geometry optimizer can optimize a crystal unit cell. Currently this type of

optimization works only if the geometry is specified in fractional coordinates. The following SET directive is used to tell the DRIVER geometry optimizer to optimize the crystal unit cell in addition to the geometry.

```
set includestress .true.
```

12.1.14 SMEAR - Fractional Occupation of the Molecular Orbitals

The smear keyword to turn on fractional occupation of the molecular orbitals in PSPW and BAND calculations

```
SMEAR <sigma default 0.001> [TEMPERATURE <temperature>]
                [FERMI || GAUSSIAN || MARZARI-VANDERBILT default FERMI]
                [ORBITALS <integer orbitals default 4>]
```

Fermi-Dirac (FERMI), Gaussian, and Marzari-Vanderbilt broadening functions are available. The ORBITALS keyword is used to change the number of virtual orbitals to be used in the calculation. Note to use this option the user must currently use the SCF minimizer. The following SCF options are recommended for running fractional occupation

```
SCF Anderson outer_iterations 0 Kerker 2.0
```

12.1.15 Spin Penalty Functions

Spin-penalty functions makes it easier to define antiferromagnetic structures. These functions are implemented by adding a scaling factor to the non-local psp for up/down spins on atoms and angular momentum that you specify.

Basically, the pseudopotential energy

$$E_{psp}= \sum_{\sigma=\uparrow,\downarrow} \sum_{i=1}^{n_{elc}^\sigma} \sum_{I=1}^{n_{ions}} \left(<\psi_i^\sigma|V_{local}^{I}|\psi_i^\sigma> + \sum_{l=0}^{l_{max}^I} \sum_{m=-l}^{l} \sum_{n=1}^{n_{max}^I} \sum_{n'=1}^{n_{max}^I} <\psi_i^\sigma|P_{nlm}^I> h_{l,n,n'}^I <P_{n'lm}^{I}|\psi_i^\sigma> \right)$$

was modified to

$$E_{psp}= \sum_{\sigma=\uparrow,\downarrow} \sum_{i=1}^{n_{elc}^\sigma} \sum_{I=1}^{n_{ions}} \left(<\psi_i^\sigma|V_{local}^{I}|\psi_i^\sigma> + \sum_{l=0}^{l_{max}^I} \sum_{m=-l}^{l} \sum_{n=1}^{n_{max}^I} \sum_{n'=1}^{n_{max}^I} \left(1-\delta_{l,l^\sigma} \delta_{I,ionlist^\sigma}(\xi^\sigma-1)\right) <\psi_i^\sigma|P_{nlm}^I> h_{l,n,n'}^I <P_{n'lm}^{I}|\psi_i^\sigma> \right)$$

An example input is as follows:

```
title "hematite 10 atoms"

start hema10

memory 1900 mb

permanent_dir ./perm
scratch_dir   ./perm

geometry center noautosym noautoz print
 system crystal
   lat_a 5.42
   lat_b 5.42
   lat_c 5.42
   alpha 55.36
   beta  55.36
   gamma 55.36
 end
Fe                0.355000   0.355000   0.355000
Fe                0.145000   0.145000   0.145000
Fe               -0.355000  -0.355000  -0.355000
Fe                0.855000   0.855000   0.855000
O                 0.550000  -0.050000   0.250000
O                 0.250000   0.550000  -0.050000
O                -0.050000   0.250000   0.550000
O                -0.550000   0.050000  -0.250000
O                -0.250000  -0.550000   0.050000
O                 0.050000  -0.250000  -0.550000\
end

nwpw
  virtual 8
  odft
  ewald_rcut 3.0
  ewald_ncut 8
  xc pbe96
  lmbfgs
  mult 1
  dplot
     density diff diff1.cube
  end

  #spin penalty functions
  pspspin up   d -1.0 1:2
  pspspin down d -1.0 3:4
end
task pspw energy
task pspw pspw_dplot

nwpw
  pspspin off
  dplot
     density diff diff2.cube
  end
end
task pspw energy
task pspw pspw_dplot
```

12.1.16 AIMD/MM (QM/MM)

A QM/MM capability is available that is integrated with PSPW module and can be used with Car-Parrinello simulations. Currently, the input is not very robust but it is straightforward. The first step to run a QM/MM simulations is to define the MM atoms in the geometry block. The MM atoms must be at the end of the geometry and a carat, ” ^ “, must be appended to the end of the atom name, e.g.

```
geometry units angstrom nocenter noautosym noautoz print xyz
 C -0.000283 0.000106 0.000047
 Cl -0.868403 1.549888 0.254229
 Cl 0.834043 -0.474413 1.517103
 Cl -1.175480 -1.275747 -0.460606
 Cl 1.209940 0.200235 -1.310743
 O^ 0.3226E+01 -0.4419E+01 -0.5952E+01
 H^ 0.3193E+01 -0.4836E+01 -0.5043E+01
 H^ 0.4167E+01 -0.4428E+01 -0.6289E+01
 O^ 0.5318E+01 -0.3334E+01 -0.1220E+01
 H^ 0.4978E+01 -0.3040E+01 -0.2113E+01
 H^ 0.5654E+01 -0.2540E+01 -0.7127E+00
end
```

Another way to specify the MM atoms is to use the mm_tags option which appends the atoms with a " ^ ".

```
geometry units angstrom nocenter noautosym noautoz print xyz
 C -0.000283 0.000106 0.000047
 Cl -0.868403 1.549888 0.254229
 Cl 0.834043 -0.474413 1.517103
 Cl -1.175480 -1.275747 -0.460606
 Cl 1.209940 0.200235 -1.310743
 O 0.3226E+01 -0.4419E+01 -0.5952E+01
 H 0.3193E+01 -0.4836E+01 -0.5043E+01
 H 0.4167E+01 -0.4428E+01 -0.6289E+01
 O 0.5318E+01 -0.3334E+01 -0.1220E+01
 H 0.4978E+01 -0.3040E+01 -0.2113E+01
 H 0.5654E+01 -0.2540E+01 -0.7127E+00
end
NWPW
 QMMM
  mm_tags 6:11
 END
END
```

The option "mm_tags off" can be used to remove the " ^ " from the atoms, i.e.

```
NWPW
 QMMM
  mm_tags 6:11 off
 END
END
```

Next the pseudopotentials have be defined for the every type of MM atom contained in the geometry blocks. The following local pseudopotential suggested by Laio, VandeVondele and Rothlisberger can be automatically generated.

$$\begin{matrix}V(\vec{r}) = -Z_{ion}\frac{{r_c}^{n_{\sigma}} - r^{n_{\sigma}}}{-sign(Z_{ion})*{r_c}^{n_{\sigma}+1}-r^{n_{\sigma}+1}}\end{matrix}$$

The following input To define this pseudopo the O^ MM atom using the following input

```
NWPW
 QMMM
  mm_psp O^ -0.8476 4 0.70
 END
END
```

defines the local pseudopotential for the O^ MM atom , where $Z_{ion}=-0.8476$, $n_{\sigma}=4$, and $r_c=0.7$. The following input can be used to define the local pseudopotentials for all the MM atoms in the geometry block defined above

```
NWPW
 QMMM
  mm_psp O^ -0.8476 4 0.70
  mm_psp H^ 0.4238 4 0.40
 END
END
```

Next the Lenard-Jones potentials for the QM and MM atoms need to be defined. This is done as as follows

```
NWPW
 QMMM
  lj_ion_parameters C 3.41000000d0 0.10d0
  lj_ion_parameters Cl 3.45000000d0 0.16d0
  lj_ion_parameters O^ 3.16555789d0 0.15539425d0
 END
END
```

Note that the Lenard-Jones potential is not defined for the MM H atoms in this example. The final step is to define the MM fragments in the simulation. MM fragments are a set of atoms in which bonds and angle harmonic potentials are defined, or alternatively shake constraints are defined. The following input defines the fragments for the two water molecules in the above geometry,

```
NWPW
 QMMM
  fragment spc
    size 3                #size of fragment
    index_start 6:9:3       #atom index list that defines the start of
                          # the fragments (start:final:stride)
    bond_spring 1 2 0.467307856 1.889726878   #bond i j Kspring r0
    bond_spring 1 3 0.467307856 1.889726878   #bond i j Kspring r0
    angle_spring 2 1 3 0.07293966 1.910611932 #angle i j k Kspring theta0
  end
 END
END
```

The fragments can be defined using shake constraints as

```
NWPW
 QMMM
  fragment spc
    size 3                #size of fragment
    index_start 6:9:3       #atom index list that defines the start of
                          # the fragments (start:final:stride)
    shake units angstroms 1 2 3 cyclic 1.0 1.632993125 1.0
  end
 END
END
```

Alternatively, each water could be defined independently as follows.

```
NWPW
 QMMM
  fragment spc1
    size 3                #size of fragment
    index_start 6           #atom index list that defines the start of
                          #the fragments
    bond_spring 1 2 0.467307856 1.889726878 #bond i j Kspring r0
    bond_spring 1 3 0.467307856 1.889726878 #bond i j Kspring r0
    angle_spring 2 1 3 0.07293966 1.910611932 #angle i j k Kspring theta0
  end
  fragment spc2
    size 3                #size of fragment
    index_start 9           #atom index list that defines the start of
                          #the fragments
    shake units angstroms 1 2 3 cyclic 1.0 1.632993125 1.0
  end
 END
END
```

12.1.17 PSP_GENERATOR

A one-dimensional pseudopotential code has been integrated into NWChem. This code allows the user to modify and develop pseudopotentials. Currently, only the Hamann and Troullier-Martins norm-conserving pseudopotentials can be generated. In future releases, the pseudopotential library (section Pseudopotential and PAW basis Libraries) will be more complete, so that the user will not have explicitly generate pseudopotentials using this module.

Input to the PSP_GENERATOR task is contained within the PSP_GENERATOR sub-block.

```
NWPW
...
 PSP_GENERATOR
  ...
 END
...
END
```

To run a PSP_GENERATOR calculation the following directive is used:

```
TASK PSPW PSP_GENERATOR
```

Listed below is the format of a PSP_GENERATOR sub-block.

```
NWPW
...
 PSP_GENERATOR
  PSEUDOPOTENTIAL_FILENAME: <string psp_name>
  ELEMENT: <string element>
  CHARGE: <real charge>
  MASS_NUMBER: <real mass_number>
  ATOMIC_FILLING: <integer ncore nvalence> ( (1||2||...) (s||p||d||f||...) <real filling>  ...)

  [CUTOFF: <integer lmax> ( (s||p||d||f||g) <real rcut> ...) ]

  PSEUDOPOTENTIAL_TYPE: (TROULLIER-MARTINS || HAMANN default HAMANN)
  SOLVER_TYPE: (PAULI || SCHRODINGER default PAULI)
  EXCHANGE_TYPE: (dirac || PBE96 default DIRAC)
  CORRELATION_TYPE: (VOSKO || PBE96 default VOSKO)
  [SEMICORE_RADIUS: <real rcore>]

 END
...
END
```

The following list describes the input for the PSP_GENERATOR sub-block.

- psp_name - name that points to a.
- element - Atomic symbol.
- charge - charge of the atom
- mass - mass number for the atom
- ncore - number of core states
- nvalence - number of valence states.
- ATOMIC_FILLING:.....(see below)
- filling - occupation of atomic state
- CUTOFF:....(see below)
- rcore - value for the semicore radius (see below)

12.1.17.1 ATOMIC_FILLING Block

This required block is used to define the reference atom which is used to define the pseudopotential. After the ATOMIC_FILLING: line, the core states are listed (one per line), and then the valence states are listed (one per line). Each state contains two integer and a value. The first integer specifies the radial quantum number, \(n\), the second integer specifies the angular momentum quantum number, \(l\), and the third value specifies the occupation of the state.

For example to define a pseudopotential for the Neon atom in the\(1s^2 2s^2 2p^6\) state could have the block

```
ATOMIC_FILLING: 1 2
 1 s 2.0 #core state - 1s^2
 2 s 2.0 #valence state - 2s^2
 2 p 6.0 #valence state - 2p^6
```

for a pseudopotential with a \(2s\) and \(2p\) valence electrons or the block

```
ATOMIC_FILLING: 3 0
 1 s 2.0 #core state
 2 s 2.0 #core state
 2 p 6.0 #core state
```

could be used for a pseudopotential with no valence electrons.

12.1.17.2 CUTOFF

This optional block specifies the cutoff distances used to match the all-electron atom to the pseudopotential atom. For Hamann pseudopotentials \(r_{cut}(l)\) defines the distance where the all-electron potential is matched to the

pseudopotential, and for Troullier-Martins pseudopotentials \(r_{cut}(l)\) defines the distance where the all-electron orbital is matched to the pseudowavefunctions. Thus the definition of the radii depends on the type of pseudopotential. The cutoff radii used in Hamann pseudopotentials will be smaller than the cutoff radii used in Troullier-Martins pseudopotentials.

For example to define a softened Hamann pseudopotential for Carbon would be

```
ATOMIC_FILLING: 1 2
 1 s 2.0
 2 s 2.0
 2 p 2.0
CUTOFF: 2
 s 0.8
 p 0.85
 d 0.85
```

while a similarly softened Troullier-Marting pseudopotential for Carbon would be

```
ATOMIC_FILLING: 1 2
 1 s 2.0
 2 s 2.0
 2 p 2.0
CUTOFF: 2
 s 1.200
 p 1.275
 d 1.275
```

12.1.17.3 SEMICORE_RADIUS

Specifying the SEMICORE_RADIUS option turns on the semicore correction approximation proposed by Louie et al (S.G. Louie, S. Froyen, and M.L. Cohen, Phys. Rev. B, **26**(, 1738, (1982)). This approximation is known to dramatically improve results for systems containing alkali and transition metal atoms.

The implementation in the PSPW module defines the semi-core density,\(\rho_{semicore}\), by using the sixth-order polynomial

\[\rho_{semicore}(r) = \begin{cases} \rho_{core} \mbox{if } r \ge r_{semicore} \\ c_0 + c_3 r^3 + c_4 r^4 + c_5 r^5 + c_6 r^6 \mbox{if } r < r_{semicore} \end{cases}\]

This expansion was suggested by Fuchs and Scheffler (M. Fuchs, and M. Scheffler, Comp. Phys. Comm.,**119**,67 (1999)), and is better behaved for taking derivatives (i.e. calculating ionic forces) than the expansion suggested by Louie et al.

12.1.18 PAW Tasks: Legacy Implementation

(*This capability is now available in PSPW. It is recommended that this module only be used for testing purposes.*)

All input to the PAW Tasks is contained within the compound NWPW block,

```
NWPW
 ...
END
```

To perform an actual calculation a TASK PAW directive is used (Task).

```
TASK PAW
```

In addition to the directives listed inTask, i.e.

```
TASK paw energy
TASK paw gradient
TASK paw optimize
TASK paw saddle
TASK paw freqencies
TASK paw vib
```

there are additional directives that are specific to the PSPW module, which are:

```
TASK PAW [Car-Parrinello || steepest_descent ]
```

Once a user has specified a geometry, the PAW module can be invoked with no input directives (defaults invoked throughout). There are sub-directives which allow for customized application; those currently provided as options for the PAW module are:

```
NWPW
 CELL_NAME <string cell_name default 'cell_default'>
 [GEOMETRY_OPTIMIZE]
 INPUT_WAVEFUNCTION_FILENAME <string input_wavefunctions default input_movecs>
 OUTPUT_WAVEFUNCTION_FILENAME <string output_wavefunctions default input_movecs>
 FAKE_MASS <real fake_mass default 400000.0>
 TIME_STEP <real time_step default 5.8>
 LOOP <integer inner_iteration outer_iteration default 10 100>
 TOLERANCES <real tole tolc default 1.0e-7 1.0e-7>
 CUTOFF <real cutoff>
 ENERGY_CUTOFF <real ecut default (see input description)>
 WAVEFUNCTION_CUTOFF <real wcut default (see input description)>
 EWALD_NCUT <integer ncut default 1>]
 EWALD_RCUT <real rcut default (see input description)>
 XC (Vosko || PBE96 || revPBE default Vosko)
 DFT||ODFT||RESTRICTED||UNRESTRICTED
 MULT <integer mult default 1>
 INTEGRATE_MULT_L <integer imult default 1>
 SIMULATION_CELL
   ... (see input description)
 END
 CAR-PARRINELLO
   ... (see input description)
 END
 MAPPING <integer mapping default 1>
END
```

The following list describes these keywords.

- cell_name - name of the the simulation_cell named cell_name . The current version of PAW only accepts periodic unit cells. See Simulation Cell.
- GEOMETRY_OPTIMIZE - optional keyword which if specified turns on geometry optimization.
- input_wavefunctions - name of the file containing one-electron orbitals
- output_wavefunctions - name of the file that will contain the one-electron orbitals at the end of the run.
- fake_mass - value for the electronic fake mass\(\mu\). This parameter is not presently used in a conjugate gradient simulation
- time_step - value for the time step (\(\Delta t\)). This parameter is not presently used in a conjugate gradient simulation.
- inner_iteration - number of iterations between the printing out of energies and tolerances
- outer_iteration - number of outer iterations
- tole - value for the energy tolerance.
- tolc - value for the one-electron orbital tolerance.
- cutoff - value for the cutoff energy used to define the wavefunction. In addition using the CUTOFF keyword automatically sets the cutoff energy for the density to be twice the wavefunction cutoff.
- ecut - value for the cutoff energy used to define the density. Default is set to be the maximum value that will fit within the simulation_cell cell_name .
- wcut - value for the cutoff energy used to define the one-electron orbitals. Default is set to be the maximum value that will fix within the simulation_cell cell_name .
- ncuth - value for the number of unit cells to sum over (in each direction) for the real space part of the smooth compensation summation.
- rcut - value for the cutoff radius used in the smooth compensation summation.

Default set to be \(\frac{MIN(\left| \vec{a_i} \right|)}{\pi}, i=1,2,3\).

- (Vosko || PBE96 || revPBE) - Choose between Vosko et al's LDA parametrization or the original and revised Perdew, Burke, and Ernzerhof GGA functional.
- MULT - optional keyword which if specified allows the user to define the spin multiplicity of the system
- INTEGRATE_MULT_L - optional keyword which if specified allows the user to define the angular XC integration of the augmented region
- SIMULATION_CELL (see Simulation Cell)
- CAR-PARRINELLO (see Car-Parrinello)
- mapping - for a value of 1 slab FFT is used, for a value of 2 a 2d-Hilbert FFT is used.

12.1.19 Pseudopotential and PAW basis Libraries

A library of pseudopotentials used by PSPW and BAND is currently available in the directory

$NWCHEM_TOP/src/nwpw/libraryp/pspw_default

The elements listed in the following table are present:

```
 H                                  He
-------                          ----------
 Li Be                            B  C  N  O  F  Ne
-------                          ----------
 Na Mg                             Al Si P  S  Cl Ar
---------------------------------------------------------
 K  Ca Sc Ti V  Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
---------------------------------------------------------
 Rb Sr Y  Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I  Xe
---------------------------------------------------------
 Cs Ba La Hf Ta W  Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
---------------------------------------------------------
 Fr Ra .
----------
      -----------------------------------------
      . . . . . Gd . . . . . .
      -----------------------------------------
      . . U . Pu . . . . . .
      -----------------------------------------
```

The pseudopotential libraries are continually being tested and added. Also, the PSPW program can read in pseudopotentials in CPI and TETER format generated with pseudopotential generation programs such as the OPIUM package of Rappe et al. The user can request additional pseudopotentials from Eric J. Bylaska at (Eric.Bylaska@pnl.gov).

Similarly, a library of PAW basis used by PAW is currently available in the directory $NWCHEM_TOP/src/nwpw/libraryp/paw_default

```
 H                                  He
-------                          ----------
 Li Be                            B  C  N  O  F  Ne
-------                          ----------
 Na Mg                             Al Si P  S  Cl Ar
---------------------------------------------------------
 K  Ca Sc Ti V  Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
---------------------------------------------------------
 . . . . . .
---------------------------------------------------------
 . . . . . .
---------------------------------------------------------
 . . .
----------
      -----------------------------------------
      . . . . . .
      -----------------------------------------
      . . . . . .
      -----------------------------------------
```

Currently there are not very many elements available for PAW. However, the user can request additional basis sets from Eric J. Bylaska at (Eric.Bylaska@pnl.gov).

A preliminary implementation of the HGH pseudopotentials (Hartwigsen, Goedecker, and Hutter) has been implemented into the PSPW module. To access the pseudopotentials the pseudopotentials input block is used. For example, to redirect

the code to use HGH pseudopotentials for carbon and hydrogen, the following input would be used.

```
nwpw
...
 pseudopotentials
  C library HGH_LDA
  H library HGH_LDA
 end
...
end
```

The implementation of HGH pseudopotentials is rather limited in this release. HGH pseudopotentials cannot be used to optimize unit cells, and they do not work with the MULLIKEN option. They also have not yet been implemented into the BAND structure code.
To read in pseudopotentials in CPI format the following input would be used.

```
nwpw
...
 pseudopotentials
  C CPI c.cpi
  H CPI h.cpi
 end
...
end
```

In order for the program to recognize the CPI format the CPI files, e.g. c.cpi have to be prepended with the""keyword.

To read in pseudopotentials in TETER format the following input would be used.

```
nwpw
...
 pseudopotentials
  C TETER c.teter
  H TETER h.teter
 end
...
end
```

In order for the program to recognize the TETER format the TETER files, e.g. c.teter have to be prepended with the" " keyword.

If you wish to redirect the code to a different directory other than the default one, you need to set the environmental variable NWCHEM_NWPW_LIBRARY to the new location of the libraryps directory.

12.1.20 NWPW RTDB Entries and Miscellaneous DataFiles

Input to the PSPW and Band modules are contained in both the RTDB and datafiles. The RTDB is used to store input that the user will need to directly specify. Input of this kind includes ion positions, ion velocities, and simulation cell parameters. The datafiles are used to store input, such the one-electron orbitals, one-electron orbital velocities, formatted pseudopotentials, and one-dimensional pseudopotentials, that the user will in most cases run a program to generate.

12.1.20.1 Ion Positions

The positions of the ions are stored in the default geometry structure in the RTDB and must be specified using the GEOMETRY directive.

12.1.20.2 Ion Velocities

The velocities of the ions are stored in the default geometry structure in the RTDB, and must be specified using the GEOMETRY directive.

12.1.20.3 Wavefunction Datafile

The one-electron orbitals are stored in a wavefunction datafile. This is a binary file and cannot be directly edited. This datafile is used by steepest_descent and Car-Parrinello tasks and can be generated using the wavefunction_initializer or wavefunction_expander tasks.

12.1.20.4 Velocity Wavefunction Datafile

The one-electron orbital velocities are stored in a velocity wavefunction datafile. This is a binary file and cannot be directly edited. This datafile is automatically generated the first time a Car-Parrinello task is run.

12.1.20.5 Formatted Pseudopotential Datafile

The pseudopotentials in Kleinman-Bylander form expanded on a simulation cell (3d grid) are stored in a formatted pseudopotential datafile (PSPW-“atomname.vpp”, BAND-“atomname.cpp”, PAW-“atomname.jpp”). These are binary files and cannot be directly edited. These datafiles are automatically generated.

12.1.20.6 One-Dimensional Pseudopotential Datafile

The one-dimensional pseudopotentials are stored in a one-dimensional pseudopotential file (“atomname.psp”). This is an ASCII file and can be directly edited with a text editor or can be generated using the pspw_generator task. However, these datafiles are usually atomatically generated.

The data stored in the one-dimensional pseudopotential file is

```
character*2 element :: element name
integer charge :: valence charge of ion
real mass :: mass of ion
integer lmax :: maximum angular component
real rcut(lmax) :: cutoff radii used to define pseudopotentials
integer nr :: number of points in the radial grid
real dr :: linear spacing of the radial grid
real r(nr):: one-dimensional radial grid
real Vpsp(nr,lmax) :: one-dimensional pseudopotentials
real psi(nr,lmax) :: one-dimensional pseudowavefunctions
real r_semicore :: semicore radius
real rho_semicore(nr) :: semicore density
```

and the format of it is:

```
[line 1: ] element [line 2: ] charge mass lmax
[line 3: ] (rcut(l), l=1,lmax)
[line 4: ] nr dr
[line 5: ] r(1) (Vpsp(1,l), l=1,lmax)
[line 6: ] ....
[line nr+4: ] r(nr) (Vpsp(nr,l), l=1,lmax)
[line nr+5: ] r(1) (psi(1,l), l=1,lmax) [line nr+6: ] ....
[line 2*nr+4:] r(nr) (psi(nr,l), l=1,lmax)
[line 2*nr+5:] r_semicore
if (r_semicore read) then
 [line 2*nr+6:] r(1) rho_semicore(1)
 [line 2*nr+7:] ....
 [line 3*nr+5:] r(nr) rho_semicore(nr)
end if
```

12.1.21 Car-Parrinello Scheme for Ab Initio Molecular Dynamics

Car and Parrinello developed a unified scheme for doing *ab initio* molecular dynamics by combining the motion of the ion cores and a fictitious motion for the Kohn-Sham orbitals of density-functional theory (R. Car and M. Parrinello, Phys. Rev. Lett. **55**, 2471, (1985) - simple introductioncpmd-lecture.pdf). At the heart of this method they introduced a fictitious kinetic energy functional for the Kohn-Sham orbitals.

$$\begin{align}KE \left (\bigl[\psi_{i,\sigma}(\vec{r}) \bigr] \right) = \sum_{i,\sigma}^{occ} \int d\vec{r}\ \mu \left | \dot{\psi}_{i,\sigma}(\vec{r}) \right | ^2\end{align}$$

Given this kinetic energy the constrained equations of motion are found by taking the first variation of the auxiliary Lagrangian.

$$ L = \sum_{i,\sigma}^{occ} \int d\vec{r}\ \mu \left | \dot{\psi}_{i,\sigma}(\vec{r}) \right | ^2; + \frac{1}{2} \sum_{I} M_I \left | \dot{\vec{R}}_{I} \right | ^2 - E \left [\bigl[\psi_{i,\sigma}(\vec{r}) \bigr] , \bigl[\vec{R}_I \bigr] \right] \\ + \sum_{ij,\sigma} \Lambda_{ij,\sigma} \left (\int d\vec{r}\ \psi_{i,\sigma}^{*}(\vec{r}) \psi_{j,\sigma}(\vec{r}) - \delta_{ij\sigma} \right)$$

Which generates a dynamics for the wavefunctions $\psi_{i,\sigma}(\vec{r})$ and atoms positions $\vec{R}_I$ through the constrained equations of motion:

$$\begin{matrix}\mu \ddot{\psi}_{i,\sigma}(\vec{r},t) = -\frac{\delta E}{\delta \psi_{i,\sigma }^{*} \left(\vec{r},t \right) } + \sum\limits_j \Lambda_{ij,\sigma} \psi_{j,\sigma} \left(\vec{r},t \right) \end{matrix}$$
$$\begin{matrix}M_I \ddot{\vec{R}}_I = -\frac{\partial E}{\partial \vec{R}_I} \end{matrix}$$

where μ is the fictitious mass for the electronic degrees of freedom andM_I are the ionic masses. The adjustable parameter μ is used to describe the relative rate at which the wavefunctions change with time.$\Lambda_{ij,\sigma}$ are the Lagrangian multipliers for the orthonormalization of the single-particle orbitals $\psi_{i,\sigma}(\vec{r})$. They are defined by the orthonormalization constraint conditions and can be rigorously found. However, the equations of motion for the Lagrange multipliers depend on the specific algorithm used to integrate the Eqns. above.

For this method to give ionic motions that are physically meaningful the kinetic energy of the Kohn-Sham orbitals must be relatively small when compared to the kinetic energy of the ions. There are two ways where this criterion can fail. First, the numerical integrations for the Car-Parrinello equations of motion can often lead to large relative values of the kinetic energy of the Kohn-Sham orbitals relative to the kinetic energy of the ions. This kind of failure is easily fixed by requiring a more accurate numerical integration, i.e. use a smaller time step for the numerical integration. Second, during the motion of the system a the ions can be in locations where there is an Kohn-Sham orbital level crossing, i.e. the density-functional energy can have two states that are nearly degenerate. This kind of failure often occurs in the study of chemical reactions. This kind of failure is not easily fixed and requires the use of a more sophisticated density-functional energy that accounts for low-lying excited electronic states.

12.1.21.1 Verlet Algorithm for Integration

Integrating the Eqns. above using the Verlet algorithm results in

$$\begin{matrix}\psi_{i,\sigma}^{t+ \Delta t} \leftarrow 2 \psi_{i,\sigma}^{t} - \psi_{i,\sigma}^{t-\Delta t} + \frac{(\Delta t)^2}{\mu} \left[\frac{\delta E}{\delta \psi_{i,\sigma}^{*}} + \sum_{j} \psi_{j,\sigma} \Lambda_{ji,\sigma} \right]_{t} \end{matrix}$$
$$\begin{matrix}\vec{R}_I^{t+\Delta t} \leftarrow 2 \vec{R}_I^{t} - \vec{R}_I^{t-\Delta t} + \frac{(\Delta t)^2}{M_I} \frac{\partial E} {\partial \vec{R}_I}\end{matrix}$$

In this molecular dynamic procedure we have to know variational derivative

$$\frac{\delta E}{\delta \psi_{i,\sigma}^{*}}$$

and the matrix $\Lambda_{ij,\sigma}$.
The variational derivative

$$\frac{\delta E}{\delta \psi_{i,\sigma}^{*}}$$

can be analytically found and is

$$\begin{aligned}\frac{\delta E}{\delta \psi_{i,\sigma}^{*}} = -\frac{1}{2} \nabla^2 \psi_{i,\sigma}(\vec{r}) \\ + \int d\vec{r^{\prime}} W_{ext}(\vec{r},\vec{r^{\prime}}) \psi_{i,\sigma}(\vec{r^{\prime}}) \\ + \int d\vec{r^{\prime}} \frac{n(\vec{r^{\prime}})}{|\vec{r}-\vec{r^{\prime}}|} \psi_{i,\sigma}(\vec{r})\\ + \mu_{xc}^{\sigma}(\vec{r}) \psi_{i,\sigma}(\vec{r}) \\ \equiv \hat{H} \psi_{i,\sigma} \end{aligned}$$

To find the matrix $\Lambda_{ij,\sigma}$ impose the orthonormality constraint on$\psi_{i,\sigma}^{t+\Delta t}$ obtain a matrix Riccatti equation, and then Riccatti equation is solved by an iterative solution.

12.1.21.2 Constant Temperature Simulations: Nose-Hoover Thermostats

Nose-Hoover Thermostats for the electrons and ions can also be added to the Car-Parrinello simulation. In this type of simulation thermostats variables x_e and x_R are added to the simulation by adding the auxiliary energy functionals to the total energy.

$$ION_{THERMOSTAT}(x_R) = \frac{1}{2} Q_R \dot{x_R} + E_{R0}x_R$$
$$ELECTRON_{THERMOSTAT}(x_e) = \frac{1}{2} Q_e \dot{x_e} + E_{e0}x_e$$

In these equations, the average kinetic energy for the ions is

$$\begin{matrix}E_{R0} = \frac{1}{2} f k_B T \end{matrix}$$

where f is the number of atomic degrees of freedom,k_B is Boltzmann's constant, and T is the desired temperature. Defining the average fictitious kinetic energy of the electrons is not as straightforward. Blöchl and Parrinello (P.E. Blöchl and M. Parrinello, Phys. Rev. B, **45**, 9413, (1992)) have suggested the following formula for determining the average fictitious kinetic energy

$$\begin{matrix}E_{e0} = 4 k_B T \frac{\mu}{M} \sum_{i} <\psi_i|-\frac{1}{2} \nabla^2 |\psi_{i}> \end{matrix}$$

where μ is the fictitious electronic mass, M is average mass of one atom, and $\sum_{i} <\psi_i|-\frac{1}{2} \nabla^2 |\psi_{i}>$ is the kinetic energy of the electrons.

Blöchl and Parrinello suggested that the choice of mass parameters,Q_e), and Q_R should be made such that the period of oscillating thermostats should be chosen larger than the typical time scale for the dynamical events of interest but shorter than the simulation time.

$$P_{ion} = 2\pi \sqrt{\frac{Q_R}{4E_{R0}}}$$
$$P_{electron} = 2\pi \sqrt{\frac{Q_e}{4E_{e0}}}$$

where P_{ion} and $P_{electron}$ are the periods of oscillation for the ionic and fictitious electronic thermostats.

In simulated annealing simulations the electronic and ionic Temperatures are scaled according to an exponential cooling schedule,

$$T_e(t) = T_e^0 \exp^{-\frac{t}{\tau_e}}$$
$$T_{ionic}(t) = T_{ionic}^0 \exp^{-\frac{t}{\tau_{ionic}}}$$

where T_e^0 and T_{ionic}^0 are the initial temperatures, and τ_e and τ_{ionic} are the cooling rates in atomic units.

12.1.22 NWPW Tutorial 1: S_2 dimer examples with PSPW

A description of all the examples in NWPW Tutorial 1 can be found in the attached pdf nwpwexample1.pdf

12.1.22.1 Total energy of S_2 dimer with LDA approximation

(input:Media:s2-example1.nw, output:Media:s2-example1.nwout)

In this example, the total energy of the S_2 dimer using LDA approximation for the exchange-correlation functional is calculated.

```
echo
title "total energy of s2-dimer LDA/30Ry with PSPW method"
scratch_dir   ./scratch
permanent_dir ./perm
start s2-pspw-energy
geometry
S 0.0 0.0 0.0
S 0.0 0.0 1.88
end
nwpw
  simulation_cell
    SC 20.0
  end
  cutoff 15.0
  mult 3
  xc lda
  lmbfgs
end
task pspw energy
```

The energies from the simulation will be

```
...
==  Summary Of Results  ==

 number of electrons: spin up=    7.00000  down=    5.00000 (real space)

 total     energy    : -0.2041363137E+02 (   -0.10207E+02/ion)
 total orbital energy:  -0.4944372503E+01 (   -0.41203E+00/electron)
 hartree   energy    :  0.1680529987E+02 (    0.14004E+01/electron)
 exc-corr  energy    : -0.4320620600E+01 (   -0.36005E+00/electron)
 ion-ion   energy    :  0.8455644190E-02 (    0.42278E-02/ion)

 kinetic (planewave) :  0.7529965882E+01 (    0.62750E+00/electron)
 V_local (planewave) : -0.4506036741E+02 (   -0.37550E+01/electron)
 V_nl    (planewave) :  0.4623635248E+01 (    0.38530E+00/electron)
 V_Coul  (planewave) :  0.3361059973E+02 (    0.28009E+01/electron)
 V_xc.   (planewave) : -0.5648205953E+01 (   -0.47068E+00/electron)
 Virial Coefficient  : -0.1656626150E+01

 orbital energies:
   -0.2001309E+00 (  -5.446eV)
   -0.2001309E+00 (  -5.446eV)
   -0.3294434E+00 (  -8.965eV)    -0.2991148E+00 (  -8.139eV)
   -0.3294435E+00 (  -8.965eV)    -0.2991151E+00 (  -8.139eV)
   -0.3582269E+00 (  -9.748eV)    -0.3352434E+00 (  -9.123eV)
   -0.5632339E+00 ( -15.326eV)    -0.5246249E+00 ( -14.276eV)
   -0.7642738E+00 ( -20.797eV)    -0.7413909E+00 ( -20.174eV)

 Total PSPW energy   : -0.2041363137E+02
...
```

12.1.22.2 Structural optimization of S_2 dimer with LDA approximation

(input:Media:s2-example2.nw, output:Media:s2-example2.nwout)

In this example, the structure of the S_2 dimer using results generated from prior energy calculation is calculated. Since most of the parameters are already stored in the run-time database the input is very simple.

```
echo
title "optimization of s2-dimer LDA/30Ry with PSPW method"
scratch_dir   ./scratch
permanent_dir ./perm
restart s2-pspw-energy
driver
  maxiter 20
  xyz s2
end
task pspw optimize
```

As the optimization process consists of series of total energy evaluations the contents of the output file are very much similar to that in Example I. At each step the total energy and force information will be outputed as follows

```
 Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
 ---- ---------------- -------- -------- -------- -------- -------- --------
@    1    -20.41364254 -7.1D-05  0.00004  0.00004  0.00605  0.01048      7.8
```

The best way to keep track of the optimization calculation is to run the following grep command on the output file.

```
grep @ outputfile

@ Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
@ ---- ---------------- -------- -------- -------- -------- -------- --------
@    0    -20.41357202  0.0D+00  0.00672  0.00672  0.00000  0.00000      1.5
@    1    -20.41364254 -7.1D-05  0.00004  0.00004  0.00605  0.01048      7.8
@    2    -20.41364256 -2.3D-08  0.00020  0.00020  0.00003  0.00005      9.7
@    2    -20.41364256 -2.3D-08  0.00020  0.00020  0.00003  0.00005      9.7
```

The optimized energy and geometry will be

```
...
     ----------------------
     Optimization converged
     ----------------------


  Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
  ---- ---------------- -------- -------- -------- -------- -------- --------
@    2    -20.41364256 -2.3D-08  0.00020  0.00020  0.00003  0.00005      9.7
                           ok       ok       ok       ok


                        Z-matrix (autoz)
                        --------

  Units are Angstrom for bonds and degrees for angles

     Type          Name      I    J    K    L    M      Value     Gradient
     ----------- --------  ----- ----- ----- ----- ----- ---------- ----------
   1 Stretch                  1     2                    1.89115    0.00020


                    Geometry "geometry" -> "geometry"
                    ---------------------------------

  Output coordinates in angstroms (scale by  1.889725989 to convert to a.u.)

  No.       Tag         Charge          X              Y              Z
 ---- ---------------- ---------- -------------- -------------- --------------
    1 S               16.0000     0.00000000     0.00000000    -0.94557591
    2 S               16.0000     0.00000000     0.00000000     0.94557591

...
```

12.1.22.3 Frequency calculation of S_2 dimer with LDA approximation

(input:Media:s2-example3.nw, output:Media:s2-example3.nwout)

In this example, the vibrational frequency of the S_2 dimer using results generated from prior geometry optimization is calculated. Since most of the parameters are already stored in the run-time database the input is very simple.

```
echo
title "frequency calculation of s2-dimer LDA/30Ry with PSPW method"
scratch_dir   ./scratch
permanent_dir ./perm
restart s2-pspw-energy
freq
  animate
end
task pspw freq
```

The frequency and thermodynamic analysis generated

```
...
 Temperature                   =   298.15K
 frequency scaling parameter   =   1.0000


 Linear Molecule

 Zero-Point correction to Energy  =    1.034 kcal/mol  (  0.001647 au)
 Thermal correction to Energy     =    2.579 kcal/mol  (  0.004110 au)
 Thermal correction to Enthalpy   =    3.171 kcal/mol  (  0.005054 au)

 Total Entropy                 =   52.277 cal/mol-K
  - Translational              =   38.368 cal/mol-K (mol. weight =  63.9441)
  - Rotational                 =   13.630 cal/mol-K (symmetry #  =       2)
  - Vibrational                =    0.279 cal/mol-K

 Cv (constant volume heat capacity) =    5.750 cal/mol-K
  - Translational              =    2.979 cal/mol-K
  - Rotational                 =    1.986 cal/mol-K
  - Vibrational                =    0.785 cal/mol-K
...
 ------------------------------------------------------------------------------
 Normal Eigenvalue ||           Projected Infra Red Intensities
  Mode   [cm**-1]  || [atomic units] [(debye/angs)**2] [(KM/mol)] [arbitrary]
 ------ ---------- || -------------- ----------------- ---------- -----------
    1       0.000 ||    0.000030          0.001        0.029       0.000
    2       0.000 ||    2.466908         56.914     2404.864      15.000
    3       0.000 ||    2.466908         56.914     2404.864      15.000
    4       0.000 ||    2.466908         56.914     2404.864      15.000
    5       0.000 ||    2.466908         56.914     2404.864      15.000
    6     723.419 ||    0.000000          0.000        0.000       0.000
 ------------------------------------------------------------------------------
...
```

12.1.22.4 Ab initio molecular dynamics simulation (Car-Parrinello) of S_2 dimer using the LDA approximation

(input:Media:s2-example4.nw, output:Media:s2-example4.nwout Media:s2-md.xyz Media:s2-md.emotion.dat)

In this example, a constant energy Car-Parrinello simulation of S_2 dimer using LDA approximation is calculated. A brief introduction to the Car-Parrinello method can be found in cpmd-lecture.pdf

```
echo
title "AIMD simulation of s2-dimer"
scratch_dir   ./scratch
permanent_dir ./perm
start s2-md
geometry
S 0.0 0.0 0.0
S 0.0 0.0 1.95
end
nwpw
  simulation_cell
    SC 20.0
  end
  cutoff 15.0
  mult 3
  xc lda
  lmbfgs
  car-parrinello
    time_step 5.0
    fake_mass 600.0
    loop 1 1000
    xyz_filename s2-md.xyz
  end
end
task pspw energy
task pspw car-parrinello
```

A plotting program (e.g. gnuplot, xmgrace) can be used to look at the total, potential, kinetic energies, contained in the s2-md.emotion file (see section EMOTION motion file for datafile format) i.e.,

```
seattle-1604% gnuplot

        G N U P L O T
        Version 4.0 patchlevel 0
        last modified Thu Apr 15 14:44:22 CEST 2004
        System: Linux 2.6.18-194.8.1.el5

        Copyright (C) 1986 - 1993, 1998, 2004
        Thomas Williams, Colin Kelley and many others

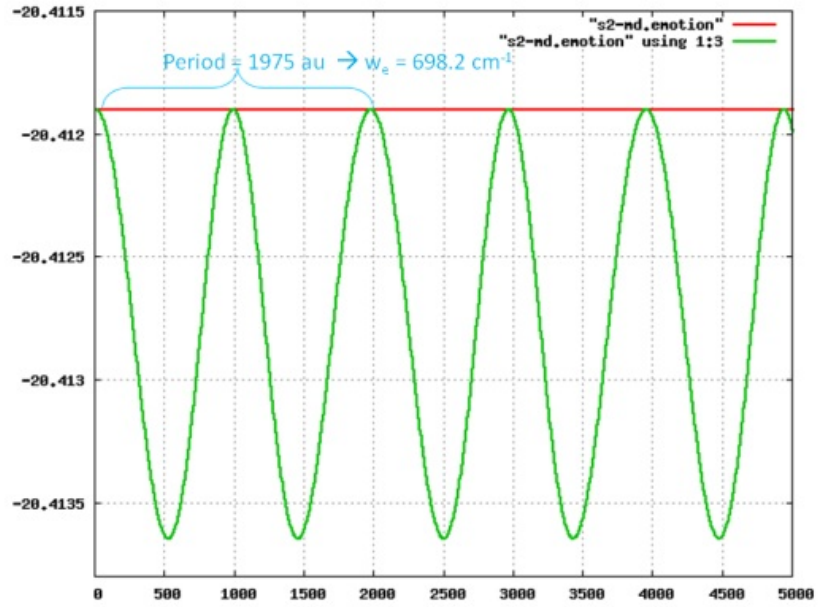
        This is gnuplot version 4.0.  Please refer to the documentation
        for command syntax changes.  The old syntax will be accepted
        throughout the 4.0 series, but all save files use the new syntax.

        Type help to access the on-line reference manual.
        The gnuplot FAQ is available from
                <http://www.gnuplot.info/faq/>

        Send comments and requests for help to
                <gnuplot-info@lists.sourceforge.net>
        Send bugs, suggestions and mods to
                <gnuplot-bugs@lists.sourceforge.net>


Terminal type set to 'x11'
gnuplot> plot "s2-md.emotion","s2-md.emotion" using 1:3
gnuplot>
```

The following plot shows the Car-Parrinello \(^3\Sigma_g^-\) S\(_2\) energy surface generated from the simulation.



12.1.22.5 Ab initio molecular dynamics simulation (Born-Oppenheimer) of S<sub2 dimer using the LDA approximation

(input:Media:s2-example5.nw, output:Media:s2-example5.nwout Media:s2-bomd.xyz Media:s2-bomd.emotion.dat) In this example, a constant energy Born-Oppenheimer simulation of S2 dimer using LDA approximation is calculated.

```
title "AIMD simulation of s2-dimer"
echo

scratch_dir   ./scratch
permanent_dir ./perm

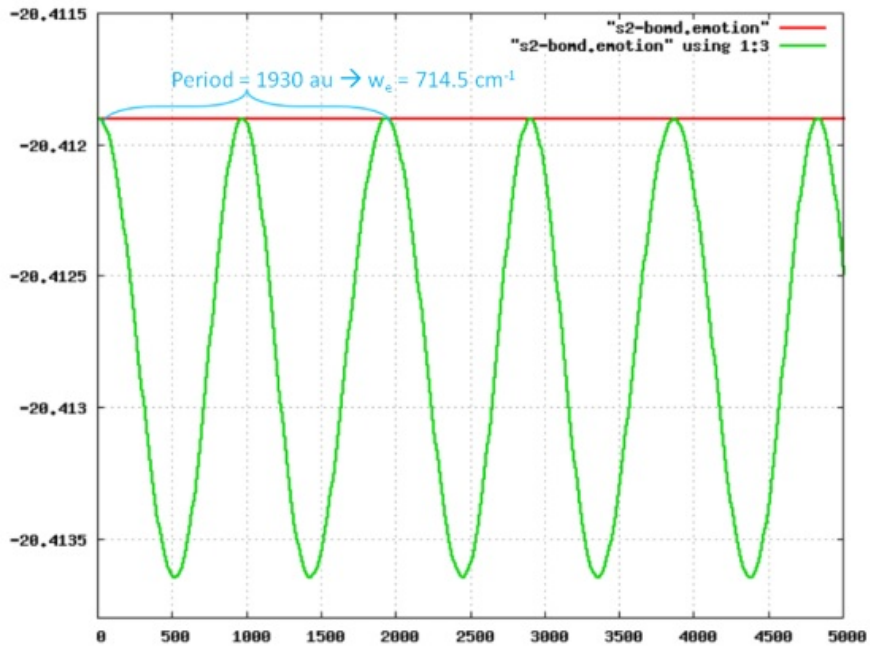
start s2-bomd

geometry
S 0.0 0.0 0.0
S 0.0 0.0 1.95
end

nwpw
  simulation_cell
    SC 20.0
  end
  cutoff 15.0
  mult 3
  xc lda
  lmbfgs
end
task pspw energy

nwpw
   bo_steps 1 500
   bo_time_step 10.0
end
task pspw born-oppenheimer
```

The following plot shows the \(^3\Sigma_g^{-}_{2}\) energy surface generated from the simulation.

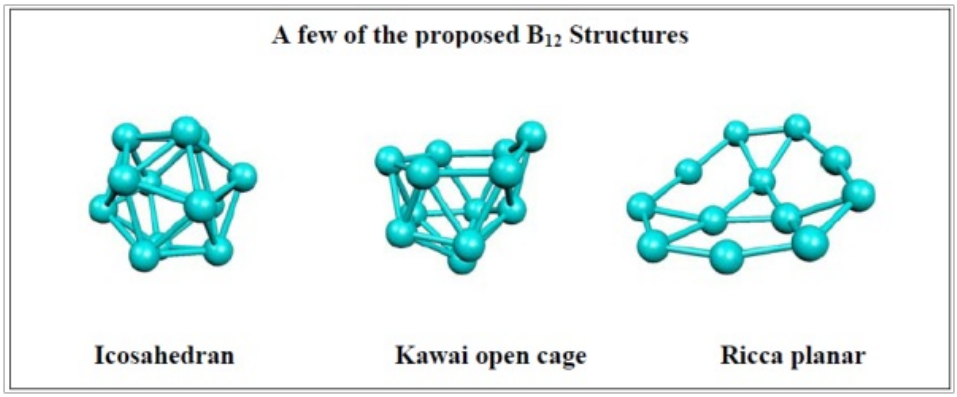


12.1.23 NWPW Tutorial 2: Using PSPW Car-Parrinello Simulated Annealing Simulations to Optimize Structures

In principle quantum mechanical calculations can be used to determine the structure of any chemical system. One chooses a structure, calculates the total energy of the system, and repeats the calculation for all possible geometries. Of course the major limitation of this approach is that the number of local minima structures increases dramatically with system size and the cost of quantum mechanical calculations also increases dramatically with system size. Not surprisingly most quantum mechanical calculations limit the number of structures to be calculated by using experimental results or chemical intuition. One could speed up the calculations by using simplified inter-atomic force fields instead of quantum mechanical calculations. However, inter-atomic forces fields have many simplifying assumptions that can severely limit their predictability. Another approach is to use ab initio molecular dynamics methods combined with simulated annealing. These methods are quite robust and allow strongly interacting many body systems to be studied by direct dynamics simulation

without the introduction of empirical interactions. In these methods, the atomic forces are calculated from ab initio calculations that are performed “on-the-fly” as the molecular dynamics trajectory is generated.

The following examples demonstrate how to use the ab initio molecular dynamics methods and simulated annealing strategies of NWChem to determine the lowest energy structures of the B_{12} cluster. This example is based on a study performed by Kiran Boggavarapu et al.. One might expect from chemical intuition that lowest energy structure of B_{12} will be an icosahedron, since B_{12} icosahedra are a common structural unit found in many boron rich materials. Despite this prevalence, ab initio calculations performed by several researchers have suggested that B_{12}, as well as B_{12}^{+} and B_{12}^{-}, will have a more open geometry.



12.1.23.1 Simulated Annealing Using Constant Energy Simulation

(input:Media:b12-example2a.nw, output:Media:b12-example2a.nwout Media:b12.00.xyz Media:b12.00.emotion.dat Media:b12.01.xyz Media:b12.01.emotion.dat)

This example uses a series of constant energy Car-Parrinello simulations with velocity scaling to do simulated annealing. The initial four Car-Parrinello simulations are used to heat up the system to several thousand Kelvin. Then the system is cooled down thru a series of constant energy simulations in which the electronic and ionic velocities are scaled by 0.99 at the start of each Car-Parrinello simulation. Energy minimization calculations are used periodically in this simulation to bring the system back down to Born-Oppenheimer surface. This is necessary because of electronic heating.

The Car-Parrinello keyword “scaling” scales the wavefunction and ionic velocities at the start of the simulation. The following input is used to increase the ionic velocities by a factor of two at the start of the Car-Parrinello simulation.

Key Input

```
...
Car-Parrinello
fake_mass 500.0
time_step 5.0
loop 10 100
** scaling 1.0 2.0**
emotion_filename b12.00.emotion
xyz_filename     b12.00.xyz
end
...
```

Output

```
...
    wavefnc cutoff= 10.000  fft= 42x 42x 42(     6027 waves     1004 per task)

technical parameters:
    translation contrained
    time step=      5.00     ficticious mass=     500.0
    **cooling/heatting rates: 0.10000E+01 (psi)
0.20000E+01
(ion)**
    maximum iterations =    1000 (   10 inner  100 outer )
    initial kinetic energy:  0.99360E-05 (psi)   0.27956E-03 (ion)
                                 0.20205E-28 (c.o.m.)
    **after scaling: 0.99360E-05 (psi) 0.11182E-02
(ion)**
    **increased energy: 0.00000E+00 (psi)
0.83868E-03 (ion)**

Constant Energy Simulation
...
```

The program checks to see if the initial input ionic velocities have a non-zero center of mass velocity. If there is a non-zero center of mass velocity in the system then by default the program removes it. To turn off this feature set the following

```
nwpw
  translation on
end
```

or

```
set nwpw:com_shift .false.
```

12.1.23.2 Simulated Annealing Using Constant Temperature Simulation

(input:Media:b12-example2b.nw, output:Media:b12-example2b.nwout Media:b12.10.xyz Media:b12.10.emotion.dat Media:b12.11.xyz.gz Media:b12.11.emotion.dat)

(mpeg movie of simulation: Media:boron.mpg)

The simulated annealing calculation in this example uses a constant temperature Car-Parrinello simulation with an exponential cooling schedule,

$$T(t)=T_0e^{-t/\tau}$$

where T0 and τ are an initial temperature and a time scale of cooling, respectively. In the present calculations T0=3500K and τ=4.134e+4 au (1.0 ps) were used and the thermostat masses were kept fixed to the initial values determined by T=Te=3500K and (2π/ω)=250 a.u. (6 fs). Annealing proceeded for 50000 steps, until a temperature of 10K was reached. After which, the metastable structure is optimized using the driver optimizer. The keyword SA_decay is used to enter the decay rates, τelectron and τion, used in the simulated annealing algorithm in the constant temperature car-parrinello simulation. The decay rates are in units of au (conversion 1 au = 2.41889e-17 seconds).

Key Input

```
....
Car-Parrinello
SA_decay 4.134d4 4.134d4 #decay rate in units of au (1au=2.41889e-17seconds)
....
```

12.1.24 NWPW Tutorial 3: using isodesmic reaction energies to estimate gas-phase thermodynamics

(isodesmic.pdf isodesmic.tgz)

The development of a computational scheme that can accurately predict reaction energies requires some care. As shown in Table 1 energy errors associated with ab initio calculations can be quite high. Even though ab initio electronic structure

methods are constantly being developed and improved upon, these methods are rarely able to give heat of formations of a broad class of molecules with error limits of less than a few kcal/mol. Only when very large basis sets such as the correlation-consistent basis sets, high level treatments of correlation energy such as coupled cluster methods (CCSD(T)), and small correction factors such as core-valence correlation energies and relativistic effects, are included will the heat of formation from ab initio electronic structure methods be accurate to within one kcal/mol. Although one can now accurately calculate the heats of formation of molecules with up to 6 first row atoms, such high-level calculations are extremely demanding and scale computationally as N^7 for N basis functions.

Examples of these types of large errors are shown in the following Table, where the enthalpies of formation of CCl_3SH are calculated by using atomization energies from different levels of ab initio theory.

	MP2/cc-pVDZ	LDA/DZVP2	BP91/DZVP2	B3LYP/DZVP2	G2 Theory
$\Delta H\ (_f^o)$	+4.9	-80.0	-2.6	+26.5	-13.0

Table 1: Standard enthalpy of formation ($\Delta H(_f^o)$(298K) for CCl_3SH in kcal/mol from atomization energies with various electronic structure methods. Results taken from reference [2].

Differences of up to 106.5 kcal/mol are found between different levels of theory. This example demonstrates that care must be taken in choosing the appropriate method for calculating the heats of formation from total atomization energies.

The difficulties associated with calculating absolute heats of formation from atomization energies can be avoided by using a set of isodesmic reactions[1]. The defining property of an isodesmic reaction - that there are an equal number of like bonds on the left-hand and right-hand sides of the reaction - helps to minimize the error in the reaction energy. These reactions are designed to separate out the interactions between molecular subsistents and non-bonding electrons from the direct bonding interactions by having the direct bonding interactions largely canceling one another. This separation is quite attractive. Most ab initio methods give substantial errors when estimating direct bonding interactions due to the computational difficulties associated with electron pair correlation, whereas ab initio methods are expected to be more accurate for estimating neighboring interactions and long-range through-bond effects.

The following isodesmic reaction can be used determine the enthalpy of formation for CCl_3SH that is significantly more accurate than the estimates based on atomization energies.

$CCl_3SH + CH_4 \rightarrow CH_3SH + CCl_3H$, ΔH_r(calc).

The first step is to calculate the reaction enthalpy of this reaction from electronic, thermal and vibrational energy differences at 298.15K at a consistent level of theory. The defining property of an isodesmic reaction that there are an equal number of like bonds on the left-hand and right-hand sides of the reaction helps to minimize the error in the calculation of the reaction energy. The enthalpy of formation of CCl_3SH can then be calculated by using Hess's law with the calculated enthalpy change and the experimentally known heats of formation of the other 3 species (see Table 3).

$\Delta H_f(CCl_3SH) = \Delta H_f(CH_3SH)(exp) + \Delta H_f(CCl_3H)(exp) - \Delta H_f(CH_4))(exp) - \Delta H_r$(calc).

In this example, try to design and run NWPW simulations that can be used to estimate the enthalpy of formation for CCl_3SH using its atomization energy and using the reaction enthalpy of the isodesmic reaction and compare your results to Table 2. Be careful to make sure that you use the same cutoff energy for all the simulations (.e.g. cutoff 35.0). You might also try to estimate enthalpies of formation for $CHCl_2SH$ and CH_2ClSH. Also try designing simulations that use the SCF, DFT, MP2, and TCE modules.

$CCl_3SH + CH_4 \rightarrow CH_3SH + CCl_3H$

Un-optimized geometries for CCl_3SH, CH_3SH, CCl_3H and CH_4 which are needed to design your simulations are contained in the file Media:thermodynamics.xyz. You will also need to calculate the energies for the H, C, S, and Cl atoms to calculate the atomization energies. The multiplicities for these atoms are 2, 3, 3 and 2 respectively. You

will also need to calculate the enthalpy of a molecule. The enthalpy of a molecule at 298.15K is sum of the total energy and a thermal correction to the enthalpy. A good estimate for the thermal correction to the enthalpy can be obtained from a frequency calculation, i.e.

H = E + H\(_{correction}\)

Thermodynamic output from a frequency calculation:

```
Temperature                   =   298.15K
frequency scaling parameter   =   1.0000

Zero-Point correction to Energy  =   27.528 kcal/mol  (  0.043869 au)
Thermal correction to Energy     =   29.329 kcal/mol  (  0.046739 au)
```

The following line contains the value for H\(_{correction}\)

```
Thermal correction to Enthalpy   =   29.922 kcal/mol  (  0.047683 au)

Total Entropy                 =   44.401 cal/mol-K
 - Translational              =   34.246 cal/mol-K (mol. weight =  16.0313)
 - Rotational                 =   10.060 cal/mol-K (symmetry #  =       12)
 - Vibrational                =    0.095 cal/mol-K

Cv (constant volume heat capacity) =    6.503 cal/mol-K
 - Translational              =    2.979 cal/mol-K
 - Rotational                 =    2.979 cal/mol-K
 - Vibrational                =    0.544 cal/mol-K
```

Compounds	MP2/cc-pVDZ	LDA/DZVP2	BP91/DZVP2	B3LYP/DZVP2	G2	Experiment
	(isodesmic)	(isodesmic)	(isodesmic)	(isodesmic)	(atomization)	
CCl$_3SH	-13.40	-11.86	-8.68	-7.64	-12.95	
CHCl\(_2\)SH	-11.48	-11.07	-8.66	-7.92	-11.52	
CH\(_2\)ClSH	-7.01	-6.66	-5.44	-5.20	-6.98	
CH\(_3\)SH					-4.76	-5.34

Compounds	ΔH\(_f^o\)(298K)
H	52.095
C	171.291
S	66.636
Cl	29.082
CCl\(_4\)	-24.59
CCl\(_3\)H	-24.65
CCl\(_2\)H\(_2\)	-22.10
CClH\(_3\)	-19.32
CH\(_4\)	-17.88
CH\(_3\)SH	-5.34

Table 1: Gas-phase standard enthalpies of formation (ΔH

Table 3: Miscellaneous experimental gas-phase enthalpies of formation (kcal/mol) taken from [3].

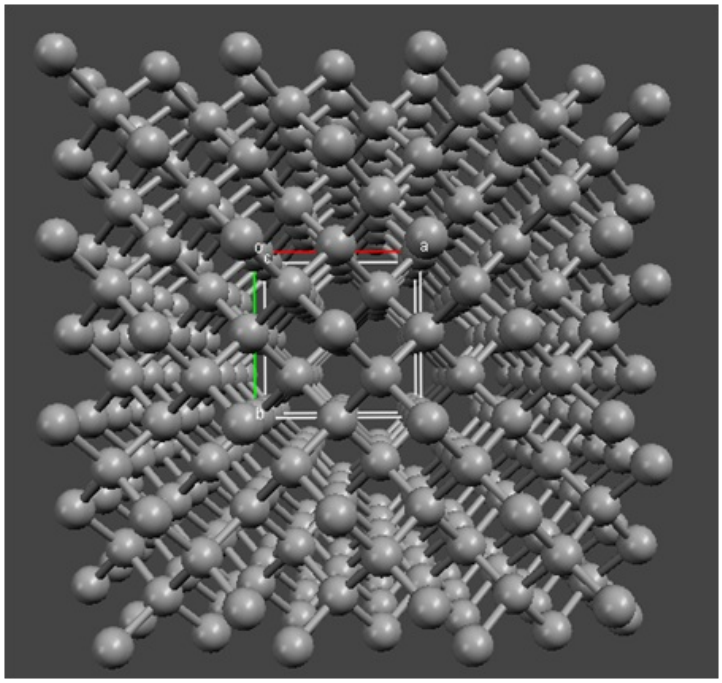
1. Hehre, W. J., L. Radom, P.v.R. Schleyer, and J.A. Pople Ab Initio Molecular Orbital Theory; John Wiley & Sons: New York, 1986).
2. E.J. Bylaska, D.A. Dixon, and A.R. Felmy(2000), “The Free Energies of Reactions of Chlorinated Methanes with Aqueous Monovalent Anions: Application of ab initio Electronic Structure Theory”, J. Phys. Chem. A, 104(3), 610-617.
3. Chase, M. W., Jr. Phys. Chem. Ref. Data, Monograph No. 9 1998, 9, 1-1951.

12.1.25 NWPW Tutorial 4: AIMD/MM simulation of CCl\(_4\) + 64 H\(_2\)O

(input:Media:ccl4-64water.nw, output:Media:ccl4-64water.nwout)

In this section we show how use the PSPW module to perform a Car-Parrinello AIMD/MM simulation for a CCl\(_4\) molecule in a box of 64 H\(_2\)O. Before running a PSPW Car-Parrinello simulation the system should be on the Born-Oppenheimer surface, i.e. the one-electron orbitals should be minimized with respect to the total energy (i.e. task pspw energy). In this example, default pseudopotentials from the pseudopotential library are used for C, Cl, O^ and H^, exchange correlation functional is PBE96, The boundary condition is periodic, and with a side length of 23.577 Bohrs and has a cutoff energy is 50 Ry). The time step and fake mass for the Car-Parrinello run are specified to be 5.0 au and 600.0 au, respectively.

12.1.26 NWPW Tutorial 5: Optimizing the Unit Cell and Geometry of Diamond



The PSPW and BAND codes can be used to determine structures and energies for a wide range of crystalline systems. It can also be used to generate band structure and density of state plots.

12.1.26.1 Optimizing the Unit Cell and Geometry for an 8 Atom Supercell of Diamond with PSPW

(input:Media:diamond-pspw.nw, output:Media:diamond-pspw.nwout, Media:diamond.opt.cif)

(input:Media:catom-pspw.nw, output:Media:catom-pspw.nwout)

The following example uses the PSPW module to optimize the unit cell and geometry for a diamond crystal. The fractional coordinates and the unit cell are defined in the geometry block. The simulation_cell block is not needed since NWPW automatically uses the unit cell defined in the geometry block.

```
title "Diamond 8 atom cubic cell - geometry and unit cell optimization"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamond

memory 950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 3.56d0
    lat_b 3.56d0
    lat_c 3.56d0
    alpha 90.0d0
    beta  90.0d0
    gamma 90.0d0
  end
  C -0.50000d0 -0.50000d0 -0.50000d0
  C  0.00000d0  0.00000d0 -0.50000d0
  C  0.00000d0 -0.50000d0  0.00000d0
  C -0.50000d0  0.00000d0  0.00000d0
  C -0.25000d0 -0.25000d0 -0.25000d0
  C  0.25000d0  0.25000d0 -0.25000d0
  C  0.25000d0 -0.25000d0  0.25000d0
  C -0.25000d0  0.25000d0  0.25000d0
end

nwpw
  ewald_rcut 3.0
  ewald_ncut 8  #The default value of 1 needs to be increased for small cells
  lmbfgs
  xc pbe96
end

driver
  clear
  maxiter 40
end

set nwpw:cif_filename diamond.opt  # create a CIF file containing optimization history
set includestress .true.           # this option tells driver to optimize the unit cell
task pspw optimize ignore
```

The optimized energy and geometry will be

...

Optimization converged

Step	Energy	Delta E	Gmax	Grms	Xrms	Xmax	Walltime
@ 6	-45.07688304	-1.1D-07	0.00037	0.00021	0.00002	0.00003	174.5
		ok	ok	ok	ok		

Geometry "geometry" -> "geometry"

Output coordinates in angstroms (scale by 1.889725989 to convert to a.u.)

No.	Tag	Charge	X	Y	Z
1	C	6.0000	1.82723789	1.82729813	1.82705440
2	C	6.0000	0.00000857	-0.00006053	1.82730027
3	C	6.0000	-0.00000584	1.82706061	0.00002852
4	C	6.0000	1.82712018	0.00006354	-0.00002544
5	C	6.0000	2.74074195	2.74072805	2.74088522
6	C	6.0000	0.91366407	0.91370055	2.74064976
7	C	6.0000	0.91351181	2.74080771	0.91352917
8	C	6.0000	2.74078843	0.91348115	0.91365446

Lattice Parameters

---

lattice vectors in angstroms (scale by 1.889725989 to convert to a.u.)

a1=< 3.654 0.000 0.000 >
a2=< 0.000 3.654 0.000 >
a3=< 0.000 0.000 3.654 >
a= 3.654 b= 3.654 c= 3.654
alpha= 90.000 beta= 90.000 gamma= 90.000
omega= 48.8

reciprocal lattice vectors in a.u.

b1=< 0.910 0.000 0.000 >
b2=< 0.000 0.910 0.000 >
b3=< 0.000 0.000 0.910 >

Atomic Mass

C 12.000000

==========

internuclear distances

---

center one	center two	atomic units	angstroms
5 C	1 C	2.99027	1.58238
6 C	1 C	2.99027	1.58238
6 C	2 C	2.99027	1.58238
7 C	1 C	2.99026	1.58238
7 C	3 C	2.99027	1.58238
8 C	1 C	2.99027	1.58238
8 C	4 C	2.99027	1.58238

---

number of included internuclear distances: 7

==========

==========

internuclear angles

---

center 1	center 2	center 3	degrees
5 C	1 C	6 C	109.46
5 C	1 C	7 C	109.48
5 C	1 C	8 C	109.48
6 C	1 C	7 C	109.47
6 C	1 C	8 C	109.46
7 C	1 C	8 C	109.48
1 C	6 C	2 C	109.48
1 C	7 C	3 C	109.47
1 C	8 C	4 C	109.47

---

number of included internuclear angles: 9

========== ...

The C-C bond distance after the geometry optimization is 1.58 Angs. and agrees very well with the experimental value of 1.54 Angs.. Another quantity that can be calculated from this simulation is the cohesive energy.The cohesive energy of a crystal is the energy needed to separate the atoms of the solid into isolated atoms, i.e.

$$ E_{coh}= -\left(E_{solid} - \sum_{a} E_{atom}^a \right) $$

where E_{solid} is the energy of the solid andE_{atom}^a are the energies of the isolated atoms. In order to calculate the cohesive energy the energy of an isolated carbon atom at the same level of theory and cutoff energy will need to be calculated. The following input can be used to the energy of an isolated carbon atom.

(input:file:catom-pspw.nw, output:file:catom-pspw.nwout)

```
title "triplet carbon atom at pbe96 level using a large unit cell"
start c1-pspw
memory 1400 mb

permanent_dir ./perm
scratch_dir   ./scratch

geometry
C 0 0 0
end

nwpw
  simulation_cell
    FCC 38.0     #large unit cell
    boundary_conditions   periodic  # periodic boundary conditions are used by default.
    #boundary_conditions aperiodic  # free-space (or aperiodic) boundary conditions could also be used.
  end
  xc pbe96
  mult 3
 lmbfgs
end
task pspw energy
```

The total energy from the simulation will be

```
 Total PSPW energy : -0.5421213534E+01
```

Using this energy and energy of diamond the cohesive energy per atom is calculated to be

$$\begin{align} E_{coh}= -\left(-45.07688304au/8 - (-5.421213534au) \right)= 0.2133968 au = 5.8 eV \end{align}$$

This value is substantially lower than the experimental value of$7.37eV$! It turns out this error is a result of the unit cell being too small for the diamond calculation (or too small of a Brillioun zone sampling). In the next section, we show how increasing the Brillouin zone sampling reduces the error in the calculated cohesive energy.

12.1.26.2 Optimizing the Unit Cell for an 8 Atom Supercell of Diamond with BAND

(input:Media:diamond-band.nw, output:Media:diamond-band.nwout)

In this example the BAND module is used to optimize the unit cell and geometry for a diamond crystal at different Brillouin zone samplings.

```
title "Diamond 8 atom cubic cell - geometry and unit cell optimization"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamond-band

memory 1950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 3.58d0
    lat_b 3.58d0
    lat_c 3.58d0
    alpha 90.0d0
    beta  90.0d0
    gamma 90.0d0
  end
  C -0.50000d0 -0.50000d0 -0.50000d0
  C  0.00000d0  0.00000d0 -0.50000d0
  C  0.00000d0 -0.50000d0  0.00000d0
  C -0.50000d0  0.00000d0  0.00000d0
  C -0.25000d0 -0.25000d0 -0.25000d0
  C  0.25000d0  0.25000d0 -0.25000d0
  C  0.25000d0 -0.25000d0  0.25000d0
  C -0.25000d0  0.25000d0  0.25000d0
end
set includestress    .true.   # option tells driver to optimize the unit cell
set nwpw:zero_forces .true.   # option zeros the forces on the atoms--> only lattice parameters optimized

nwpw
  ewald_rcut 3.0
  ewald_ncut 8    #The default value of 1 needs to be increased
  lmbfgs
  xc pbe96
end

#1x1x1 k-point mesh
nwpw
  monkhorst-pack 1 1 1
end
set nwpw:cif_filename diamond111.opt
driver; clear; maxiter 40; end; task band optimize ignore

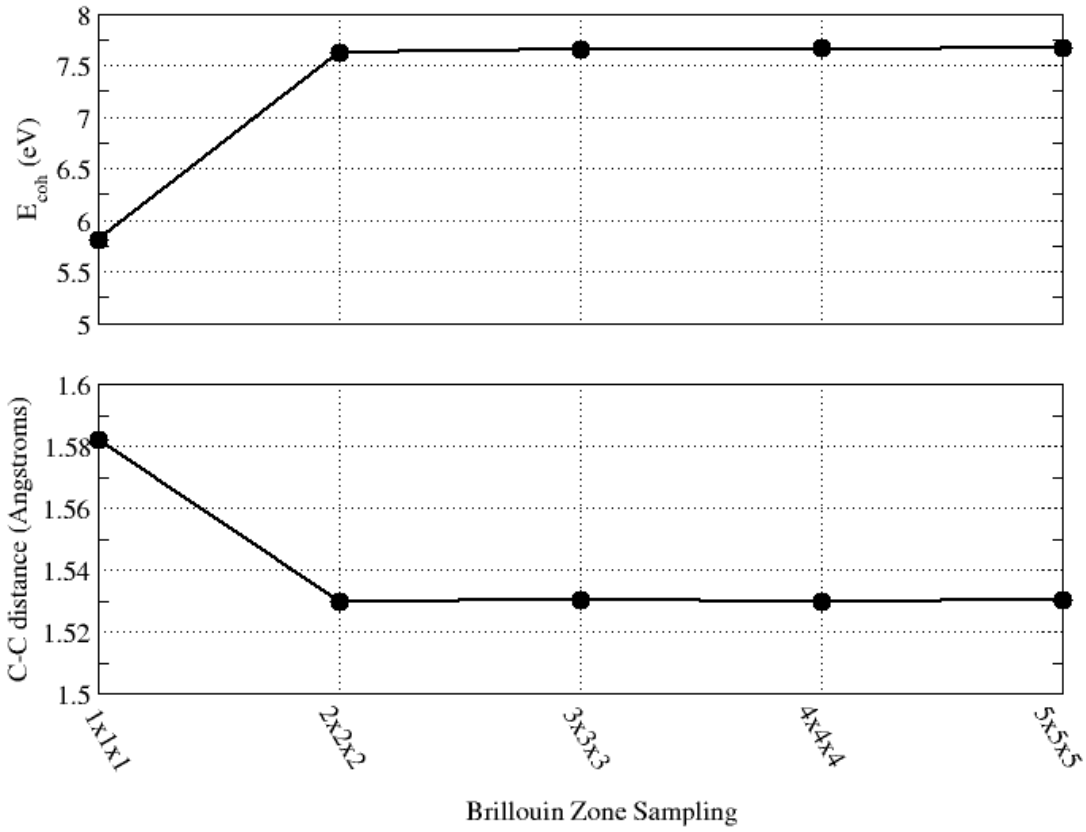
#2x2x2 k-point mesh
nwpw
  monkhorst-pack 2 2 2
end
set nwpw:cif_filename diamond222.opt
driver; clear; maxiter 40; end; task band optimize ignore

#3x3x3 k-point mesh
nwpw
  monkhorst-pack 3 3 3
end
set nwpw:cif_filename diamond333.opt
driver; clear; maxiter 40; end; task band optimize ignore

#4x4x4 k-point mesh
nwpw
  monkhorst-pack 4 4 4
end
set nwpw:cif_filename diamond444.opt
driver; clear; maxiter 40; end; task band optimize ignore

#5x5x5 k-point mesh
nwpw
  monkhorst-pack 5 5 5
end
set nwpw:cif_filename diamond555.opt
driver; clear; maxiter 40; end; task band optimize ignore
```

The following figure shows a plot of the cohesive energy and C-C bond distance versus the Brillouin zone sampling. As can be seen in this figure the cohesive energy (w/o zero-point correction) and C-C bond distance agree very well with the experimental values of 7.37 eV (including zero-point correction) and 1.54 Angs.



12.1.26.3 Using BAND to Optimize the Unit Cell for a 2 Atom Primitive Cell of Diamond

(input:Media:diamond-fcc.nw, output:Media:diamond-fcc.nwout.gz)

In this example the BAND module is used to optimize a 2 atom unit cell for a diamond crystal at different Brillouin zone samplings. The optimized energy and geometry will be (Monkhorst-Pack sampling of 11x11x11)

```
 ----------------------
 Optimization converged
 ----------------------

  Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
  ---- ---------------- -------- -------- -------- -------- -------- --------
@    1   -11.40586236  5.2D-07  0.00039  0.00018  0.00002  0.00003    662.0
                            ok       ok       ok       ok


                  Geometry "geometry" -> "geometry"
                  ---------------------------------

 Output coordinates in angstroms (scale by  1.889725989 to convert to a.u.)

  No.       Tag         Charge          X              Y              Z
 ---- ---------------- ---------- -------------- -------------- --------------
   1 C                  6.0000     0.00000000     0.00000000     0.00000000
   2 C                  6.0000     0.72201500     1.25056532     0.51054180

     Lattice Parameters
     ------------------

     lattice vectors in angstroms (scale by  1.889725989 to convert to a.u.)

     a1=<   2.165   1.251   0.001 >
     a2=<   0.001   2.500   0.001 >
     a3=<   0.722   1.251   2.041 >
     a=       2.500 b=       2.500 c=       2.500
     alpha=  59.966 beta=  59.966 gamma=  59.966
     omega=    11.0

     reciprocal lattice vectors in a.u.

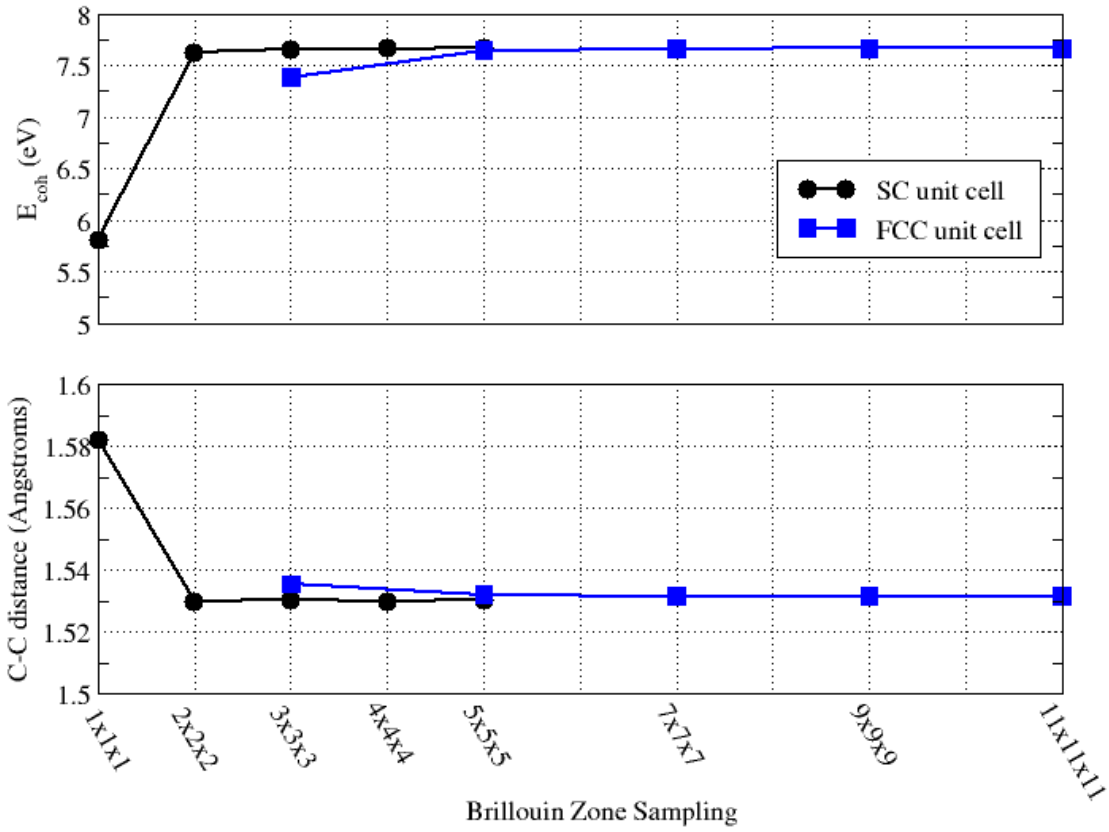
     b1=<   1.536  -0.768   0.000 >
     b2=<   0.000   1.330   0.000 >
     b3=<  -0.543  -0.543   1.629 >

     Atomic Mass
     -----------

     C                12.000000


 ================================================================================
                          internuclear distances
 --------------------------------------------------------------------------------
      center one      |      center two      | atomic units |  angstroms
 --------------------------------------------------------------------------------
   2 C                |   1 C                |     2.89435  |     1.53162
 --------------------------------------------------------------------------------
                     number of included internuclear distances:          1
 ================================================================================
```

The following figure shows a plot of the cohesive energy and C-C bond distance versus the Brillouin zone sampling for the 8 atom SC unit cell and the 2 atom FCC unit cell.



12.1.26.4 Using BAND to Calculate the Band Structures of Diamond

(input:Media:diamond-structure.nw, output:Media:diamond-structure.nwout, file:diamondfcc.restricted_band.dat)

The following example uses the BAND module to calculate the band structure for the FCC cell of the a diamond crystal. The fractional coordinates and the unit cell are defined in the geometry block. The simulation_cell block is not needed since NWPW automatically uses the unit cell defined in the geometry block.

```
title "Diamond 2 atom fcc cell Brillouin sampling=9x9x9 M-P - Band structure plot"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamondfcc

memory 1950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 2.500d0
    lat_b 2.500d0
    lat_c 2.500d0
    alpha 60.0d0
    beta  60.0d0
    gamma 60.0d0
  end
 C  0.00000d0  0.00000d0  0.00000d0
 C  0.25000d0  0.25000d0  0.25000d0
end

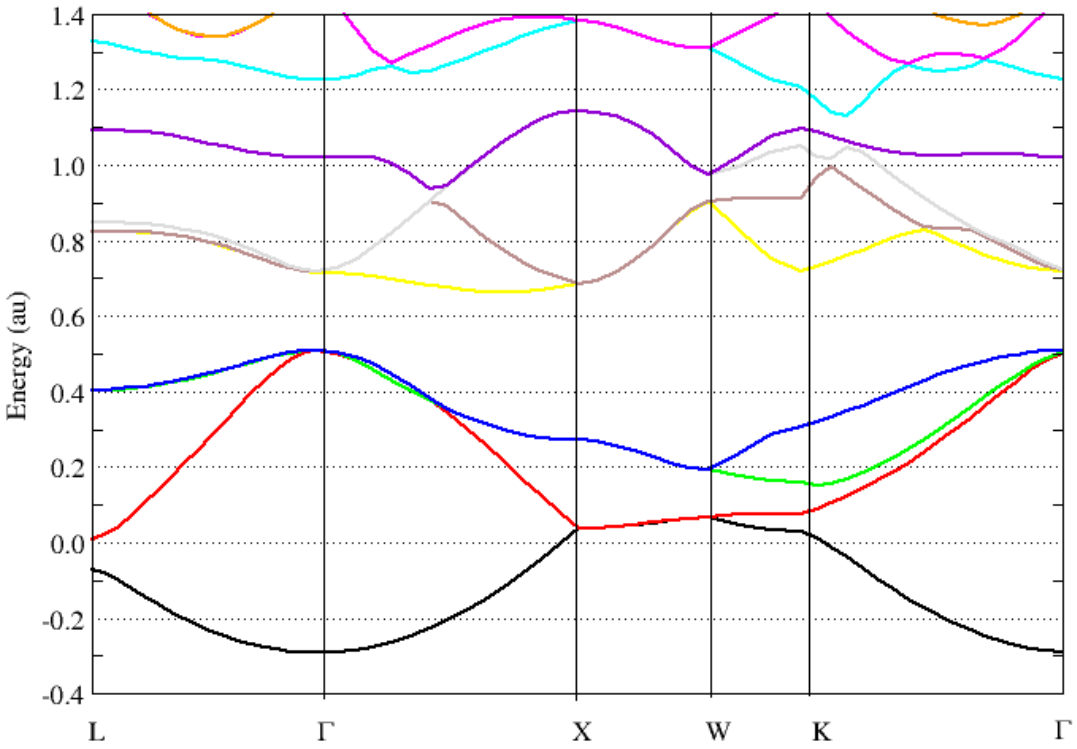
nwpw
  ewald_rcut 3.0
  ewald_ncut 8    #The default value of 1 needs to be increased
  lmbfgs
  xc pbe96

  monkhorst-pack 9 9 9
end

#need to run "task band energy" before "task band structure" can be run
task band energy

nwpw
   virtual 16
   brillouin_zone
     zone_name fccpath
     path fcc l gamma x w k gamma
   end
   zone_structure_name fccpath
end
task band structure
```

This calculation outputs the file:diamondfcc.restricted_band.dat) data file in the permanent_directory. A plotting (e.g. gnuplot or xmgrace) can be used to display the band structure.



12.1.26.5 Using BAND to Calculate the Density of States of Diamond

(2 atom cell - input:diamond-dos.nw output:diamond-dos.nwout, diamond-dos.dos.dat (8 atom cell - input:diamond-dos8.nw output: diamond-dos8.nwout.gz, diamond-dos8.dos.dat

There are two possible ways to use the BAND module to calculate the density and projected density of states. The first approach just uses the eigenvalues generated from an energy calculation to generate a density of states. The following example uses this strategy to calculate the density of states and projected density of states of diamond.

```
title "Diamond 2 atom fcc cell Brillouin sampling=9x9x9 M-P - density of states plot"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamond-dos

memory 1950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 2.500d0
    lat_b 2.500d0
    lat_c 2.500d0
    alpha 60.0d0
    beta  60.0d0
    gamma 60.0d0
  end
 C  0.00000d0  0.00000d0  0.00000d0
 C  0.25000d0  0.25000d0  0.25000d0
end

nwpw
  ewald_rcut 3.0
  ewald_ncut 8    #The default value of 1 needs to be increased
  lmbfgs
  xc pbe96

  monkhorst-pack 9 9 9
  dos                 # dos keyword tells the code to calculate dos at the end of an energy calculation
  mulliken              # turn on projected density of states
  virtual 8             # include 8 virtual states
end

task band energy
```

The other approach uses the band structure code to calculate the eigenvalues given a precomputed density. The approach is slower than the first approach, however, it can be used to substantially increase the number of k-points and virtual orbitals used to generate the density of states. The following example demonstrates this capability to calculate the density of states and projected density of states of the diamond crystal.

```
title "Diamond 2 atom fcc cell Brillouin sampling=9x9x9 M-P - density of states plot"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamond-dos

memory 1950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 2.500d0
    lat_b 2.500d0
    lat_c 2.500d0
    alpha 60.0d0
    beta  60.0d0
    gamma 60.0d0
  end
 C  0.00000d0  0.00000d0  0.00000d0
 C  0.25000d0  0.25000d0  0.25000d0
end

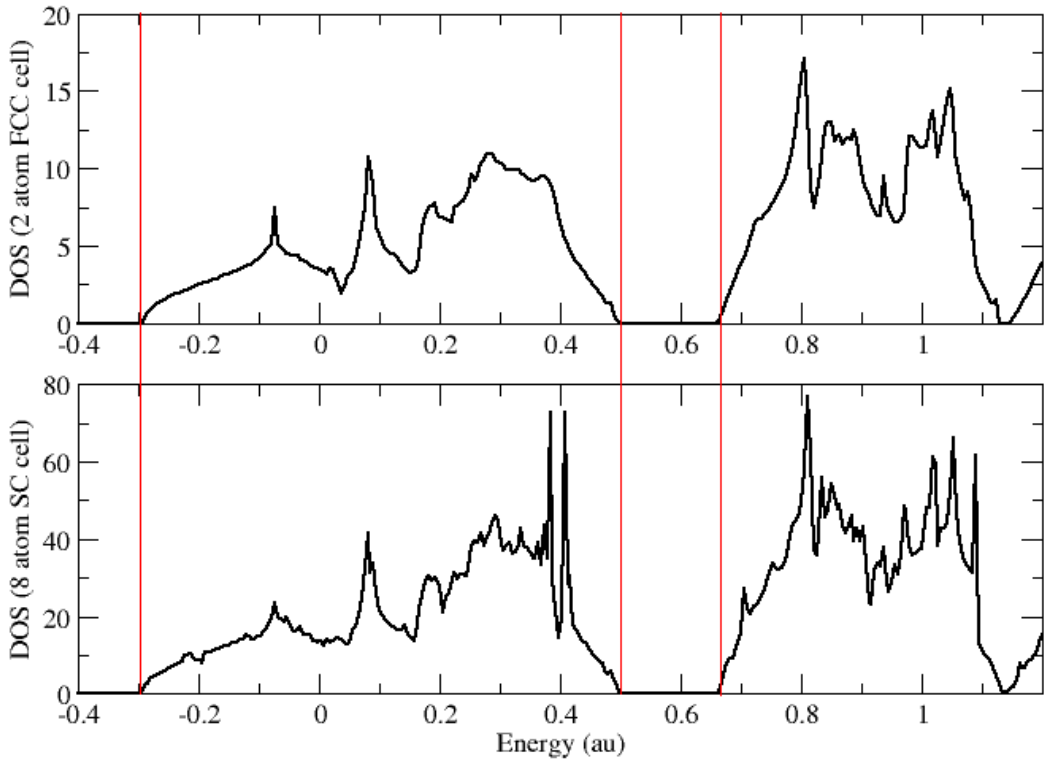
nwpw
  ewald_rcut 3.0
  ewald_ncut 8    #The default value of 1 needs to be increased
  lmbfgs
  xc pbe96

  monkhorst-pack 9 9 9
end

#need to run "task band energy" before "task band dos" can be run
task band energy

nwpw
   virtual 26                   #26 virtual orbitals included in the DOS calculation
   dos 0.002 700 -1.00000 2.0000  #alpha npoints emin emax,....,change default energy range and gridding. note alpha not used in task band dos calculations
   dos-grid 11 11 11
   mulliken                     # mulliken keyword used to turn on projected density of states
end
task band dos
```

This calculation outputs the data file in the permanent_directory. A plotting (e.g. gnuplot or xmgrace) can be used to display the density of states.



12.1.26.6 Calculate the Phonon Spectrum of Diamond

```
title "Diamond 2 atom fcc cell Brillouin sampling=9x9x9 M-P - Phonon spectra"
echo

permanent_dir ./perm
scratch_dir   ./scratch

start diamond-dos

memory 1950 mb

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 2.500d0
    lat_b 2.500d0
    lat_c 2.500d0
    alpha 60.0d0
    beta  60.0d0
    gamma 60.0d0
  end
 C  0.00000d0  0.00000d0  0.00000d0
 C  0.25000d0  0.25000d0  0.25000d0
end

nwpw
  ewald_rcut 3.0
  ewald_ncut 8    #The default value of 1 needs to be increased
  lmbfgs
  xc pbe96

  monkhorst-pack 9 9 9
end

task band energy
task band freq
```

12.1.27 NWPW Tutorial 6: optimizing the unit cell of nickel with fractional

occupation

(input:Media:Ni-band.nw output:Media:Ni-band.nwout) The following example demonstrates how to uses the BAND module to optimize the unit cell and geometry for FCC cell of Nickel metal

```
title "Ni FCC metal, monkhorst-pack=3x3x3, 5x5x5, and 7x7x7, fermi smearing, xc=pbe96"
echo

start Ni-band

memory 1900 mb

permanent_dir ./perm
scratch_dir   ./scratch

geometry units angstroms center noautosym noautoz print
 system crystal
   lat_a 3.5451d0
   lat_b 3.5451d0
   lat_c 3.5454d0
   alpha 90.0d0
   beta  90.0d0
   gamma 90.0d0
  end

Ni 0.000000   0.000000   0.000000
Ni 0.000000   0.500000   0.500000
Ni 0.500000   0.000000   0.500000
Ni 0.500000   0.500000   0.000000
end
set nwpw:cif_filename Ni-band
set nwpw:zero_forces .true.
set includestress    .true.

#turn on pseudopotential filtering
set nwpw:kbpp_ray .true.
set nwpw:kbpp_filter .true.

nwpw
   #fractional occupation
   smear fermi

   #scf option used with smear
   scf anderson outer_iterations 0 kerker 2.0

   ewald_ncut 8
   ewald_rcut 3.0
   xc pbe96
   monkhorst-pack 3 3 3
   np_dimensions -1 -1 4
end

#generate initial wavefunctions w/ low cutoff energy
nwpw
   loop 10 10
   cutoff 10.0
end
task band energy

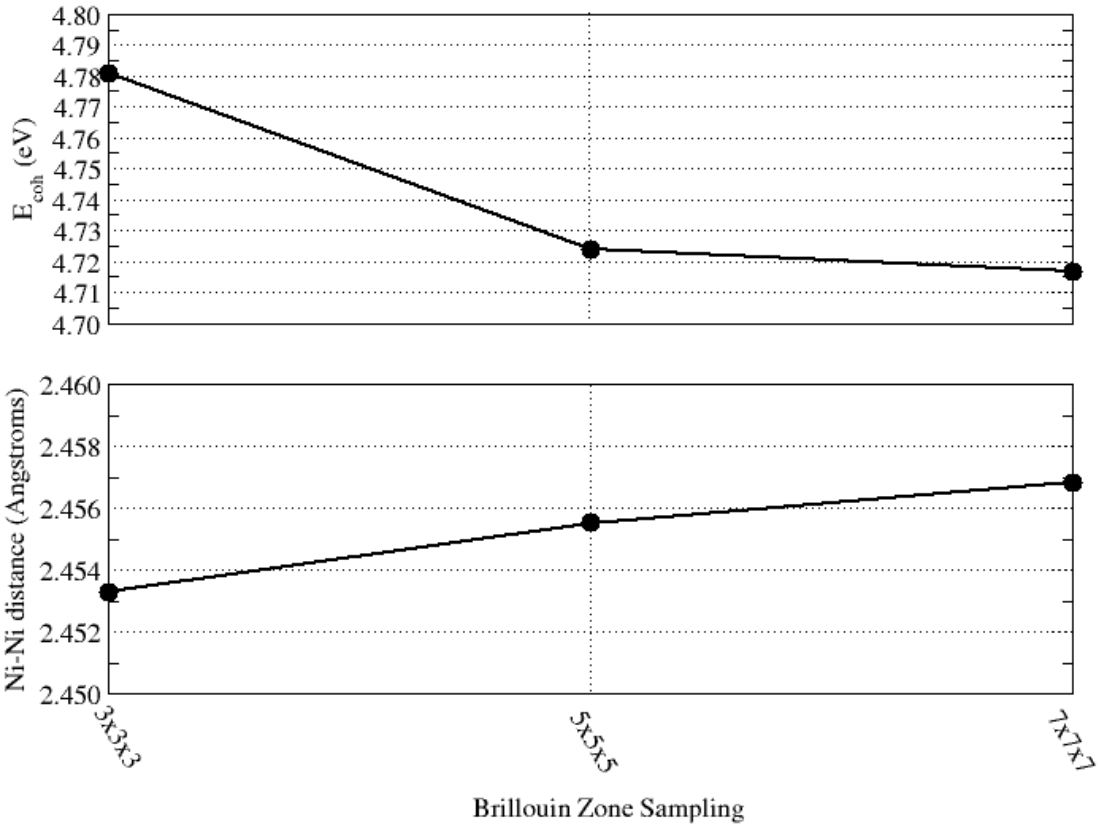
#increase cutoff energy and number of iterations
nwpw
   cutoff 50.0
   loop 10 100
end

#3x3x3 k-point mesh
nwpw
  monkhorst-pack 3 3 3
end
set nwpw:cif_filename nickel333.opt
driver; clear; maxiter 40; end; task band optimize ignore

#5x5x5 k-point mesh
nwpw
  monkhorst-pack 5 5 5
end
set nwpw:cif_filename nickel555.opt
driver; clear; maxiter 40; end; task band optimize ignore

#7x7x7 k-point mesh
nwpw
  monkhorst-pack 7 7 7
end
set nwpw:cif_filename nickel777.opt
driver; clear; maxiter 40; end; task band optimize ignore
```

The following figure shows a plot of the cohesive energy and Ni-Ni bond distance versus the Brillouin zone sampling. As can be seen in this figure the cohesive energy (w/o zero-point correction) and Ni-Ni bond distance agree very well with the experimental values of 4.44 eV (including zero-point correction) and 2.49 Angs.



12.1.28 NWPW Tutorial 7: Optimizing the unit cells with symmetry: Diamond with Fd-3m symmetry and Brucite with P-3m1 symmetry

(Diamond example, input:Media:diamond-symmetry.nw, output:Media:diamond-symmetry.nwout)

(Brucite example, input:Media:brucite-symmetry.nw, output:Media:brucite-symmetry.nwout)

The following example uses the BAND module to optimize the unit cell and geometry for a Diamond crystal with Fd-3m symmetry. The fractional coordinates, unit cell, and symmetry are defined in the geometry block.

```
title "Diamond 8 atom cubic cell generated using Fd-3m symmetry - geometry and unit cell optimization"
echo

memory 1500 mb

permanent_dir ./perm
scratch_dir   ./scratch

start diamond-symmetry


geometry nocenter noautosym noautoz print
 system crystal
   lat_a 3.58
   lat_b 3.58
   lat_c 3.58
   alpha 90.0
   beta  90.0
   gamma 90.0
 end
symmetry Fd-3m
C 0.0 0.0 0.0
end
set nwpw:cif_filename diamond-symmetry

#turn on pseudopotential filtering
set nwpw:kbpp_ray    .true.
set nwpw:kbpp_filter .true.

#***** setup the nwpw Band code - 3x3x3 k-point mesh ****
nwpw
  ewald_rcut 3.0
  ewald_ncut 8
  xc pbe96
  lmbfgs
  monkhorst-pack 3 3 3
  np_dimensions -1 -1 4
end

set includestress  .true.   # tell driver to optimize unit cell
set includelattice .true.   # tell driver to optimize with a,b,c,alpha,beta,gamma
task band optimize ignore
```

The optimized geometry will also contain the information about the symmetry being used

```
....
     ----------------------
     Optimization converged
     ----------------------


 Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
 ---- ---------------- -------- -------- -------- -------- -------- --------
@    7    -45.62102901 -4.1D-07  0.00010  0.00003  0.00019  0.00060    287.1
                              ok       ok       ok       ok


                    Geometry "geometry" -> "geometry"
                    ---------------------------------

 Output coordinates in angstroms (scale by  1.889725989 to convert to a.u.)

 No.       Tag         Charge          X              Y              Z
---- ---------------- ---------- -------------- -------------- --------------
   1 C                6.0000     0.00000000     0.00000000     0.00000000
   2 C                6.0000     0.00000000     1.76715074     1.76715074
   3 C                6.0000     1.76715074     1.76715074     0.00000000
   4 C                6.0000     1.76715074     0.00000000     1.76715074
   5 C                6.0000     2.65072611     0.88357537     2.65072611
   6 C                6.0000     0.88357537     0.88357537     0.88357537
   7 C                6.0000     0.88357537     2.65072611     2.65072611
   8 C                6.0000     2.65072611     2.65072611     0.88357537

     Lattice Parameters
     ------------------

     lattice vectors in angstroms (scale by  1.889725989 to convert to a.u.)

     a1=<   3.534   0.000   0.000 >
     a2=<   0.000   3.534   0.000 >
     a3=<   0.000   0.000   3.534 >
     a=       3.534 b=       3.534 c=       3.534
     alpha=  90.000 beta=  90.000 gamma=  90.000
     omega=    44.1
```

reciprocal lattice vectors in a.u.

b1=< 0.941 0.000 0.000 >
b2=< 0.000 0.941 0.000 >
b3=< 0.000 0.000 0.941 >

Atomic Mass

C 12.000000

Symmetry information

Group name Fd-3m
Group number 227
Group order 192
No. of unique centers 1
Setting number 1

Symmetry unique atoms

1

internuclear distances

center one	center two	atomic units	angstroms
5 C	4 C	2.89203	1.53040
6 C	1 C	2.89203	1.53040
6 C	2 C	2.89203	1.53040
6 C	3 C	2.89203	1.53040
6 C	4 C	2.89203	1.53040
7 C	2 C	2.89203	1.53040
8 C	3 C	2.89203	1.53040

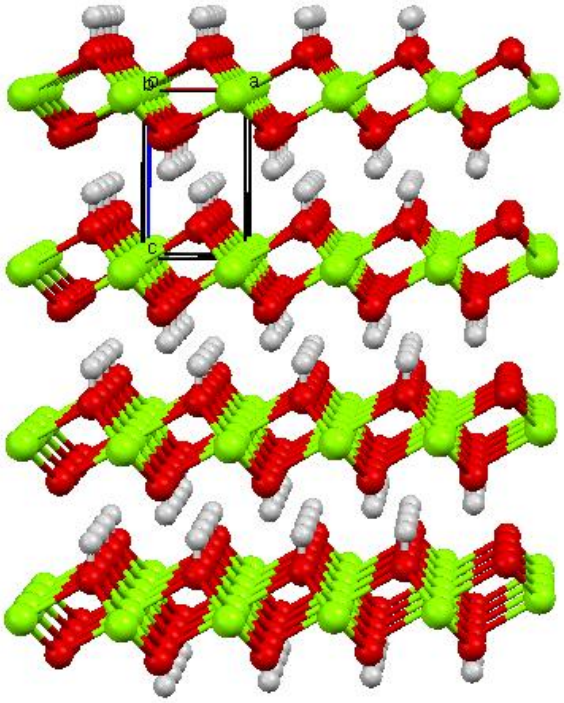
number of included internuclear distances: 7

internuclear angles

center 1	center 2	center 3	degrees
6 C	2 C	7 C	109.47
6 C	3 C	8 C	109.47
5 C	4 C	6 C	109.47
1 C	6 C	2 C	109.47
1 C	6 C	3 C	109.47
1 C	6 C	4 C	109.47
2 C	6 C	3 C	109.47
2 C	6 C	4 C	109.47
3 C	6 C	4 C	109.47

number of included internuclear angles: 9

The following example uses the BAND module to optimize the unit cell and geometry for a Brucite crystal ($Mg(OH)_2$ with P-3m1 symmetry.




```
title "brucite testing - using P-3m1 symmetry"
echo

memory 1500 mb

permanent_dir ./perm
scratch_dir      ./scratch

geometry nocenter noautosym noautoz print
 system crystal
   lat_a 3.14979
   lat_b 3.14979
   lat_c 4.7702
   alpha 90.0
   beta  90.0
   gamma 120.0
 end
symmetry P-3m1
Mg   0.00000   0.00000  0.00000
O   0.33333   0.66667   0.22030
H   0.33333   0.66667   0.41300
end
set nwpw:cif_filename brucite

#turn on pseudopotential filtering
set nwpw:kbpp_ray .true.
set nwpw:kbpp_filter .true.

#***** setup the nwpw gamma point code ****
nwpw
  ewald_rcut 3.0
  ewald_ncut 8
  xc pbe96
  lmbfgs
  monkhorst-pack 3 3 2
  #np_dimensions -1 -1 4
end

driver
  clear
  maxiter 31
end

set includestress  .true.          # tell driver to optimize unit cell
set includelattice .true.

task band optimize ignore
```

Optimizing Brucite, which is a soft layered material (2.5-3 Mohs scale), is more difficult to optimize than a hard material such as Diamond. For these types of materials using symmetry can often result in a faster optimization. For example, with symmetry the optimization converges within 20 to 30 geometry optimization steps,

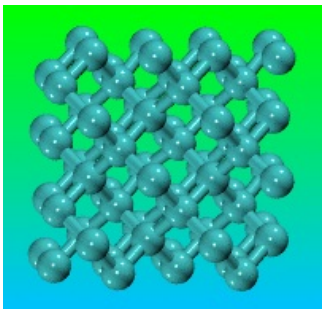
```
@ Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
@ ---- ---------------- -------- -------- -------- -------- -------- --------
@    0    -34.39207476  0.0D+00  0.24673  0.10223  0.00000  0.00000    172.7
@    1    -34.39340208 -1.3D-03  0.00872  0.00302  0.00198  0.00485    328.5
....
@   20    -34.39042736 -1.2D-05  0.00195  0.00083  0.00440  0.01964   3019.2
@   21    -34.39043463 -7.3D-06  0.00028  0.00011  0.00493  0.02042   3150.6
@   22    -34.39043484 -2.1D-07  0.00043  0.00014  0.00002  0.00008   3278.5
@   22    -34.39043484 -2.1D-07  0.00043  0.00014  0.00002  0.00008   3278.5
```

whereas, without symmetry the optimization may not be converged even at 100 geometry steps (input:Media:brucite-nosymmetry.nw, output:Media:brucite-nosymmetry.nwout).

```
@ Step       Energy      Delta E   Gmax     Grms     Xrms     Xmax   Walltime
@ ---- ---------------- -------- -------- -------- -------- -------- --------
@    0    -34.39207476  0.0D+00  0.24673  0.10250  0.00000  0.00000    18.4
@    1    -34.39340765 -1.3D-03  0.02963  0.00715  0.00202  0.00500    30.7
...
@   49    -34.39027641 -2.1D-06  0.01870  0.00646  0.00074  0.00202   595.7
@   50    -34.39027503  1.4D-06  0.01962  0.00669  0.00069  0.00197   608.4
...
@  100    -34.39034236 -3.8D-07  0.00380  0.00150  0.00036  0.00132  1155.3
@  101    -34.39034431 -1.9D-06  0.00305  0.00118  0.00012  0.00045  1166.8
@  102    -34.39034449 -1.8D-07  0.00370  0.00144  0.00006  0.00020  1177.9
...
```

12.1.29 NWPW Tutorial 8: NVT Metropolis Monte-Carlo Simulations

In this example the PSPW module is used to run an NVT simulation for a diamond crystal using the a Metropolis Monte-Carlo algorithm.



```
title "Metropolis NVT simulation of diamond - this input is used to put the system in equilibrium"
echo

start diamond-nvt

#permanent_dir ./perm
#scratch_dir   ./perm

#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
system crystal
  lat_a 3.56d0
  lat_b 3.56d0
  lat_c 3.56d0
  alpha 90.0d0
  beta  90.0d0
  gamma 90.0d0
end
C -0.50000d0 -0.50000d0 -0.50000d0
C  0.00000d0  0.00000d0 -0.50000d0
C  0.00000d0 -0.50000d0  0.00000d0
C -0.50000d0  0.00000d0  0.00000d0
C -0.25000d0 -0.25000d0 -0.25000d0
C  0.25000d0  0.25000d0 -0.25000d0
C  0.25000d0 -0.25000d0  0.25000d0
C -0.25000d0  0.25000d0  0.25000d0
end
set nwpw:cif_filename diamond_nvt_234

###### setup the nwpw gamma point code ######
set nwpw:kbpp_ray .true.
set nwpw:kbpp_filter .true.
set nwpw:frozen_lattice:thresh 999.0
nwpw
  lmbfgs
  ewald_rcut 3.0
  ewald_ncut 8
  xc pbe
end
task pspw energy

##### optimize the unit cell #####
set includestress  .true.  #this option tells driver to optimize the unit cell
set includelattice .true.  #this option tells driver to optimize cell using a,b,c,alpha,beta,gamma
driver
  clear
  maxiter 51
end
task pspw optimize ignore

###############################################################################################
###### setup Metropolis NVT code - input will change in a forthcoming release ######
###############################################################################################
set nwpw:mc_seed 234              # Seed for random number generator
set nwpw:mc_algorithm 1           # 1-NVT; 2-NPT
set nwpw:mc_aratio 0.234          # targeted acceptance ratio
set nwpw:mc_ddx 0.1               # parameter used to adjust geometry dispacement to have sampling with targeted acceptance
set nwpw:mc_temperature 300.0     # Temperature in K
set nwpw:mc_step_size   0.250     # initial geometry displacement step size

nwpw
  mc_steps 10 100  #total number of iterations = 10*100, number of iterations between step size adjustments = 10
  cpmd_properties on
end
task pspw Metropolis
```

12.1.30 NWPW Tutorial 9: NPT Metropolis Monte-Carlo Simulations

In this example the PSPW module is used to run an NPT simulation for a diamond crystal using the a Metropolis Monte-Carlo algorithm.

(input:Media:diamond-metropolis.nw, output:Media:diamond-metropolis.nwout.gz, datafiles:Media:diamond-metropolis.emotion.gz, Media:diamond-metropolis.ion_motion.gz, Media:diamond-metropolis.xyz.gz, Media:diamond_metropolis_1234.cif.gz)

```
title "Metropolis NPT simulation of diamond - this input is used to put the system in equilibrium"
echo

start diamond-metropolis

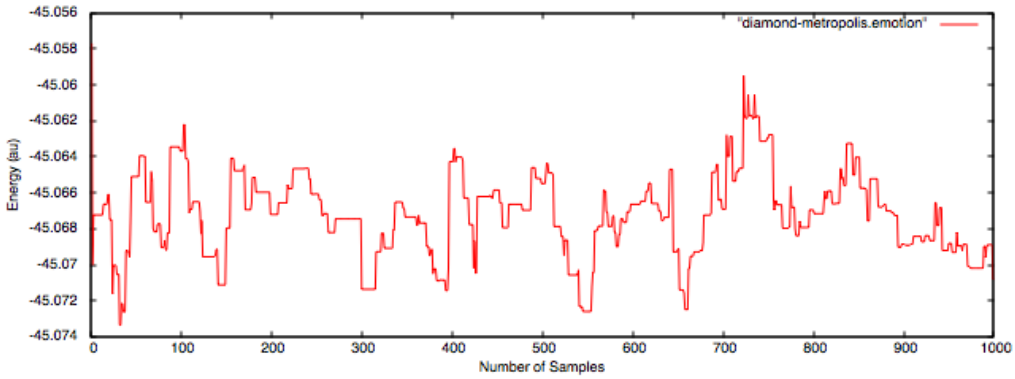
#permanent_dir ./perm
#scratch_dir   ./perm

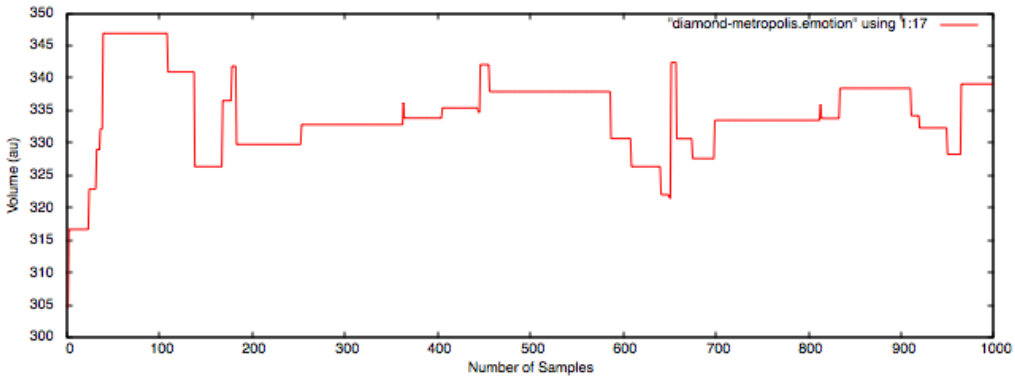
#**** Enter the geometry using fractional coordinates ****
geometry center noautosym noautoz print
  system crystal
    lat_a 3.56d0
    lat_b 3.56d0
    lat_c 3.56d0
    alpha 90.0d0
    beta  90.0d0
    gamma 90.0d0
  end
 C -0.50000d0 -0.50000d0 -0.50000d0
 C  0.00000d0  0.00000d0 -0.50000d0
 C  0.00000d0 -0.50000d0  0.00000d0
 C -0.50000d0  0.00000d0  0.00000d0
 C -0.25000d0 -0.25000d0 -0.25000d0
 C  0.25000d0  0.25000d0 -0.25000d0
 C  0.25000d0 -0.25000d0  0.25000d0
 C -0.25000d0  0.25000d0  0.25000d0
end
set nwpw:cif_filename pspw_metropolis

###### setup the nwpw gamma point code ######
set nwpw:kbpp_ray .true.
set nwpw:kbpp_filter .true.
set nwpw:frozen_lattice:thresh 999.0
nwpw
   lmbfgs
   ewald_rcut 3.0
   ewald_ncut 8
   xc pbe
end
task pspw energy


################################################################################
###### setup Metropolis NPT code - input will change in a forthcoming release ######
################################################################################
set nwpw:mc_seed 1234              # Seed for random number generator
set nwpw:mc_algorithm 2            # 1-NVT; 2-NPT
set nwpw:mc_aratio 0.234           # targeted acceptance ratio
set nwpw:mc_ddx 0.1                # parameter used to adjust geometry dispacement to have sampling with targeted acceptance
set nwpw:mc_ddv 0.1                # parameter used to adjust volume change to have sampling with targeted acceptance
set nwpw:mc_temperature 300.0      # Temperature in K
set nwpw:mc_step_size   0.250      # geometry displacement step size
set nwpw:mc_volume_step 0.130      # volume displacement step size

nwpw
   bo_steps 10 100  #total number of iterations = 10*100, number of iterations between step size adjustments = 10
end
task pspw Metropolis
```





(inputs:Media:diamond-metropolis-sampling.nw.tgz)

(python analysis program:Media:makehistogram.gz)

```
[WE27972:~/Projects/NWChem/Metropolis] bylaska% makehistogram -t 300 -c 2 1235/diamond-metropolis-1235.emotion 1236/diamond-metropolis-1236.emotion
1237/diamond-metropolis-1237.emotion 1238/diamond-metropolis-1238.emotion 1239/diamond-metropolis-1239.emotion 1240/diamond-metropolis-1240.emotion
1241/diamond-metropolis-1241.emotion 1242/diamond-metropolis-1242.emotion 1243/diamond-metropolis-1243.emotion 1244/diamond-metropolis-1244.emotion
1245/diamond-metropolis-1245.emotion 1246/diamond-metropolis-1246.emotion 1248/diamond-metropolis-1248.emotion 1249/diamond-metropolis-1249.emotion
makehistogram Program
len(args)= 14

unitconversion  = 1.0
temperature (K) = 300
RT (au)         = 0.000949482834326  ( 0.5958  kcal/mol)

data columns -1 = [1]
data rows (n)   = 52000

delta (au)      = 0.01
xmin-delta (au)  = -45.08080365
xmax+delta (au)  = -45.05079515

data averaging:
- xbar       (au)          = -45.0668093497
- S2_{n-1}   (au)          = 1.08378343101e-05
- <exp((x-xmin)/RT)> (au)   = 5293374903.39
- <exp((x-xbar)/RT)> (au)   = 2102.44405413
- Free energy               = -45.0595449934
- Free energy1              = -45.0595449934

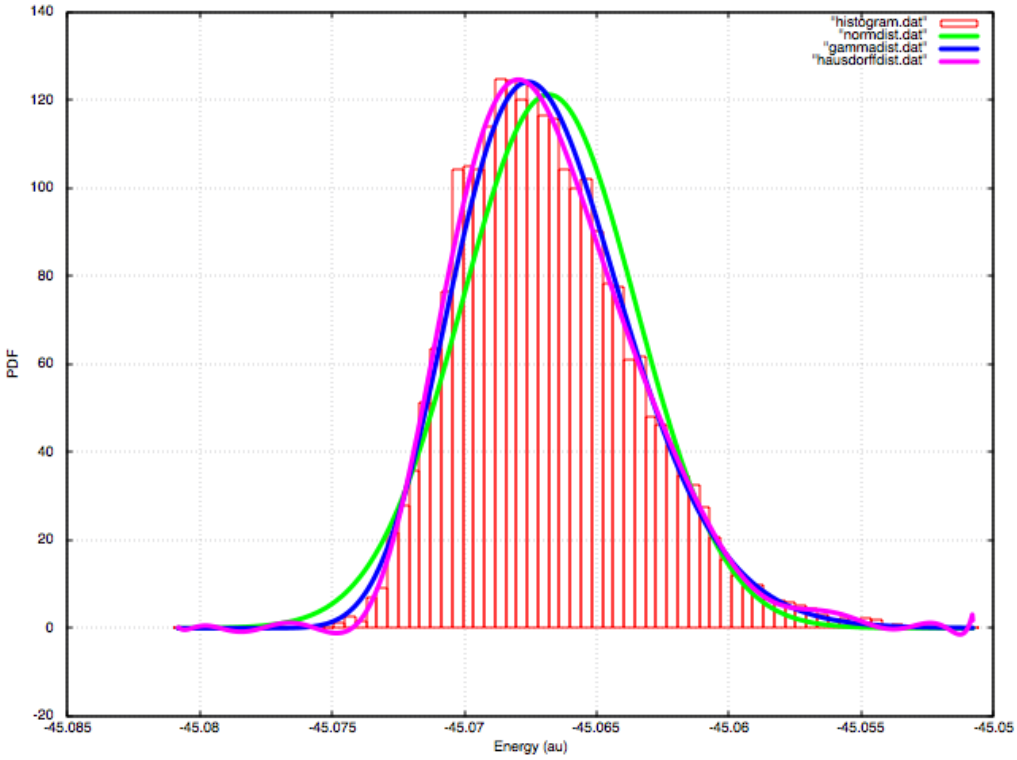
histogram distribution parameters:
- number of bins (Rice k) = 75
- bin width               = 0.00040552027027
- norm                    = 1.0
- xbar (au)               = -45.0668107987  (error= -1.44908364064e-06 )
- S2_{n-1} (au)           = 1.0858459744e-05  (error= 2.06254339582e-08 )
- <exp((x-xmin)/RT)> (au) = 5184600342.01  (error= -108774561.378 )
- <exp((x-xbar)/RT)> (au) = 2062.38570923  (error= -40.0583449011 )
- Free energy             = -45.0595647078  (error= -1.9714360235e-05 )
- Free energy1            = -45.0595647078  (error= -1.9714360235e-05 )
- histogram plot file     = histogram.dat

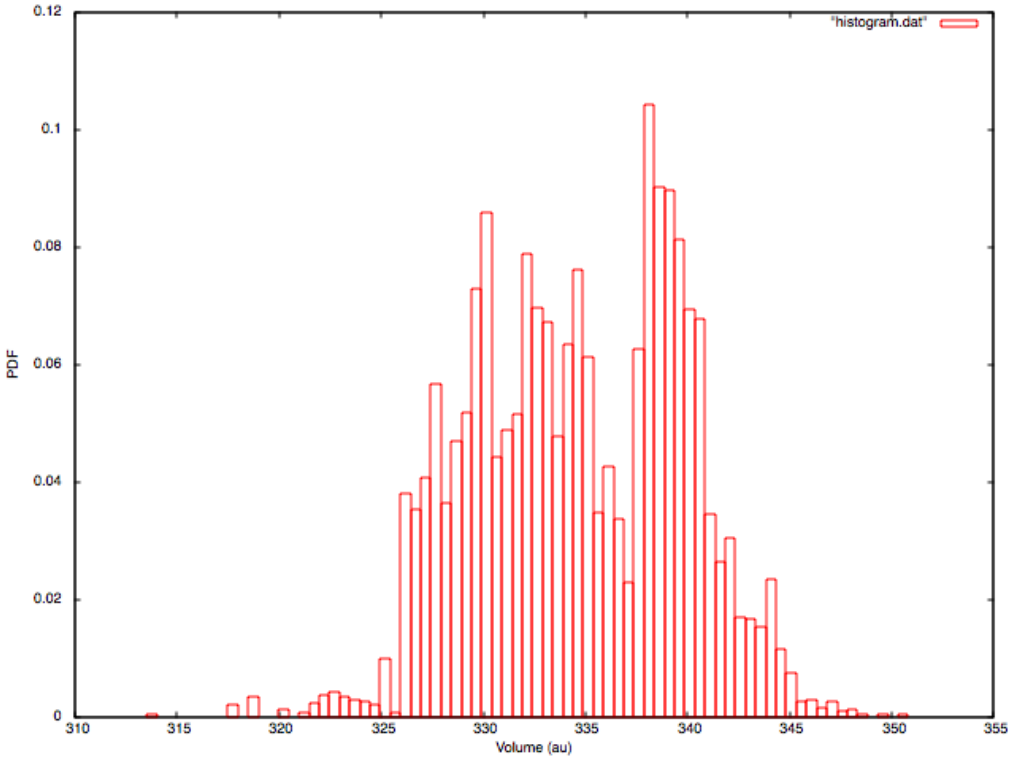
normal distribution parameters:
- average x (input xbar)                = -45.0668093497
- unbiased sample variance (input S2_(n-1))= 1.08378343101e-05
- xbar-xmin                             = 0.0139943003357
- norm                    = 0.99998877936
- xbar (au)               = -45.0663035243  (error= 0.000505825397738 )
- S2_{n-1} (au)           = 1.1091077321e-05  (error= 2.53243010936e-07 )
- <exp((x-xmin)/RT)> (au) = 943482808.939  (error= -4349892094.45 )
- <exp((x-xbar)/RT)> (au) = 219.968603653  (error= -1882.47545048 )
- Free energy             = -45.061182503  (error= -0.00163750957643 )
- Free energy1            = -45.061182503  (error= -0.00163750957643 )
```

- normal distribution plot file = normdist.dat
- number data points = 1500

gamma distribution parameters:
- alpha0= 18.0700715921
- beta0 = 1291.24508969
- xmin + alpha0/beta0 = -45.0668093497
- alpha = 18.5003178357
- beta = 1321.98948086
- xmin + alpha/beta = -45.0668093497
- norm = 0.999923464137 0.99993569948
- xbar (au) = -45.0633614482 -45.0639126423 (error= 0.00344790150088 0.00289670733491)
- S2_{n-1} (au) = 2.27110055327e-05 1.89632753897e-05 (error= 1.18731712226e-05 8.12544107961e-06)
- <exp((x-xmin)/RT)> (au) = 7932775654.26 7060892836.07 (error= 2639400750.87 1767517932.68)
- <exp((x-xbar)/RT)> (au) = 83.43400035 132.707151194 (error= -2019.01005378 -1969.73690294)
- Free energy = -45.059160883 -45.0592714327 (error= 0.000384110406969 0.000273560709338)
- Free energy1 = -45.059160883 -45.0592714327 (error= 0.000384110406969 0.000273560709338)
- gamma distribution plot file = gammadist.dat
- number data points = 1500

Hausdorff distribution parameters:
- xmin = -45.08080365
- xmax = -45.05079515
- number moments = 15
  -- < x^0 > = 1.000000000000000
  -- < x^1 > = 0.466344546904007
  -- < x^2 > = 0.229512222180349
  -- < x^3 > = 0.119040323347820
  -- < x^4 > = 0.064946164109284
  -- < x^5 > = 0.037186896798964
  -- < x^6 > = 0.022287980659815
  -- < x^7 > = 0.013942929105868
  -- < x^8 > = 0.009076370636747
  -- < x^9 > = 0.006128509645342
  -- < x^10 > = 0.004278147917961
  -- < x^11 > = 0.003077410986590
  -- < x^12 > = 0.002273768533280
  -- < x^13 > = 0.001720304299285
  -- < x^14 > = 0.001328990330385
- norm = 1.0000000003
- xbar (au) = -45.066809363 (error= -1.33426993898e-08)
- S2_{n-1} (au) = 1.08376258908e-05 (error= -2.08419282206e-10)
- <exp((x-xmin)/RT)> (au) = 5423305875.35 (error= 129930971.958)
- <exp((x-xbar)/RT)> (au) = 2154.08083332 (error= 51.6367791881)
- Free energy = -45.0595219689 (error= 2.30245307122e-05)
- Free energy1 = -45.0595219689 (error= 2.30245307122e-05)
- Hausdorff moment history file = moment_hist.dat
- Hausdorff distribution plot file = hausdorff.dat
- number data points = 1500





12.1.31 NWPW Tutorial 9: Free Energy Simulations

A description of using the WHAM method for generating free energy of the gas-phase dissociation reaction CH\(_3\)Cl \(\rightarrow\) CH\(_3\)+Cl can be found in the attached pdf (nwchem-new-pmf.pdf)

12.1.32 PAW Tutorial

12.1.32.1 Optimizing a water molecule

The following input deck performs for a water molecule a PSPW energy calculation followed by a PAW energy calculation and a PAW geometry optimization calculation. The default unit cell parameters are used (SC=20.0, ngrid 32 32 32). In this simulation, the first PAW run optimizes the wavefunction and the second PAW run optimizes the wavefunction and geometry in tandem.

```
title "paw steepest descent test"
start paw_test
charge 0
geometry units au nocenter noautoz noautosym
O      0.00000    0.00000    0.01390
H     -1.49490    0.00000   -1.18710
H      1.49490    0.00000   -1.18710
end
nwpw
  time_step 15.8
  ewald_rcut 1.50
  tolerances 1.0d-8 1.0d-8
end
set nwpw:lcao_iterations 1
set nwpw:minimizer 2
task pspw energy
task paw energy
nwpw
  time_step 5.8
  geometry_optimize
  ewald_rcut 1.50
  tolerances 1.0d-7 1.0d-7 1.0d-4
end
task paw steepest_descent
task paw optimize
```

12.1.32.2 Optimizing a unit cell and geometry for Silicon-Carbide

The following example demonstrates how to uses the PAW module to optimize the unit cell and geometry for a silicon-carbide crystal.

```
title "SiC 8 atom cubic cell - geometry and unit cell optimization"
start SiC
#**** Enter the geometry using fractional coordinates ****
geometry units au center noautosym noautoz print
 system crystal
   lat_a 8.277d0
   lat_b 8.277d0
   lat_c 8.277d0
   alpha 90.0d0
   beta  90.0d0
   gamma 90.0d0
 end
Si    -0.50000d0  -0.50000d0  -0.50000d0
Si     0.00000d0   0.00000d0  -0.50000d0
Si     0.00000d0  -0.50000d0   0.00000d0
Si    -0.50000d0   0.00000d0   0.00000d0
C     -0.25000d0  -0.25000d0  -0.25000d0
C      0.25000d0   0.25000d0  -0.25000d0
C      0.25000d0  -0.25000d0   0.25000d0
C     -0.25000d0   0.25000d0   0.25000d0
end
#***** setup the nwpw gamma point code ****
nwpw
  simulation_cell
    ngrid 16 16 16
  end
  ewald_ncut 8
end
set nwpw:minimizer 2
set nwpw:psi_nolattice .true.  # turns of unit cell checking for wavefunctions
driver
  clear
  maxiter 40
end
set includestress .true.         # this option tells driver to optimize the unit cell
set nwpw:stress_numerical .true. #currently only numerical stresses implemented in paw
task paw optimize
```

12.1.32.3 Running a Car-Parrinello Simulation

In this section we show how use the PAW module to perform a Car-Parrinello molecular dynamic simulation for a C_2 molecule at the LDA level. Before running a PAW Car-Parrinello simulation the system should be on the Born-Oppenheimer surface, i.e. the one-electron orbitals should be minimized with respect to the total energy (i.e. task pspw energy). The input needed is basically the same as for optimizing the geometry of a C_2 molecule at the LDA level,except that and additional Car-Parrinello sub-block is added.

In the following example we show the input needed to run a Car-Parrinello simulation for a C_2 molecule at the LDA level. In this example, default pseudopotentials from the pseudopotential library are used for C, the boundary condition is free-space, the exchange correlation functional is LDA, The boundary condition is free-space, and the simulation cell cell is aperiodic and cubic with a side length of 10.0 Angstroms and has 40 grid points in each direction (cutoff energy is 44 Ry). The time step and fake mass for the Car-Parrinello run are specified to be 5.0 au and 600.0 au, respectively.

```
start c2_paw_lda_md
title "C2 restricted singlet dimer, LDA/44Ry - constant energy Car-Parrinello simulation"
geometry
 C    -0.62 0.0 0.0
 C     0.62 0.0 0.0
end
pspw
  simulation_cell units angstroms
     boundary_conditions aperiodic
     lattice
       lat_a 10.00d0
       lat_b 10.00d0
       lat_c 10.00d0
     end
     ngrid 40 40 40
  end
  Car-Parrinello
    fake_mass 600.0
    time_step 5.0
    loop 10 10
  end
end
set nwpw:minimizer 2
task paw energy
task paw Car-Parrinello
```

12.1.33 NWPW Capabilities and Limitations

- Hybrid Functionals (e.g. PBE0, LDA-SIC) only work in PSPW.
- Wannier orbital task only works in PSPW.
- AIMD/MM simulation only works with PSPW.

12.1.34 Questions and Difficulties

Questions and encountered problems should be reported to the NWChem Community Forum or to Eric J. Bylaska, Eric.Bylaska@pnl.gov

12.2 Gaussian Basis AIMD

12.2.1 Overview

This module performs adiabatic ab initio molecular dynamics on finite systems. The nuclei are integrated using the velocity-Verlet algorithm, and the electronic potential can be provided by any of the Gaussian basis set based methods in NWChem, e.g. DFT, TDDFT, TCE, MP2, SCF, MCSCF, etc. If analytic gradients are not available for the selected level of theory, numerical gradients will automatically be used. Initial velocities are randomly selected from the Maxwell-Boltzmann distribution at the specified temperature, unless a restart file (`.qmdrst`) is present. If a restart file is present, the trajectory information will be read from that file and the trajectory will resume from that point.

For computational details and a case study using the module, please refer to the 2016 paper by Fischer[1].

```
QMD
  [dt_nucl <double default 10.0>]
  [nstep_nucl <integer default 1000>]
  [targ_temp <double default 298.15>]
  [thermostat <string default none> <thermostat parameters>]
  [rand_seed <integer default new one generated for each run>]
  [com_step <integer default 100>]
  [print_xyz <integer default 1>]
  [linear]
  [property <integer default 1>]
  [tddft <integer default 1>]
  [namd ]
END
```

The module is called as:

```
task <level of theory> qmd
```

where is any Gaussian basis set method in NWChem

12.2.2 QMD Keywords

12.2.2.1 DT_NUCL: Nuclear time step

This specifies the nuclear time step in atomic units (1 a.u. = 0.02419 fs). Default: 10.0 a.u.

12.2.2.2 NSTEP_NUCL: Simulation steps

This specifies the number of steps to take in the nuclear dynamics.
Default: 1000

12.2.2.3 TARG_TEMP: Temperature of the system

This specifies the temperature to use with the thermostat. Also it is used in generating initial velocities from the Maxwell-Boltzmann distribution.
Default: 298.15 K

12.2.2.4 THERMOSTAT: Thermostat for controling temperature of the simulation

This specifies the thermostat to use for regulating the temperature of the nuclei. Possible options are:

- **none**

No thermostat is used, i.e. an NVE ensemble is simulated. Default

- **svr** <double default 1000.0>

Stochastic velocity rescaling thermostat of Bussi, Donadio, and Parrinello[2].
Number sets the relaxation parameter of the thermostat

- **langevin** <double default 0.1>

Langevin dynamics, implementation according to Bussi and Parrinello[3].
The optional input parameter sets the value of the friction

- **berendsen** <double default 1000.0>

Berendsen thermostat, the optional input parameter sets the relaxation parameter of the thermostat

- **rescale**

Velocity rescaling, i.e. isokinetic ensemble

- **nose-hoover** <integer default 3>

Nosé–Hoover thermostat (only available in release 7.2.0 and later). The optional input parameter defines the number of oscillators.

12.2.2.5 RAND_SEED: Seed for the random number generator

rand_seed specifies the seed for initializing the random number generator. If not given, a unique random seed will be generated. Even without a thermostat, this will influence the initial velocities.

12.2.2.6 COM_STEP: How often center-of-mass translations and rotations are removed

`com_step` specifies that center-of-mass translations and rotations will be removed every `com_step` steps. Default 10 COM translations and rotations are removed on startup (either randomized initial velocities or those read from the restart file).

12.2.2.7 PRINT_XYZ: How often to print trajectory information to xyz file

`print_xyz` specifies how often the trajectory information (coordinates, velocities, total energy, step number, dipole (if available)) is written to the xyz file. The units for the coordinates and velocities in the xyz file are Angstrom and Angstrom/fs, respectively.
For example, `print_xyz 5` will write the xyz trajectory file every 5 steps.
Default: 1

12.2.2.8 LINEAR: Flag for linear molecules

If the `linear` keyword is present, the code assumes the molecule is linear.

12.2.2.9 PROPERTY: How often to calculate molecular properties as part of the MD simulation

If the `property` keyword present, the code will look for the property block and calculate the requested properties.
For example, `property 5` will calculate properties on the current geometry every 5 steps.
Default: 0 (e.g properties are not computed)

12.2.2.10 TDDFT: How often to perform TDDFT calculation as part of the MD simulation

If the `tddft` keyword is present, the code will look for the tddft block and calculate the absorption spectrum.
For example, `tddft 5` will perform tddft calculations on the current geometry every 5 steps.
Default: 0 (e.g tddft is not run)

12.2.3 Sample input files

12.2.3.1 Ground state Molecular Dynamics

The following is a sample input for a ground state MD simulation. The simulation is 200 steps long with a 10 a.u. time step, using the stochastic velocity rescaling thermostat with a relaxation parameter of 100 a.u. and a target temperature of 200 K. Center-of-mass rotations and translations will be removed every 10 steps and trajectory information will be output to the xyz file every 5 steps.

```
start qmd_dft_h2o_svr
echo
print low
geometry noautosym noautoz
  O   0.00000000    -0.01681748     0.11334792
  H   0.00000000     0.81325914    -0.34310308
  H   0.00000000    -0.67863597    -0.56441201
end
basis
  * library 6-31G*
end
dft
  xc pbe0
end
qmd
  nstep_nucl  200
  dt_nucl     10.0
  targ_temp   200.0
  com_step    10
  thermostat  svr 100.0
  print_xyz   5
end
task dft qmd
```

12.2.3.2 Excited state Molecular Dynamics

The following is a sample input for an excited state MD simulation on the first excited state. The simulation is 200 steps long with a 10 a.u. time step, run in the microcanonical ensemble. Center-of-mass rotations and translations will be removed every 10 steps and trajectory information will be output to the xyz file every 5 steps.

```
start qmd_tddft_h2o_svr
echo
print low
geometry noautosym noautoz
  O   0.00000000    -0.01681748     0.11334792
  H   0.00000000     0.81325914    -0.34310308
  H   0.00000000    -0.67863597    -0.56441201
end
basis
  * library 6-31G*
end
dft
  xc pbe0
end
tddft
  nroots 5
  notriplet
  target 1
  civecs
  grad
    root 1
  end
end
qmd
  nstep_nucl  200
  dt_nucl     10.0
  com_step    10
  thermostat  none
  print_xyz   5
end
task tddft qmd
```

12.2.3.3 Property calculation in a Molecular Dynamics simulation

Thefollowing is a sample input for an MD simulation that compute polarizability by means of theSOS method at each time step.

```
start qmd_prop_h2o_svr
echo
print low
geometry noautosym noautoz
  O   0.00000000    -0.01681748     0.11334792
  H   0.00000000     0.81325914    -0.34310308
  H   0.00000000    -0.67863597    -0.56441201
end
basis
  * library 6-31G*
end
dft
  xc pbe0
end

qmd
  nstep_nucl  200
  dt_nucl     10.0
  com_step    10
  thermostat  none
  print_xyz   5
  property 1
end

property
 polfromsos
end
task tddft qmd
```

Additional sample inputs can be found in $NWCHEM_TOP/QA/tests/qmd_* (e.g. https://github.com/nwchemgit/nwchem/tree/master/QA/tests/qmd_dft_h2o_berendsen_props)

12.2.4 Processing the output of a QMD run

The xyz file produced by the QMD module contains the velocities (given in Angstrom/fs), in addition to the coordinates (given in Angstrom). The comment lines also contain the time step, total energy (atomic units), and dipole moment (atomic units). In the directory $NWCHEM_TOP/contrib/qmd_tools, the code qmd_analysis.f90 will used the xyz trajectory as input to calculate the IR spectrum and vibrational density of states from Fourier transforms of the dipole and atomic momenta autocorrelation functions, respectively. The code needs to be linked to a LAPACK library when compiled; the Makefile in the directory will compile the code with the LAPACK routines included with the NWChem source.

Here we compute the IR spectrum and the element-wise breakdown of the vibrational density of states for silicon tetrachloride ($SiCl_4$). The following input file was used.

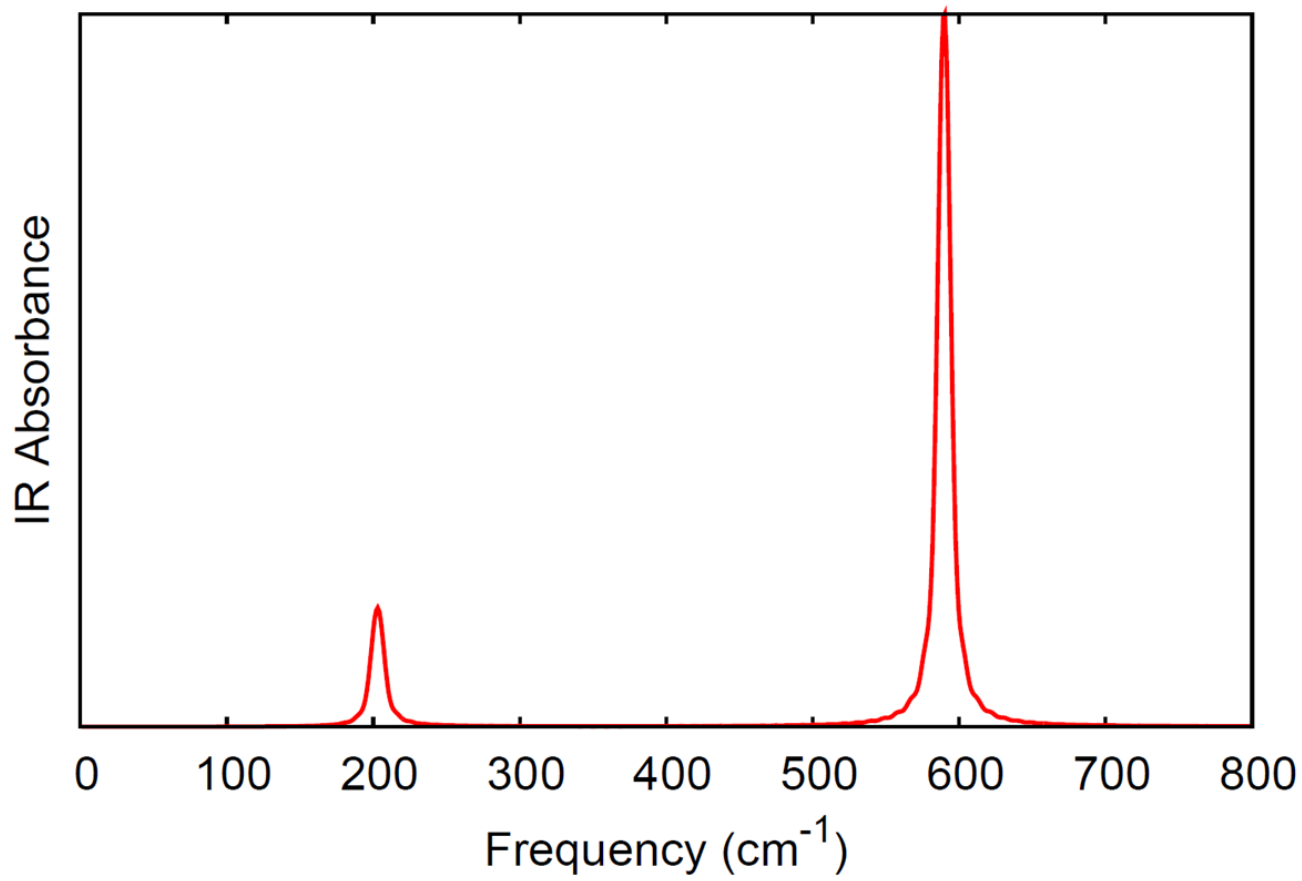
```
start SiCl4
echo
print low
geometry noautosym noautoz
  Si          -0.00007905     0.00044148     0.00000001
  Cl           0.71289590     1.00767685     1.74385011
  Cl          -2.13658008    -0.00149375    -0.00000001
  Cl           0.71086735    -2.01430142    -0.00000001
  Cl           0.71289588     1.00767684    -1.74385011
end
basis
  * library 6-31G
end
dft
  xc hfexch 1.0
end
qmd
  nstep_nucl  20000
  dt_nucl     10.0
  targ_temp   20.0
  com_step    10
  rand_seed   12345
  thermostat  none
end
task dft qmd
```

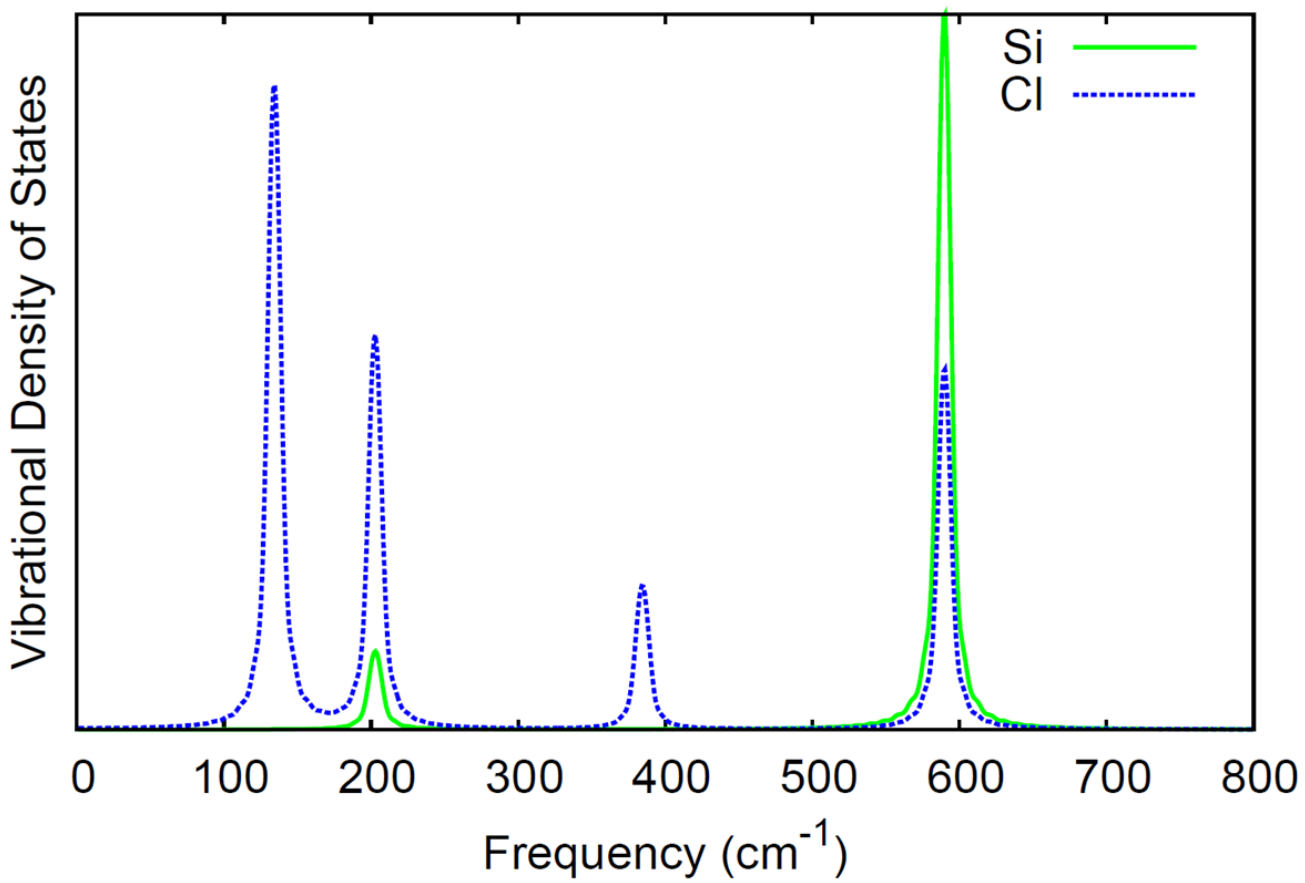
The IR spectrum and vibrational density of states were generated from the qmd_analysis code with the following command.

```
./qmd_analysis -xyz SiCl4.xyz -steps 15000 -skip 5000 -ts 10.0 -temp 20.0 -smax 800 -width 10.0
```

where we have skipped the first 5000 steps from the simulation and only used the data from the last 15000 steps to compute the spectra. The time step is given as 10 a.u. since that was the time step in the simulation and we output the trajectory information every step. The temperature was set to 20 K (for analysis, this is only used in the calculation of the quantum correction factor for the autocorrelation function of the dipole moment). The option smax sets the maximum of the spectral window that is output to 800 wave numbers. The width option sets the full-width at half-maximum of the peaks in the resulting spectra.

The computed IR spectrum and vibrational density of states are shown here.





12.2.5 NAMD: Non-adiabatic Excited Stated Molecular Dynamics

For details of the NAMD implementation, please refer to the 2020 paper by Song[4].

```
[namd]
  [init_state <integer default 2>]
  [nstates <integer default 2>]
  [dt_elec <double default 0.01>]
  [deco <logical default .false.]
  [tdks <integer default 1>]
[end]
```

In the `namd` sub-block within the `qmd` block, please note:

- The number of roots requested in the tddft block must be at least nstates-1.
- The nuclear time step (dt_nucl) must be an integer multiple of the electronic time step (`mod(dt_nucl,dt_elec)=0`).

12.2.5.1 DECO: Decoherence flag

The `deco` flag applies the EDC electronic decoherence correction described in the papey bt Granucci and Persico[5].
The default value is `.false.`, i.e. no decoherence correction is applied.

12.2.5.2 DT_ELEC: Electronic dynamics time step

The keyword `dt_elec` sets the electronic time step in atomic units.

12.2.5.3 N_STATES: Number of states

The keyword `nstates` sets the number of electronic states to include in the calculation, i.e. the number of states for use with

Eq. 5 of the 2020 Song paper.

12.2.5.4 INIT_STATE: Initial state

The keyword `init_state` sets the initial electronic state to be occupied; the numbering for this keyword and the output that reports the currently occupied state runs from 0 (ground state) to `nstates-1`. So if you want to start a calculation in the first excited state, you would set `init_state` to 1.

12.2.5.5 TDKS: Time-Dependent Kohn-Sham

The keyword `tdks` will use Time-Dependent Kohn-Sham instead of the defaultTamm-Dancoff approximation.
The keyword requires the keyword `odft` in the `dft` input block to work.
It can have two values:

- `1` (default) selects the alpha spin channel
- `2` selects the beta spin channel

12.2.5.6 NAMD Input Example

Example input for fewest-switches surface-hopping (FSSH) approach.

```
geometry noautosym nocenter
O 0.0000  0.0000  0.1197
H 0.0000  0.7615 -0.4790
H 0.0000 -0.7615 -0.4790
end
basis
* library 6-31G*
end

dft
  xc b3lyp
end

tddft
  nroots 10
  notriplet
  cis
  civecs
  grad
    root 1
  end
end

qmd
  nstep_nucl 50
  dt_nucl 0.5
  targ_temp 300.0
  thermostat svr 500
  namd
    nstates 5
    init_state 3
    dt_elec 0.1
    deco .true.
  end
end
task tddft qmd
```

12.2.6 References

1. Fischer, S. A.; Ueltschi, T. W.; El-Khoury, P. Z.; Mifflin, A. L.; Hess, W. P.; Wang, H.-F.; Cramer, C. J.; Govind, N. Infrared and Raman Spectroscopy from Ab Initio Molecular Dynamics and Static Normal Mode Analysis: The C-H Region of DMSO as a Case Study. *The Journal of Physical Chemistry B* **2015**, *120* (8), 1429–1436. https://doi.org/10.1021/acs.jpcb.5b03323
2. Bussi, G.; Donadio, D.; Parrinello, M. Canonical Sampling Through Velocity Rescaling. *The Journal of Chemical Physics* **2007**, *126* (1), 014101. https://doi.org/10.1063/1.2408420
3. Bussi, G.; Parrinello, M. Accurate Sampling Using Langevin Dynamics. *Physical Review E* **2007**, *75* (5), 056707.

https://doi.org/10.1103/PhysRevE.75.056707

4. Song, H.; Fischer, S. A.; Zhang, Y.; Cramer, C. J.; Mukamel, S.; Govind, N.; Tretiak, S. First Principles Nonadiabatic Excited-State Molecular Dynamics in NWChem. *Journal of Chemical Theory and Computation* **2020**, *16* (10), 6418–6427. https://doi.org/10.1021/acs.jctc.0c00295
5. Granucci, G.; Persico, M. Critical Appraisal of the Fewest Switches Algorithm for Surface Hopping. *The Journal of Chemical Physics* **2007**, *126* (13), 134114. https://doi.org/10.1063/1.2715585

13 Hybrid Approaches

13.1 Solvation Models

13.1.1 Overview

Two solvation models are available in NWChem: COSMO and SMD. Since some of the COSMO parameters are used for SMD, we suggest to read the COSMO section before the SMD one.

13.1.2 COSMO

13.1.2.1 Overview

COSMO is the continuum solvation 'COnductor-like Screening MOdel' of A. Klamt and G. Schüürmann to describe dielectric screening effects in solvents[1]. This model has been enhanced by D.M. York and M. Karplus[2] to create a smooth potential energy surface. The latter facilitates geometry optimization and dynamics and the implementation has been adapted to take advantage of those ideas.

The NWChem COSMO module implements algorithm for calculation of the energy for the following methods:

1. Restricted Hartree-Fock (RHF),
2. Restricted open-shell Hartree-Fock (ROHF),
3. Restricted Kohn-Sham DFT (DFT),
4. Unrestricted Kohn-Sham DFT (ODFT),

by determining the solvent reaction field self-consistently with the solute charge distribution from the respective methods. Note that COSMO for unrestricted Hartree-Fock (UHF) method can also be performed by invoking the DFT module with appropriate keywords.

Correlation energy of solvent molecules may also be evaluated at

1. MP2,
2. CCSD,
3. CCSD+T(CCSD),
4. CCSD(T),

levels of theory. It is cautioned, however, that these correlated COSMO calculations determine the solvent reaction field using the HF charge distribution of the solute rather than the charge distribution of the correlation theory and are not entirely self consistent in that respect. In other words, these calculations assume that the correlation effect and solvation effect are largely additive, and the combination effect thereof is neglected. COSMO for MCSCF has not been implemented yet.

In the current implementation the code calculates the gas-phase energy of the system followed by the solution-phase energy, and returns the electrostatic contribution to the solvation free energy. At the present gradients are calculated analytically, but frequencies are calculated by finite difference of the gradients.

The non-electrostatic contributions can be calculated by turning on the SMD model. It should be noted that one must in general take into account the standard state correction besides the electrostatic and cavitation/dispersion contribution to the solvation free energy, when a comparison to experimental data is made.

13.1.2.2 COSMO Input Parameters

Invoking the COSMO solvation model is done by specifying the input COSMO input block with the input options as:

```
cosmo
  [off]
  [dielec  <real dielec default 78.4>]
  [parameters <filename>]
  [radius  <real atom1>
        <real atom2>
     . . .
        <real atomN>]
  [rsolv <real rsolv default 0.5>]
  [screen  <ideal||ks||st>]
  [iscren  <integer iscren default 0, deprecated>]
  [minbem  <integer minbem default 2>]
  [maxbem  <integer maxbem default 4, deprecated>]
  [ificos  <integer ificos default 0>]
  [lineq   <integer lineq default 1>]
  [zeta <real zeta default 0.98>]
  [gamma_s <real gammas default 1.0>]
  [sw_tol <real swtol default 1.0e-4>]
  [charge_correction <no||scale||lagrangian default lagrangian>]
  [do_gasphase  <logical do_gasphase default true>]
  [do_cosmo_ks]
  [do_cosmo_yk]
  [do_cosmo_smd]
  [cavity <vdw||ses default vdw>]
  [ndiv <coarse||fine||xfine||huge||xxl default fine>]
  [potcorr <logical potocorr default false>]
end
```

followed by the task directive specifying the wavefunction and type of calculation, e.g.,`task scf energy` , `task mp2 energy` , `task dft optimize` , etc.

13.1.2.2.1 COSMO: OFF keyword

`off` can be used to turn off COSMO in a compound (multiple task) run. By default, once the COSMO solvation model has been defined it will be used in subsequent calculations. Add the keyword `off` if COSMO is not needed in subsequent calculations.

13.1.2.2.2 COSMO: DIELEC keyword

`dielec` is the value of the dielectric constant\(\epsilon \) of the medium, with a default value of 78.4 (the dielectric constant for water). This value is used to scale the dielectric screening energy, as detailed in the SCREEN section.

13.1.2.2.3 COSMO: PARAMETERS keyword

`parameters` specifies COSMO radii parameters file that stores custom setting for COSMO parameters. The format for such file consists of the atom or element name followed by the radii. The program will first attempt to match based on atom name and only then the element name. Otherwise radius will be set based on default parameters. The file has to present in one of the three location (in the order of preference) - directory specified by the environmental variable `NWCHEM_COSMO_LIBRARY` , permanent directory, and run directory.

13.1.2.2.4 COSMO: RADIUS keyword

`radius` is an array that specifies the radius of the spheres associated with each atom and that make up the molecule-shaped cavity. These values will override default radii setting including those specified in the COSMO parameter file (if any) Default values are Van der Waals radii. Values are in units of angstroms. The codes uses the following Van der Waals radii by default:

Default radii provided by Andreas Klamt (Cosmologic)

vdw radii: 1.17 (± 0.02) * Bondi radius[3]

optimal vdw radii for H, C, N, O, F, S, Cl, Br, f

for heavy elements: 1.17*1.9

```
 data (vander(i),i=1,102)
1 / 1.300,1.638,1.404,1.053,2.0475,2.00,
2   1.830,1.720,1.720,1.8018,1.755,1.638,
3   1.404,2.457,2.106,2.160,2.05,2.223,
4   2.223,2.223,2.223,2.223,2.223,2.223,
5   2.223,2.223,2.223,2.223,2.223,2.223,
6   2.223,2.223,2.223,2.223,2.160,2.223,
7   2.223,2.223,2.223,2.223,2.223,2.223,
8   2.223,2.223,2.223,2.223,2.223,2.223,
9   2.223,2.223,2.223,2.223,2.320,2.223,
1   2.223,2.223,2.223,2.223,2.223,2.223,
2   2.223,2.223,2.223,2.223,2.223,2.223,
3   2.223,2.223,2.223,2.223,2.223,2.223,
4   2.223,2.223,2.223,2.223,2.223,2.223,
5   2.223,2.223,2.223,2.223,2.223,2.223,
6   2.223,2.223,2.223,2.223,2.223,2.223,
7   2.223,2.223,2.223,2.223,2.223,2.223,
7   2.223,2.223,2.223,2.223,2.223,2.223/
```

For examples see Stefanovich et al.[5] and Barone et al.[6]

13.1.2.2.5 COSMO: RSOLV keyword

rsolv is a parameter used to define the solvent accessible surface. See the original reference of Klamt and Schüürmann[1] for a description. The default value is 0.5 (in angstroms). It is not used in the York and Karplus model (used by default in NWChem).

13.1.2.2.6 COSMO: SCREEN keyword

The screen parameter defines the dielectric screening scaling.

- screen ideal applies no scaling, meaning $f(\epsilon) = 1$. In this case, the dielectric constant ϵ is irrelevant, as the solvent is treated as a perfect conductor with $\epsilon = \infty$.
- screen ks uses the original scaling by Klamt and Schüürmann[1], given by $f(\epsilon) = \frac{\epsilon - 1}{\epsilon + 1/2}$.
- screen st applies the modified scaling proposed by Stefanovich and Truong[5], defined as $f(\epsilon) = \frac{\epsilon - 1}{\epsilon}$.

By default, the modified scaling is used. For high dielectric values, the differences between these scaling methods become negligible.

13.1.2.2.7 COSMO: ISCREEN keyword (deprecated, see SCREEN)

iscren is a flag to define the dielectric charge scaling option. iscren 1 implies the original scaling from Klamt and Schüürmann, mainly “(ε-1)/(ε+1/2)”, where ε is the dielectric constant. iscren 0 implies the modified scaling suggested by Stefanovich and Truong[5], mainly “(ε-1)/ε“. Default is to use the modified scaling. For high dielectric the difference between the scaling is not significant.

13.1.2.2.8 COSMO: MINBEM and IFICOS keywords

The minbem and ificos parameters define the tesselation of the unit sphere. The approach still follows the original proposal by Klamt and Schüürmann to some degree. Basically a tesselation is generated from minbem refining passes starting from either an octahedron or an icosahedron. Each level of refinement partitions the triangles of the current tesselation into four triangles. This procedure is repeated recursively until the desired granularity of the tesselation is reached. The induced point charges from the polarization of the medium are assigned to the centers of the tesselation. The default value is minbem 2. The flag ificos serves to select the original tesselation, ificos 0 for an octahedron (default) and ificos 1 for an icosahedron. Starting from an icosahedron yields a somewhat finer tesselation that converges somewhat faster. Solvation energies are

not really sensitive to this choice for sufficiently fine tesselations. The old maxbem directive is no longer used for the York-Karplus model, while it was used in the Klamt-Schüürmann model .

13.1.2.2.9 COSMO: LINEQ keyword

The lineq parameter serves to select the numerical algorithm to solve the linear equations yielding the effective charges that represent the polarization of the medium. lineq 0 selects a dense matrix linear equation solver (default), lineq 1 selects an iterative method. For large molecules where the number of effective charges is large, the code selects the iterative method.

13.1.2.2.10 COSMO: ZETA keyword

zeta sets the width of the Gaussian charge distributions that were suggested by York and Karplus to avoid singularities when two surface charges coincide. The default value is zeta 0.98 this value was chosen to ensure that the results of the current implementation are as close as possible to those of the original Klamt and Schüürmann based implementation.

13.1.2.2.11 COSMO: GAMMA_S keyword

gamma_s modifies the width of the smooth switching function that eliminates surface charges when their positions move into the sphere of a neighboring atom. gamma_s 0.0 leads to a heavyside or abrupt switching function, whereas gamma_s 1.0 maximizes the width of the switching function. The default value is gamma_s 1.0 .

13.1.2.2.12 COSMO: SW_TOL keyword

sw_tol specifies the cutoff of the switching function below which a surface charge at a particular point is eliminated. The values of the switching function lie in the domain from 0 to 1. This value should not be set too small as that leads to instabilities in the linear system solvers. The default value is sw_tol 1.0e-4 .

13.1.2.2.13 COSMO: CHARGE_CORRECTION keyword

The charge_correction input option defines the surface charge correction to use.
With the input line charge_correction no , no surface charge corrections are used.
With the input line charge_correction scale , both the the electron surface charges and the nuclear surface charges are scaled.
With the default input line charge_correction lagrangian , a lagrangian constraint is used to variationally correct the charges.
The default is charge_correction lagrangian unless the Klamt-Schuurmann COSMO model is used when the default becomes charge_correction scale .

13.1.2.2.14 COSMO: DO_GASPHASE keyword

do_gasphase is a flag to control whether the calculation of the solvation energy is preceded by a gas phase calculation. The default is to always perform a gas phase calculation first and then calculate the solvation starting from the converged gas phase electron density. However, in geometry optimizations this approach can double the cost. In such a case setting do_gasphase false suppresses the gas phase calculations and only the solvated system calculations are performed. This option needs to be used with care as in some cases starting the COSMO solvation from an unconverged electron density can generate unphysical charges that lock the calculation into strange electron distributions.

13.1.2.2.15 COSMO: DO_COSMO_KS keyword

do_cosmo_ks is a flag to turn on the Klamt-Schüürmann model

13.1.2.2.16 COSMO: DO_COSMO_YK keyword

do_cosmo_yk is a flag to turn on the York-Karplus model (default)

13.1.2.2.17 COSMO: DO_COSMO_SMD keyword

do_cosmo_smd is a flag to turn on the SMD model. More details can be found at the SMD Model documentation

13.1.2.2.18 COSMO: CAVITY keyword

(New since August 31st 2025 in master branch)

cavity [vdw|ses]

The vdw dewfault option uses the York Karplus[2] prescription for creating the cavity.

Instead, the ses cavity construction approach creates a solvent-excluding surface (SES), using the GEPOL93 algorithm[7,8].

13.1.2.2.19 COSMO: NDIV keyword

(New since August 31st 2025 in master branch)

ndiv [coarse|fine|xfine|huge|xxl]

The ndiv keyword define the size of the SES cavity. ndiv fine is the default value.

13.1.2.2.20 COSMO: POTCORR keyword

(New since August 31st 2025 in master branch) potcorr [true| false]

Apply the outlying charge correction to the electrostatic potential as described in [8]

13.1.2.3 Examples

The following example is for a water molecule in ‘water’, using the HF/6-31G** level of theory:

```
start

geometry
 o              .0000000000      .0000000000      -.0486020332
 h              .7545655371      .0000000000       .5243010666
 h             -.7545655371      .0000000000       .5243010666
end
basis
 o library 6-31g**
 h library 6-31g**
end
cosmo
 dielec 78.0
 radius 1.40
        1.16
        1.16
 lineq  0
end
task scf energy
```

Alternatively, instead of listing COSMO radii parameters in the input, the former can be loaded using an external file through the parameters directive

```
start

geometry
 ow             .0000000000      .0000000000      -.0486020332
 hw             .7545655371      .0000000000       .5243010666
 h             -.7545655371      .0000000000       .5243010666
end
basis
 * library 6-31g**
end

cosmo
 dielec 78.0
 lineq  0
 parameters water.par
end

task scf energy
```

where the water.par file has the following form:

```
O 1.40
H 1.16
```

This will set radii of all oxygen atoms to 1.4 and all hydrogen atoms to 1.16. More fine grained control may be achieved using specific atom names. For example, the following parameter file

```
O   1.40
H   1.16
HW  1.06
```

will set a different radii of 1.06 to hydrogen atoms named HW. Note that, as per general rule in NWChem, all names are case insensitive.

and placed in one of the these locations - directory specified by the environmental variable NWCHEM_COSMO_LIBRARY , permanent directory, or run directory.

13.1.3 SMD

13.1.3.1 Overview

SMD denotes “solvation model based on density” and it is described in detail in the 2009 paper by Marenich, Cramer and Truhlar[9].

The SMD model is a universal continuum solvation model where “universal” denotes its applicability to any charged or uncharged solute in any solvent or liquid medium for which a few key descriptors are known. The word “continuum” denotes that the solvent is not represented explicitly as a collection of discrete solvent molecules but rather as a dielectric medium with surface tensions at the solute-solvent interface.

SMD directly calculates the free energy of solvation of an ideal solvation process that occurs at fixed concentration (for example, from an ideal gas at a concentration of 1 mol/L to an ideal solution at a liquid-phase concentration of 1 mol/L) at 298 K, but this may converted by standard thermodynamic formulas to a standard-state free energy of solvation, which is defined as the transfer of molecules from an ideal gas at 1 bar to an ideal 1 molar solution.

The SMD model separates the fixed-concentration free energy of solvation into two components. The first component is the bulk-electrostatic contribution arising from a self-consistent reaction field (SCRF) treatment. The SCRF treatment involves an integration of the nonhomogeneous-dielectric Poisson equation for bulk electrostatics in terms of the COSMO model of Klamt and Schüürmann with the modified COSMO scaling factor suggested by Stefanovich and Truong and by using the SMD intrinsic atomic Coulomb radii. These radii have been optimized for H, C, N, O, F, Si, P, S, Cl, and Br. For any other atom the current implementation of the SMD model uses scaled values of the van der Waals radii of Mantina et al[10].

The scaling factor equals 1.52 for group 17 elements heavier than Br (i.e., for I and At) and 1.18 for all other elements for which there are no optimized SMD radii.

The second contribution to the fixed-concentration free energy of solvation is the contribution arising from short-range interactions between the solute and solvent molecules in the first solvation shell. This contribution is called the cavity–dispersion–solvent-structure (CDS) term, and it is a sum of terms that are proportional (with geometry-dependent proportionality constants called atomic surface tensions) to the solvent-accessible surface areas (SASAs) of the individual atoms of the solute.

13.1.3.2 SMD Input Parameters

The SMD model requires additional parameters in the COSMO input block

```
cosmo
  [do_cosmo_smd <logical>]
  [solvent (keyword)]
  [icds <integer>]
  [sola <real>]
  [solb <real>]
  [solc <real>]
  [solg <real>]
  [solh <real>]
  [soln <real>]
end
```

At the moment the SMD model is available in NWChem only with the DFT block

The SMD input options are as follows:

```
do_cosmo_smd <logical>
```

The `do_cosmo_smd` keyword instructs NWChem to perform a ground-state SMD calculation when set to a `true` value.

13.1.3.2.1 SMD: SOLVENT keyword

```
solvent (keyword)
```

a `solvent` keyword from the short name entry in the list of available SMD solvent names.

When a solvent is specified by name, the descriptors for the solvent are based on the Minnesota Solvent Descriptor Database[11].

The user can specify a solvent (by using a string using up to eight characters) that is not on the list by using a new solvent keyword and introducing user-provided values for the following solvent descriptors:

13.1.3.2.2 SMD: DIELEC keyword

```
dielec (real input)
```

dielectric constant at 298 K

13.1.3.2.3 SMD: SOLA keyword

```
sola (real input)
```

Abraham's hydrogen bond acidity

13.1.3.2.4 SMD: SOLB keyword

```
solb (real input)
```

Abraham's hydrogen bond basicity

13.1.3.2.5 SMD: SOLC keyword

```
solc (real input)
```

aromaticity as a fraction of non-hydrogenic solvent atoms that are aromatic carbon atoms

13.1.3.2.6 SMD: SOLG keyword

```
solg (real input)
```

macroscopic surface tension of the solvent at an air/solvent interface at 298 K in units of cal mol^{-1} $Å^{-2}$
(note that 1 dyne/cm = 1.43932 cal mol^{-1} $Å^{-2}$)

13.1.3.2.7 SMD: SOLH keyword

```
solh (real input)
```

electronegative halogenicity as the fraction of non-hydrogenic solvent atoms that are F, Cl, or Br

13.1.3.2.8 SMD: SOLN keyword

```
soln (real input)
```

index of refraction at optical frequencies at 293 K

13.1.3.2.9 SMD: ICDS keyword

```
icds (integer input)
```

`icds` should have a value of 1 for water. `icds` should have a value of 2 for any nonaqueous solvent.
If `icds` is set equal to 2, then you need to provide the following solvent descriptors (see the MN solvent descriptor database):

- sola
- solb
- solc
- solg
- solh
- soln

13.1.3.3 SMD Examples

13.1.3.4 SMD Example: water solvent

```
echo
title "SMD/M06-2X/6-31G(d) solvation free energy for CF3COO- in water"
start
charge -1
geometry nocenter
C    0.512211   0.000000  -0.012117
C   -1.061796   0.000000  -0.036672
O   -1.547400   1.150225  -0.006609
O   -1.547182  -1.150320  -0.006608
F    1.061911   1.087605  -0.610341
F    1.061963  -1.086426  -0.612313
F    0.993255  -0.001122   1.266928
symmetry c1
end
basis
* library 6-31G*
end
dft
 XC m06-2x
end
cosmo
 do_cosmo_smd true
 solvent water
end
task dft energy
```

13.1.3.5 SMD Example: new solvent

Example using a user defined solvent, not present in the SMD list of solvents

```
echo
title "SMD/M06-2X/6-31G(d) solvation free energy for CF3COO- in my solvent"
start
charge -1
geometry nocenter
C    0.512211   0.000000  -0.012117
C   -1.061796   0.000000  -0.036672
O   -1.547400   1.150225  -0.006609
O   -1.547182  -1.150320  -0.006608
F    1.061911   1.087605  -0.610341
F    1.061963  -1.086426  -0.612313
F    0.993255  -0.001122   1.266928
symmetry c1
end
basis
* library 6-31G*
end
dft
 XC m06-2x
end
cosmo
 do_cosmo_smd true
 solvent mysolv
 dielec 11.4
 sola 1.887
 solb 0.0
 soln 0.98
 icds 2
end
task dft energy
```

13.1.4 Solvents List - Solvent keyword

The short name for the solvent from the table can be used with the `solvent` keyword to define the solvent. Example with acetonitrile.

```
cosmo
  solvent acetntrl
end
```

Long name	short name	dielec
acetic acid	acetacid	6.2528
acetone	acetone	20.493
acetonitrile	acetntrl	35.688
acetophenone	acetphen	17.440
aniline	aniline	6.8882
anisole	anisole	4.2247
benzaldehyde	benzaldh	18.220
benzene	benzene	2.2706
benzonitrile	benzntrl	25.592
benzyl chloride	benzylcl	6.7175
1-bromo-2-methylpropane	brisobut	7.7792
bromobenzene	brbenzen	5.3954
bromoethane	brethane	9.01

Long name	short name	dielec
bromoform	bromform	4.2488
1-bromooctane	broctane	5.0244
1-bromopentane	brpentan	6.269
2-bromopropane	brpropa2	9.3610
1-bromopropane	brpropan	8.0496
butanal	butanal	13.450
butanoic acid	butacid	2.9931
1-butanol	butanol	17.332
2-butanol	butanol2	15.944
butanone	butanone	18.246
butanonitrile	butantrl	24.291
butyl acetate	butile	4.9941
butylamine	nba	4.6178
n-butylbenzene	nbutbenz	2.360
sec-butylbenzene	sbutbenz	2.3446
tert-butylbenzene	tbutbenz	2.3447
carbon disulfide	cs2	2.6105
carbon tetrachloride	carbntet	2.2280
chlorobenzene	clbenzen	5.6968
sec-butyl chloride	secbutcl	8.3930
chloroform	chcl3	4.7113
1-chlorohexane	clhexane	5.9491
1-chloropentane	clpentan	6.5022
1-chloropropane	clpropan	8.3548
o-chlorotoluene	ocltolue	4.6331
m-cresol	m-cresol	12.440
o-cresol	o-cresol	6.760
cyclohexane	cychexan	2.0165
cyclohexanone	cychexon	15.619
cyclopentane	cycpentn	1.9608

Long name	short name	dielec
cyclopentanol	cycpntol	16.989
cyclopentanone	cycpnton	13.58
cis-decalin	declncis	2.2139
trans-decalin	declntra	2.1781
decalin (cis/trans mixture)	declnmix	2.196
n-decane	decane	1.9846
1-decanol	decanol	7.5305
1,2-dibromoethane	edb12	4.9313
dibromomethane	dibrmetn	7.2273
dibutyl ether	butyleth	3.0473
o-dichlorobenzene	odiclbnz	9.9949
1,2-dichloroethane	edc12	10.125
cis-dichloroethylene	c12dce	9.200
trans-dichloroethylene	t12dce	2.140
dichloromethane	dcm	8.930
diethyl ether	ether	4.2400
diethyl sulfide	et2s	5.723
diethylamine	dietamin	3.5766
diiodomethane	mi	5.320
diisopropyl ether	dipe	3.380
dimethyl disulfide	dmds	9.600
dimethylsulfoxide	dmso	46.826
N,N-dimethylacetamide	dma	37.781
cis-1,2-dimethylcyclohexane	cisdmchx	2.060
N,N-dimethylformamide	dmf	37.219
2,4-dimethylpentane	dmepen24	1.8939
2,4-dimethylpyridine	dmepyr24	9.4176
2,6-dimethylpyridine	dmepyr26	7.1735
1,4-dioxane	dioxane	2.2099
diphenyl ether	phoph	3.730

Long name	short name	dielec
dipropylamine	dproamin	2.9112
n-dodecane	dodecan	2.0060
1,2-ethanediol	meg	40.245
ethanethiol	etsh	6.667
ethanol	ethanol	24.852
ethyl acetate	etoac	5.9867
ethyl formate	etome	8.3310
ethylbenzene	eb	2.4339
ethylphenyl ether	phenetol	4.1797
fluorobenzene	c6h5f	5.420
1-fluorooctane	foctane	3.890
formamide	formamid	108.94
formic acid	formacid	51.100
n-heptane	heptane	1.9113
1-heptanol	heptanol	11.321
2-heptanone	heptnon2	11.658
4-heptanone	heptnon4	12.257
n-hexadecane	hexadecn	2.0402
n-hexane	hexane	1.8819
hexanoic acid	hexnacid	2.600
1-hexanol	hexanol	12.51
2-hexanone	hexanon2	14.136
1-hexene	hexene	2.0717
1-hexyne	hexyne	2.615
iodobenzene	c6h5i	4.5470
1-iodobutane	iobutane	6.173
iodoethane	c2h5i	7.6177
1-iodohexadecane	iohexdec	3.5338
iodomethane	ch3i	6.8650
1-iodopentane	iopentan	5.6973

Long name	short name	dielec
1-iodopropane	iopropan	6.9626
isopropylbenzene	cumene	2.3712
p-isopropyltoluene	p-cymene	2.2322
mesitylene	mesityln	2.2650
methanol	methanol	32.613
2-methoxyethanol	egme	17.200
methyl acetate	meacetat	6.8615
methyl benzoate	mebnzate	6.7367
methyl butanoate	mebutate	5.5607
methyl formate	meformat	8.8377
4-methyl-2-pentanone	mibk	12.887
methyl propanoate	mepropyl	6.0777
2-methyl-1-propanol	isobutol	16.777
2-methyl-2-propanol	terbutol	12.470
N-methylaniline	nmeaniln	5.9600
methylcyclohexane	mecychex	2.024
N-methylformamide (E/Z mixture)	nmfmixtr	181.56
2-methylpentane	isohexan	1.890
2-methylpyridine	mepyrid2	9.9533
3-methylpyridine	mepyrid3	11.645
4-methylpyridine	mepyrid4	11.957
nitrobenzene	c6h5no2	34.809
nitroethane	c2h5no2	28.290
nitromethane	ch3no2	36.562
1-nitropropane	ntrprop1	23.730
2-nitropropane	ntrprop2	25.654
o-nitrotoluene	ontrtolu	25.669
n-nonane	nonane	1.9605
1-nonanol	nonanol	8.5991
5-nonanone	nonanone	10.600

Long name	short name	dielec
n-octane	octane	1.9406
1-octanol	octanol	9.8629
2-octanone	octanon2	9.4678
n-pentadecane	pentdecn	2.0333
pentanal	pentanal	10.000
n-pentane	npentane	1.8371
pentanoic acid	pentacid	2.6924
1-pentanol	pentanol	15.130
2-pentanone	pentnon2	15.200
3-pentanone	pentnon3	16.780
1-pentene	pentene	1.9905
E-2-pentene	e2penten	2.051
pentyl acetate	pentacet	4.7297
pentylamine	pentamin	4.2010
perfluorobenzene	pfb	2.029
phenylmethanol	benzalcl	12.457
propanal	propanal	18.500
propanoic acid	propacid	3.440
1-propanol	propanol	20.524
2-propanol	propnol2	19.264
propanonitrile	propntrl	29.324
2-propen-1-ol	propenol	19.011
propyl acetate	propacet	5.5205
propylamine	propamin	4.9912
pyridine	pyridine	12.978
tetrachloroethene	c2cl4	2.268
tetrahydrofuran	thf	7.4257
tetrahydrothiophene-S,S-dioxide	sulfolan	43.962
tetralin	tetralin	2.771
thiophene	thiophen	2.7270

Long name	short name	dielec
thiophenol	phsh	4.2728
toluene	toluene	2.3741
tributyl phosphate	tbp	8.1781
1,1,1-trichloroethane	tca111	7.0826
1,1,2-trichloroethane	tca112	7.1937
trichloroethene	tce	3.422
triethylamine	et3n	2.3832
2,2,2-trifluoroethanol	tfe222	26.726
1,2,4-trimethylbenzene	tmben124	2.3653
2,2,4-trimethylpentane	isoctane	1.9358
n-undecane	undecane	1.9910
m-xylene	m-xylene	2.3478
o-xylene	o-xylene	2.5454
p-xylene	p-xylene	2.2705
xylene (mixture)	xylenemx	2.3879
water	h2o	78.400

13.1.5 Usage Tips

Authors of paper [9] report that

" … *the SMD/COSMO/NWChem calculations we employed finer grids (options minbem=3, maxbem=4, ificos=1) because the default NWChem tessellation parameters (options: minbem=2, maxbem=3, ificos=0) produced very large errors in solvation free energies*."

Since the `maxbem` keyword is no longer in use, this paper's recommended input translate into

```
cosmo
 minbem 3
 ificos 1
end
```

13.1.6 References

1. Klamt, A.; Schüürmann, G. COSMO: A New Approach to Dielectric Screening in Solvents with Explicit Expressions for the Screening Energy and Its Gradient. *J. Chem. Soc., Perkin Trans. 2* **1993**, No. 5, 799–805. https://doi.org/10.1039/p29930000799
2. York, D. M.; Karplus, M. A Smooth Solvation Potential Based on the Conductor-Like Screening Model. *The Journal of Physical Chemistry A* **1999**, *103* (50), 11060–11079. https://doi.org/10.1021/jp992097l
3. Bondi, A. Van Der Waals Volumes and Radii. *The Journal of Physical Chemistry* **1964**, *68* (3), 441–451. https://doi.org/10.1021/j100785a001
4. Klamt, A.; Jonas, V.; Bürger, T.; Lohrenz, J. C. W. Refinement and Parametrization of COSMO-RS. *The Journal of Physical Chemistry A* **1998**, *102* (26), 5074–5085. https://doi.org/10.1021/jp980017s
5. Stefanovich, E. V.; Truong, T. N. Optimized Atomic Radii for Quantum Dielectric Continuum Solvation Models. *Chemical Physics Letters* **1995**, *244* (1-2), 65–74. https://doi.org/10.1016/0009-2614(95)00898-e

6. Barone, V.; Cossi, M.; Tomasi, J. A New Definition of Cavities for the Computation of Solvation Free Energies by the Polarizable Continuum Model. *The Journal of Chemical Physics* **1997**, *107* (8), 3210–3221. https://doi.org/10.1063/1.474671
7. Pascual-Ahuir, J. L.; Silla, E.; Tuñon, I. GEPOL: An Improved Description of Molecular Surfaces. III. A New Algorithm for the Computation of a Solvent-Excluding Surface. *Journal of Computational Chemistry* **1994**, *15* (10), 1127–1138. https://doi.org/10.1002/jcc.540151009
8. Soares, R. de P.; Mejía-Rodriguez, D.; Aprà, E. Recent Improvements to the NWChem COSMO Module. *Journal of Chemical Theory and Computation* **2025**, *21* (22), 11573–11584. https://doi.org/10.1021/acs.jctc.5c01368
9. Marenich, A. V.; Cramer, C. J.; Truhlar, D. G. Universal Solvation Model Based on Solute Electron Density and on a Continuum Model of the Solvent Defined by the Bulk Dielectric Constant and Atomic Surface Tensions. *The Journal of Physical Chemistry B* **2009**, *113* (18), 6378–6396. https://doi.org/10.1021/jp810292n
10. Haynes, W. M. CRC Handbook of Chemistry and Physics; Mantina, M., Valero, R., Cramer, C. J., Truhlar, D. G., Eds.; Taylor & Francis Group, 2013; pp 9–49.
11. Winget, P.; Dolney, D. M.; Giesen, D. J.; Cramer, C. J.; Truhlar, D. G. Minnesota Solvent Descriptor Database. *Minneapolis, MN: Department of Chemistry and Supercomputer Institute* **1999**.

13.2 VEM (Vertical Excitation or Emission) Model

13.2.1 Overview

The VEM is a model for calculating the vertical excitation (absorption) or vertical emission (fluorescence) energy in solution according to a two-time-scale model of solvent polarization. The model is described in reference[1].

The current implementation is based on the VEM(d,RD) algorithm as described in the above paper. The method is available only at the TDDFT level of theory, including both full-linear response TDDFT (sometimes called LR-TDDFT or regular TDDFT) and the Tamm–Dancoff approximation, TDDFT-TDA (sometimes just called TDA). The configuration interaction singles (CIS) wave function method can also be used along with VEM by considering CIS to be a special case of TDDFT-TDA.

The abbreviation VEM originally referred to the vertical excitation model of reference[2], but the current implementation of VEM extends to both excitation and emission calculations in solution, and the E in VEM now stands for excitation/emission. Furthermore, the current version of VEM (based on the Marenich et al. paper[1]) does not reduce to the original VEM of Li et al., but is improved as described in reference[1].

The VEM model is based on a nonequilibrium dielectric-continuum approach in terms of two-time-scale solvent response. The solvent's bulk-electrostatic polarization is described by using the reaction field partitioned into slow and fast components, and only the fast component is self-consistently (iteratively) equilibrated with the charge density of the solute molecule in its final state. During the VEM calculation, the slow component is kept in equilibrium with the initial state's solute charge density but not with the final state's one. In the case of vertical absorption the initial state is the ground electronic state of the solute molecule in solution and the final state is an excited electronic state in solution (and vice versa in the case of an emission spectrum). Both the ground- and excited-state calculations involve an integration of the nonhomogeneous-dielectric Poisson equation for bulk electrostatics in terms of the COSMO model as implemented in NWChem with the modified COSMO scaling factor (iscren 0) and by using the SMD intrinsic atomic Coulomb radii (by default; see the section of the manual describing SMD). The excited-state electron density is calculated using the Z-Vector "relaxed density" approach.

The VEM excitation or emission energy includes only a bulk-electrostatic contribution without any cavity–dispersion–solvent-structure (CDS) contributions (such contributions are used in SMD ground-state calculations as described in the SMD section of this manual, but are not used in VEM calculations). When one considers solvatochromic shifts, the main contributions beyond bulk electrostatics are solute–solvent dispersion interactions, hydrogen bonding (the latter is most important in protic solvents), and perhaps charge transfer between the solute and the solvent. To explicitly account for solute–solvent charge transfer and hydrogen bonding, the user can run a VEM calculation on a supersolute that involves a solute–solvent molecular cluster with one or a few solvent molecules added explicitly to a bare solute. The solute–solvent dispersion contribution to the solvatochromic shift, if desired, can be estimated by the solvation model with state-specific polarizability (SMSSP) described in reference[3].

In this case, the user needs to provide values of ground- and excited-state spherically averaged molecular polarizabilities

of the solvent.

13.2.2 Syntax

The VEM-specific input options are as follows:

13.2.2.1 DO_COSMO_VEM:

do_cosmo_vem <integer do_cosmo_vem default 0>

The do_cosmo_vem can be set to the following values:
- 0 (do not do any VEM calculation even if the task tddft gradient line is present; default).
- 1 (do a nonequilibrium VEM excitation energy calculation;
in this case the task tddft gradient line should be present, too)
- 2 (do an equilibrium VEM excitation energy calculation followed by
a nonequilibrium emission energy calculation;
task tddft gradient line should be present)

13.2.2.2 VEM Solvent

The VEM solvent (which is water by default) can be specified by using the solvent keyword described in the SMD section of this manual or by specifying the VEM solvent descriptors such as

dielec (real input)

static dielectric constant

dielecinf (real input)

optical dielectric constant which is set (by default) to the squared value of the solvent's index of refraction
(see the keyword soln , but note that if the solvent is specified with the solvent keyword,
the refractive index is set by the program without needing the user to supply it.)

Solvent descriptors set by the program are based on the Minnesota Solvent Descriptor Database[4]:

If the option do_cosmo_vem 1 or do_cosmo_vem 2 is specified the program will run VEM ground- and excited-state bulk-electrostatic calculations using the COSMO algorithm with the SMD Coulomb radii by default. If the user wants to use the default COSMO radii in such calculations (this is not recommended) the option do_cosmo_smd .false. should be specified.

13.2.3 SMSSP estimate of the solute–solvent dispersion contribution

If the SMSSP estimate of a solute–solvent dispersion contribution to the solvatochromic shift is desired, the following options should be used:

polgs_cosmo_vem (real input)

user-provided value of the spherically-averaged molecular polarizability of the solute in the ground state (in $Å^3$)

poles_cosmo_vem (real input)

user-provided value of the spherically-averaged molecular polarizability of the solute in an exited state of interest (in $Å^3$)

13.2.4 Examples

An example of the VEM input file is provided below.

```
echo
title  'VEM/TDDFT-B3LYP/6-311+G(d) vertical excitation energy + SMSSP for formaldehyde in methanol'
start
geometry nocenter
 O    0.0000000000    0.0000000000    0.6743110000
 C    0.0000000000    0.0000000000   -0.5278530000
 H    0.0000000000    0.9370330000   -1.1136860000
 H    0.0000000000   -0.9370330000   -1.1136860000
symmetry c1
end
basis
* library 6-311+G*
end
dft
 XC b3lyp
end
cosmo
 do_cosmo_smd true
 do_cosmo_vem 1
 solvent methanol
 polgs_cosmo_vem 2.429
 poles_cosmo_vem 3.208
end
tddft
 nroots 10
 target 1
 singlet
 notriplet
 algorithm 1
 civecs
end
grad
  root 1
  solve_thresh 1d-05
end
task tddft gradient
```

13.2.5 References

1. Marenich, A. V.; Cramer, C. J.; Truhlar, D. G.; Guido, C. A.; Mennucci, B.; Scalmani, G.; Frisch, M. J. Practical Computation of Electronic Excitation in Solution: Vertical Excitation Model. *Chemical Science* **2011**, *2* (11), 2143. https://doi.org/10.1039/c1sc00313e
2. Li, J.; Cramer, C. J.; Truhlar, D. G. Two-Response-Time Model Based on CM2/INDO/S2 Electrostatic Potentials for the Dielectric Polarization Component of Solvatochromic Shifts on Vertical Excitation Energies. *International Journal of Quantum Chemistry* **2000**, *77* (1), 264–280. https://doi.org/10.1002/(sici)1097-461x(2000)77:1<264::aid-qua24>3.0.co;2-j
3. Marenich, A. V.; Cramer, C. J.; Truhlar, D. G. Uniform Treatment of Solute-Solvent Dispersion in the Ground and Excited Electronic States of the Solute Based on a Solvation Model with State-Specific Polarizability. *Journal of Chemical Theory and Computation* **2013**, *9* (8), 3649–3659. https://doi.org/10.1021/ct400329u
4. Winget, P.; Dolney, D. M.; Giesen, D. J.; Cramer, C. J.; Truhlar, D. G. Minnesota Solvent Descriptor Database. *Minneapolis, MN: Department of Chemistry and Supercomputer Institute* **1999**.

13.3 Hybrid Calculations with ONIOM

13.3.1 Overview

ONIOM is the hybrid method of Morokuma and co-workers that enables different levels of theory to be applied to different parts of a molecule/system and combined to produce a consistent energy expression. The objective is to perform a high-level calculation on just a small part of the system and to include the effects of the remainder at lower levels of theory, with the end result being of similar accuracy to a high-level calculation on the full system.

1. M. Svensson, S. Humbel, R.D.J. Froese, T. Mastubara, S. Sieber, and K. Morokuma, J. Phys. Chem., 100, 19357 (1996).
2. S. Dapprich, I. Komaromi, K.S. Byun, K. Morokuma, and M.J. Frisch, J. Mol. Struct. (Theochem), 461-462, 1 (1999).
3. R.D.J. Froese and K. Morokuma in “Encylopedia of Computational Chemistry”, volume 2, pp.1244-1257, (ed. P. von Rague Schleyer, John Wiley and Sons, Chichester, Sussex, 1998).

The NWChem ONIOM module implements two- and three-layer ONIOM models for use in energy, gradient, geometry optimization, and vibrational frequency calculations with any of the pure quantum mechanical methods within NWChem. At the present time, it is not possible to perform ONIOM calculations with either solvation models or classical force fields. Nor is it yet possible to compute properties except as derivatives of the total energy.

Using the terminology of Morokuma et al., the full molecular geometry including all atoms is referred to as the “real” geometry and it is treated using a “low”-level of theory. A subset of these atoms (referred to as the “model” geometry) are treated using both the “low”-level and a “high”-level of theory. A three-layer model also introduces an “intermediate” model geometry and a “medium” level of theory.

The two-layer model requires a high and low level of theory and a real and model molecular geometry. The energy at the high-level of theory for the real geometry is estimated as

```
E(High,Real) = E(Low,Real) + [E(High,Model) - E(Low,Model)].
```

The three-layer model requires high, medium and low levels of theory, and real, intermediate and model geometries and the corresponding energy estimate is

```
E(High,Real) = E(Low,Real) + [E(High,Model) - E(Medium,Model)]
          +  [E(Medium,Inter) - E(Low,Inter)].
```

When does ONIOM work well? The approximation for a two-layer model will be good if

- the model system includes the interactions that dominate the energy difference being computed and the high-level of theory describes these to the required precision, and
- the interactions between the model and the rest of the real system (substitution effects) are described to sufficient accuracy at the lower level of theory.

ONIOM is used to compute energy differences and the absolute energies are not all that meaningful even though they are well defined. Due to cancellation of errors, ONIOM actually works better than you might expect, but a poorly designed calculation can yield very bad results. Please read and heed the caution at the end of the article by Dapprich et al.

The input options are as follows

```
ONIOM
  HIGH   <string theory> [basis <string basis default "ao basis">] \
                 [ecp <string ecp>] [input <string input>]
 [MEDIUM <string theory> [basis <string basis default "ao basis">] \
                 [ecp <string ecp>] [input <string input>]]
  LOW    <string theory> [basis <string basis default "ao basis">] \
                 [ecp <string ecp>] [input <string input>]
  MODEL <integer natoms> [charge <double charge>] \
                 [<integer i1 j1> <real g1> [<string tag1>] ...]
 [INTER <integer natoms> [charge <double charge>] \
                 [<integer i1 j1> <real g1> [<string tag1>] ...]]
 [VECTORS [low-real <string mofile>] [low-model <string mofile>] \
       [high-model <string mofile>] [medium-model <string mofile]\
       [medium-inter <string mofile>] [low-inter <string mofile>]]
 [PRINT ...]
 [NOPRINT ...]
END
```

which are described in detail below.

For better validation of user input, the HIGH , LOW and MODEL directives must always be specified. If the one of the MEDIUM or INTER directives are specified, then so must the other.

13.3.2 Real, model and intermediate geometries

The geometry and total charge of the full or real system should be specified as normal using the geometry directive. If N_{model} of the atoms are to be included in the model system, then these should be specified first in the geometry. Similarly, in a three-layer calculation, if there are N_{inter} atoms to be included in the intermediate system, then these should also be

arranged together at the beginning of the geometry. The implict assumption is that the model system is a subset of the intermediate system which is a subset of the real system. The number of atoms to be included in the model and intemediate systems are specified using the MODEL and INTER directives. Optionally, the total charge of the model and intermediate systems may be adjusted. The default is that all three systems have the same total charge.

Example 1. A two-layer calculation on $K^+(H_2O)$ taking the potassium ion as the model system. Note that no bonds are broken so no link atoms are introduced. The real geometry would be specified with potassium (the model) first.

```
geometry autosym
  K  0    0.00    1.37
  O  0    0.00   -1.07
  H  0   -0.76   -1.68
  H  0    0.76   -1.68
end
```

and the following directive in the ONIOM input block indicates that one atom (implicitly the first in the geometry) is in the model system

```
model 1
```

13.3.2.1 Link atoms

Link atoms for bonds spanning two regions are automatically generated from the bond information. The additional parameters on the MODEL and INTER directives describe the broken bonds including scale factors for placement of the link atom and, optionally, the type of link atom. The type of link atom defaults to hydrogen, but any type may be specified (actually here you are specifying a geometry tag which is used to associate a geometrical center with an atom type and basis sets, etc. For each broken bond specify the numbers of the two atoms (i and j), the scale factor (g) and optionally the tag of the link atom. Link atoms are placed along the vector connecting the the first to the second atom of the bond according to the equation

$$\underline{R}_{link} = (1-g)\underline{R}_{1} + g*\underline{R}_{2}$$

where *g* is the scale factor. If the scale factor is one, then the link atom is placed where the second atom was. More usually, the scale factor is less than one, in which case the link atom is placed between the original two atoms. The scale factor should be chosen so that the link atom (usually hydrogen) is placed near its equilibrium bond length from the model atom. E.g., when breaking a single carbon-carbon bond (typical length 1.528 Angstrøms) using a hydrogen link atom we will want a carbon-hydrogen bond length of about 1.084 Angstrøms, so the scale factor should be chosen as 1.084/1.528 ~ 0.709.

Example 2. A calculation on acetaldehyde (H_3C-CHO) using aldehyde (H-CHO) as the model system. The covalent bond between the two carbon atoms is broken and a link atom must be introduced to replace the methyl group. The link atom is automatically generated – all you need to do is specify the atoms in the model system that are also in the real system (here CHO) and the broken bonds. Here is the geometry of acetaldehyde with the CHO of aldehyde first

```
geometry
  C   -0.383    0.288    0.021
  H   -1.425    0.381    0.376
  O    0.259    1.263   -0.321

  H    0.115   -1.570    1.007
  H   -0.465   -1.768   -0.642
  H    1.176   -1.171   -0.352
  C    0.152   -1.150    0.005
end
```

There are three atoms (the first three) of the real geometry included in the model geometry, and we are breaking the bond between atoms 1 and 7, replacing atom 7 with a hydrogen link atom. This is all accomplished by the directive

```
model 3   1 7 0.709 H
```

Since the default link atom is hydrogen there is actually no need to specify the “H”.

See also the ONIOM three layer example for a more complex example.

13.3.2.2 Numbering of the link atoms

The link atoms are appended to the atoms of the model or intermediate systems in the order that the broken bonds are specified in the input. This is of importance only if manually constructing an initial guess.

13.3.3 High, medium and low theories

The two-layer model requires both the high-level and low-level theories be specified. The three-layer model also requires the medium-level theory. Each of these includes a theory (such as SCF, MP2, DFT, CCSD, CCSD(T), etc.), an optional basis set, an optional ECP, and an optional string containing general NWChem input.

13.3.3.1 Basis specification

The basis name on the theory directive (high, medium, or low) is that specified on a basis set directive (see Section 7) and not the name of a standard basis in the library. If not specified, the basis set for the high-level theory defaults to the standard “ao basis”. That for the medium level defaults to the high-level basis, and the low-level basis defaults to the medium-level basis. Other wavefunction parameters are obtained from the standard wavefunction input blocks. See Effective core potentials for an example.

13.3.3.2 Effective core potentials

If an effective core potential is specified in the usual fashion outside of the ONIOM input then this will be used in all calculations. If an alternative ECP name (the name specified on the ECP directive in the same manner as done for basis sets) is specified on one of the theory directives, then this ECP will be used in preference for that level of theory. See the ONIOM three layer example for sample input.

13.3.3.3 General input strings

For many purposes, the ability to specify the theory, basis and effective core potential is adequate. All of the options for each theory are determined from their independent input blocks. However, if the same theory (e.g., DFT) is to be used with different options for the ONIOM theoretical models, then the general input strings must be used. These strings are processed as NWChem input each time the theoretical model is invoked. The strings may contain any NWChem input, except for options pertaining to ONIOM and the `task` directive. The intent that the strings be used just to control the options pertaining to the theory being used.

A word of caution. Be sure to check that the options are producing the desired results. Since the NWChem database is persistent and the ONIOM calculations happen in an undefined order, the input strings should fully define the calculation you wish to have happen.

For instance, if the high model is DFT/B3LYP/6-311g** and the low model is DFT/LDA/3-21g, the ONIOM input might look like this

```
oniom
  model 3
  low  dft basis 3-21g    input "dft; xc; end"
  high dft basis 6-311g** input "dft; xc b3lyp; end"
end
```

The empty XC directive restores the default LDA exchange-correlation option (see Section 11.3). Note that semi-colons and other quotation marks inside the input string must be preceded by a backslash to avoid special interpretation.

See |DFT with and without charge fitting for another example.

13.3.4 Use of symmetry

Symmetry should work just fine as long as the model and intermediate regions respect the symmetry – i.e., symmetry equivalent atoms need to be treated equivalently. If symmetry equivalent atoms must be treated in separate regions then the symmetry must be lowered (or completely switched off).

13.3.5 Molecular orbital files

The VECTORS directive in the ONIOM block is different to that elsewhere in NWChem. For each of the necessary combinations of theory and geometry you can specify a different file for the molecular orbitals. By default each combination will store the MO vectors in the permanent directory using a file name created by appending to the name of the calculation the following string

- low-real – ".lrmos"
- low-inter – ".limos"
- low-model – ".lmmos"
- medium-inter – ".mimos"
- medium-model – ".mmmos"
- high-model – ".hmmos"

Each calculation will utilize the appropriate vectors which is more efficient during geometry optimizations and frequency calculations, and is also useful for the initial calculation. In the absence of existing MO vectors files, the default atomic guess is used (see |Input/output of MO vectors).

If special measures must be taken to converge the initial SCF, DFT or MCSCF calculation for one or more of the systems, then initial vectors may be saved in a file with the default name, or another name may be specified using the VECTORS directive. Note that subsequent vectors (e.g., from a geometry optimization) will be written back to this file, so take a copy if you wish to preserve it. To generate the initial guess for the model or intermediate systems it is necessary to generate the geometries which is most readily done, if there are link atoms, by just running NWChem on the input for the ONIOM calculation on your workstation. It will print these geometries before starting any calculations which you can then terminate.

E.g., in a calculation on Fe(III) surrounded by some ligands, it is hard to converge the full (real) system from the atomic guess so as to obtain a d^5 configuration for the iron atom since the *d* orbitals are often nominally lower in energy than some of the ligand orbitals. The most effective mechanism is to converge the isolated Fe(III) and then to use the fragment guess as a starting guess for the real system. The resulting converged molecular orbitals can be saved either with the default name (as described above in this section), in which case no additional input is necessary. If an alternative name is desired, then the VECTORS directive may be used as follows

```
vectors low-real /u/rjh/jobs/fe_ether_water.mos
```

13.3.6 Restarting

Restart of ONIOM calculations does not currently work as smoothly as we would like. For geometry optimizations that terminated gracefully by running out of iterations, the restart will work as normal. Otherwise, specify in the input of the restart job the last geometry of the optimization. The Hessian information will be reused and the calculation should proceed losing at most the cost of one ONIOM gradient evaluation. For energy or frequency calculations, restart may not currently be possible.

13.3.7 Examples

13.3.7.1 Hydrocarbon bond energy

A simple two-layer model changing just the wavefunction with one link atom.

This reproduces the two-layer ONIOM (MP2:HF) result from Dapprich et al. for the reaction R-CH_3 ➔ R-CH_2 + H with R=CH_3 using CH_4 as the model. The geometries of R-CH_3 and R-CH_2 are optimized at the DFT-B3LYP/6-311++G** level of theory, and then ONIOM is used to compute the binding energy using UMP2 for the model system and HF for the real system. The results, including MP2 calculations on the full system for comparison, are as given in the table below.

Theory	Me-CH2	Me-Me	H	De(Hartree)	De(kcal/mol)
B3LYP	-79.185062	-79.856575	-0.502256	0.169257	106.2
HF	-78.620141	-79.251701	-0.499817	0.131741	82.7
MP2	-78.904716	-79.571654	-0.499817	0.167120	104.9
MP2:HF	-78.755223	-79.422559	-0.499817	0.167518	105.1

Energies for ONIOM example 1, hydrocarbon bond energy using MP2:HF two-layer model.

The following input first performs a calculation on CH_2-CH_2, and then on CH_3-CH_3. Note that in the second calculation we cannot use the full symmetry since we are breaking the C-C bond in forming the model system (the non-equivalence of the methyl groups is perhaps more apparent if we write R-CH_3).

```
start

basis spherical
  H library 6-311++G**; C library 6-311++G**
end

title "ONIOM Me-CH2"

geometry autosym
  H    -0.23429328     1.32498565     0.92634814
  H    -0.23429328     1.32498565    -0.92634814
  C    -0.13064265     0.77330370     0.00000000
  H    -1.01618703    -1.19260361     0.00000000
  H     0.49856072    -1.08196901    -0.88665533
  H     0.49856072    -1.08196901     0.88665533
  C    -0.02434414    -0.71063687     0.00000000
end

scf; uhf; doublet; thresh 1e-6; end
mp2; freeze atomic; end

oniom
  high mp2
  low  scf
  model 3   3 7 0.724
end

task oniom

title "ONIOM Me-Me"

geometry   # Note cannot use full D3D symmetry here, either specify noautosym, or change an atom tag (here C -> C1)
  H   -0.72023641     0.72023641    -1.16373235
  H    0.98386124     0.26362482    -1.16373235
  H   -0.26362482    -0.98386124    -1.16373235
  C    0.00000000     0.00000000    -0.76537515
  H    0.72023641    -0.72023641     1.16373235
  H   -0.98386124    -0.26362482     1.16373235
  H    0.26362482     0.98386124     1.16373235
  C1   0.00000000     0.00000000     0.76537515
end

scf; rhf; singlet; end

oniom
  high mp2
  low  scf
  model 4   4 8 0.724
end

task oniom
```

13.3.7.2 Optimization and frequencies

A two-layer model including modification of theory, basis, ECP and total charge and no link atoms.

This input reproduces the ONIOM optimization and vibrational frequency calculation of $Rh(CO)_2Cp$ of Dapprich et al. The model system is $Rh(CO)_2^+$. The low theory is the Gaussian LANL2MB model (Hay-Wadt n+1 ECP with minimal basis on Rh, STO-3G on others) with SCF. The high theory is the Gaussian LANL2DZ model (another Hay-Wadt ECP with a DZ basis set on Rh, Dunning split valence on the other atoms) with DFT/B3LYP. Note that different names should be used for the basis set and ECP since the same mechanism is used to store them in the database.

```
start

ecp LANL2DZ_ECP
  rh library LANL2DZ_ECP
end

basis LANL2DZ spherical
  rh library LANL2DZ_ECP
  o library SV_(Dunning-Hay); c library SV_(Dunning-Hay); h library SV_(Dunning-Hay)
end

ecp Hay-Wadt_MB_(n+1)_ECP
  rh library Hay-Wadt_MB_(n+1)_ECP
end

# This is the minimal basis used by Gaussian.  It is not the same
# as the one in the EMSL basis set library for this ECP.
basis Hay-Wadt_MB_(n+1) spherical
  Rh s; .264600D+01 -.135541D+01; .175100D+01  .161122D+01; .571300D+00  .589381D+00
  Rh s; .264600D+01  .456934D+00; .175100D+01 -.595199D+00; .571300D+00 -.342127D+00
       .143800D+00  .410138D+00; .428000D-01  .780486D+00
  Rh p; .544000D+01 -.987699D-01; .132900D+01  .743359D+00; .484500D+00  .366846D+00
  Rh p; .659500D+00 -.370046D-01; .869000D-01  .452364D+00; .257000D-01  .653822D+00
  Rh d; .366900D+01  .670480D-01; .142300D+01  .455084D+00; .509100D+00  .479584D+00
       .161000D+00  .233826D+00
  o  library sto-3g; c  library sto-3g; h  library sto-3g
end

charge 0
geometry autosym
  rh       0.00445705   -0.15119674    0.00000000
  c       -0.01380554   -1.45254070    1.35171818
  c       -0.01380554   -1.45254070   -1.35171818
  o       -0.01805883   -2.26420212    2.20818932
  o       -0.01805883   -2.26420212   -2.20818932
  c        1.23209566    1.89314720    0.00000000
  c        0.37739392    1.84262319   -1.15286640
  c       -1.01479160    1.93086461   -0.70666350
  c       -1.01479160    1.93086461    0.70666350
  c        0.37739392    1.84262319    1.15286640
  h        2.31251453    1.89903673    0.00000000
  h        0.70378132    1.86131979   -2.18414218
  h       -1.88154273    1.96919306   -1.35203550
  h       -1.88154273    1.96919306    1.35203550
  h        0.70378132    1.86131979    2.18414218
end

dft; grid fine; convergence gradient 1e-6 density 1e-6; xc b3lyp; end
scf; thresh 1e-6; end

oniom
  low scf basis Hay-Wadt_MB_(n+1) ecp  Hay-Wadt_MB_(n+1)_ECP
  high dft basis LANL2DZ ecp LANL2DZ_ECP
  model 5 charge 1
  print low
end

task oniom optimize
task oniom freq
```

13.3.7.3 A three-layer example

A three layer example combining CCSD(T), and MP2 with two different quality basis sets, and using multiple link atoms.

The full system is tetra-dimethyl-amino-ethylene (TAME) or $(N(Me)_2)_2$-C=C-$(N(Me)_2)_2$. The intermediate system is $(NH_2)_2$-C=C-$(NH_2)_2$ and H2C=CH2 is the model system. CCSD(T)+aug-cc-pvtz is used for the model region, MP2+aug-cc-pvtz for the intermediate region, and MP2+aug-cc-pvdz for everything.

In the real geometry the first two atoms (C, C) are the model system (link atoms will be added automatically). The first six atoms (C, C, N, N, N, N) describe the intermediate system (again with link atoms to be added automatically). The atoms have been numbered using comments to make the bonding input easier to generate.

To make the model system, four C-N bonds are broken between the ethylene fragment and the dimethyl-amino groups and replaced with C-H bonds. To make the intermediate system, eight C-N bonds are broken between the nitrogens and the methyl groups and replaced with N-H bonds. The scaling factor could be chosen differently for each of the bonds.

```
start

geometry
  C  0.40337795 -0.17516305 -0.51505208  # 1
  C -0.40328664  0.17555927  0.51466084  # 2
  N  1.87154979 -0.17516305 -0.51505208  # 3
  N -0.18694782 -0.60488524 -1.79258692  # 4
  N  0.18692927  0.60488318  1.79247594  # 5
  N -1.87148219  0.17564718  0.51496494  # 6
  C  2.46636552  1.18039452 -0.51505208  # 7
  C  2.48067731 -1.10425355  0.46161675  # 8
  C -2.46642715 -1.17982091  0.51473105  # 9
  C -2.48054940  1.10495864 -0.46156202  # 10
  C  0.30027136  0.14582197 -2.97072148  # 11
  C -0.14245927 -2.07576980 -1.96730852  # 12
  C -0.29948109 -0.14689874  2.97021079  # 13
  C  0.14140463  2.07558249  1.96815181  # 14
  H  0.78955302  2.52533887  1.19760764
  H -0.86543435  2.50958894  1.88075113
  ... and 22 other hydrogen atoms on the methyl groups
end

basis aug-cc-pvtz spherical
  C library aug-cc-pvtz; H library aug-cc-pvtz
end

basis aug-cc-pvdz spherical
  C library aug-cc-pvtz; H library aug-cc-pvtz
end

oniom
  high ccsd(t) basis aug-cc-pvtz
  medium mp2 basis aug-cc-pvtz
  low mp2 basis aug-cc-pvdz
  model 2   1 3  0.87   1 4  0.87   2 5  0.87   2 6  0.87

  inter 6   3 7  0.69   3 8  0.69   4 11 0.69   4 12 0.69 \
          5 13 0.69   5 14 0.69   6 9  0.69   6 10 0.69
end

task oniom
```

13.3.7.4 DFT with and without charge fitting

Demonstrates use of general input strings.

A two-layer model for anthracene (a linear chain of three fused benzene rings) using benzene as the model system. The high-level theory is DFT/B3LYP/TZVP with exact Coulomb. The low level is DFT/LDA/DZVP2 with charge fitting.

Note the following.

1. The semi-colons and quotation marks inside the input string must be quoted with backslash.
2. The low level of theory sets the fitting basis set and the high level of theory unsets it.

```
start
geometry
  symmetry d2h
  C    0.71237329    -1.21458940     0.0
  C   -0.71237329    -1.21458940     0.0
  C    0.71237329     1.21458940     0.0
  C   -0.71237329     1.21458940     0.0
  C   -1.39414269     0.00000000     0.0
  C    1.39414269     0.00000000     0.0
  H   -2.47680865     0.00000000     0.0
  H    2.47680865     0.00000000     0.0
  C    1.40340535    -2.48997027     0.0
  C   -1.40340535    -2.48997027     0.0
  C    1.40340535     2.48997027     0.0
  C   -1.40340535     2.48997027     0.0
  C    0.72211503     3.64518615     0.0
  C   -0.72211503     3.64518615     0.0
  C    0.72211503    -3.64518615     0.0
  C   -0.72211503    -3.64518615     0.0
  H    2.48612947     2.48094825     0.0
  H    1.24157357     4.59507342     0.0
  H   -1.24157357     4.59507342     0.0
  H   -2.48612947     2.48094825     0.0
  H    2.48612947    -2.48094825     0.0
  H    1.24157357    -4.59507342     0.0
  H   -1.24157357    -4.59507342     0.0
  H   -2.48612947    -2.48094825     0.0
end

basis small
  h library DZVP_(DFT_Orbital)
  c library DZVP_(DFT_Orbital)
end

basis fitting
  h library DGauss_A1_DFT_Coulomb_Fitting
  c library DGauss_A1_DFT_Coulomb_Fitting
end

basis big
  h library TZVP_(DFT_Orbital)
  c library TZVP_(DFT_Orbital)
end

oniom
  model 8   1 9 0.75   2 10 0.75   3 11 0.75   4 12 0.75
  high dft basis big   input "unset "cd basis"; dft; xc b3lyp; end"
  low  dft basis small input "set "cd basis" fitting; dft; xc; end"
end

task oniom
```

13.4 QMMM

13.4.1 QMMM Introduction

The combined quantum mechanical molecular mechanics (QM/MM) approach provides a simple and effective tool to study localized molecular transformations in large scale systems such as those encountered in solution chemistry or enzyme catalysis. In this method an accurate but computationally intensive quantum mechanical (QM) description is only used for the regions where electronic structure transformations are occurring (e.g. bond making and breaking). The rest of the system, whose chemical identity remains essentially the same, is treated at the approximate classical molecular mechanics (MM) level.

The QM/MM module in NWChem is built as a top level interface between the classical MD module and various QM modules,managing initialization, data transfer, and various high level operations. The size of the system (10^3 - 10^5 atoms) and the need for classical force field parameters precludes description of the system through just the geometry input block as would be done in pure QM simulations.
Instead a separate preparation stage is required. In a typical setting this preparation run will be done separately from the main QM/MM simulations resulting in the generation of topology and restart files. The topology file contains a list of all relevant force field interactions encountered in the system but has no information about the actual atom positions. Typically the topology file will be generated once and reused throughout the entire simulation. The actual structural information about

the system is contained in the restart file, which will be changing as the system coordinates are updated during the course of the simulation.

Once restart and topology files are generated, the QM/MM simulation can be initiated by defining the specifics of the QM and MM descriptions, and if necessary QM/MM interface parameters.

The actual QM/MM calculation is invoked with the following task directive.

```
task qmmm <string qmtheory> <string operation> [numerical] [ignore]
```

where qmtheory specifies quantum method for the calculation of the quantum region. It is expected that most of QM/MM simulations will be performed with with HF, DFT. or CC theories, but any other QM theory supported by NWChem should also work. NWChem supports wide range of QM/MM tasks including

- single point energy/gradient calculations
- properties
- ESP charge analysis
- optimization and transition states
- hessians and frequency
- reaction pathway calculations
- dynamics
- free energy calculations

13.4.2 QMMM Simulations

- Introduction
- Topology and Restart Files
  - Prerequisites
  - QM region definition
  - Solvation
  - Permanent Constraints
- Input File
  - QM Parameters
  - MM Parameters
  - QM/MM Parameters
- Single Point Calculations
  - Ground State Energy and Gradient
  - Excited State Energy
  - Properties
  - ESP Charge Analysis
- Potential Energy Surface Analysis
  - Optimization
  - Transition States
  - Hessians and Frequency
  - Reaction Pathway Calculations with NEB

- Dynamics
- Free Energy Calculations
- Appendix
  - Format of NWChem parameter file
  - Conversion from AMBER program parameter files to NWChem
- References

13.4.3 QMMM Restart and Topology Files

The structure of the system in QM/MM calculations is provided by the restart file (extension .rst) and not by the geometry block as it would be for pure QM calculations. The parameters of classical interaction are given by the topology file (extension .top). These two files are **REQUIRED** for QM/MM calculations and can be generated by the prepare module. In a typical setting this “preparation stage” is performed separately from the main QM/MM simulation.

- Prerequisites
- QM region definition
- Solvation
- Permanent Constraints

The necessary prerequisites for the preparation of topology and restart files for QM/MM simulations are: * Properly formatted PDB file for the system. * Fragment Files (extension .frg) * Parameter Files (extension .par)

A number of fragment files as well as standard Amber type force parameter files are provided with the NWChem distribution. However, user should be prepared to generate additional fragment and parameter files for nonstandard cases

One of major required pieces of information that has to be provided in the prepare block for QM/MM simulations is the definition of the QM region. This can be accomplished using *modify* directive used either per atom

```
modify atom <integer isgm>:<string atomname> quantum
```

or per segment/residue basis

```
modify segment <integer isgm> quantum
```

Here *isgm* and *atomname* refer to the residue number and atom name record as given in the PDB file. It is important to note that that the leading blanks in atom name record should be indicated with underscores. Per PDB format guidelines the atom name record starts at column 13. If, for example, the atom name record “OW” starts in the 14th column in PDB file, it will appear as “_OW” in the modify atom directive in the prepare block.

In the current implementation only solute atoms can be declared as quantum. If part of the solvent has to be treated quantum mechanically then it has to redeclared to be solute. The definition of QM region should be accompanied by *update lists* and *ignore* directives. Here is an example input file that will generate QM/MM restart and topology files for the ethanol molecule:

```
title "Prepare QM/MM calculation of ethanol"
start etl

[prepare](Prepare)
#--*name of the pdbfile*
  source [etl0.pdb](etl0.pdb)
#--*generate new topology and sequence file*
  new_top new_seq
#--*generate new restart file*
  new_rst
#--*define quantum region (note the use of underscore)*
  modify atom 1:_C1  quantum
  modify atom 1:2H1  quantum
  modify atom 1:3H1  quantum
  modify atom 1:4H1  quantum
  update lists
  ignore
#--*save restart file*
  write etl_ref.rst
#--*generate pdb file*
  write [etl_ref.pdb](etl_ref.pdb)
end
task prepare
```

Running the input shown above will produce (among other things) the topology file (etl.top), the restart file (etl_ref.rst), and the pdb file etl_ref.pdb. The prefix for the topology file follows after the rtdb name specified in the start directive in the input (i.e. “start etl”), while the names for the restart and pdb files were specified explicitly in the input file. In the absence of the explicit write statement for the restart file, it would be generated under the name “etl_md.rst”. The pdb file would only be written in the presence of the explicit write statement.

Tip: *It is strongly recommended to check the correctness of the generated pdb file versus the original “source” pdb file to catch possible errors in the formatting of the pdb and fragment files.*

During the preparation stage of QM/MM calculations the system may also be solvated using `solvate` directive of the prepare module. It is recommended that solvation is performed in conjunction with `center` and `orient` directives.

Here is an example where the ethanol molecule is declared quantum and solvated in a box of spce waters:

```
title "Prepare QM/MM calculation of solvated ethanol"
start etl
prepare
source etl0.pdb
new_top new_seq
new_rst
#center and orient prior to solvation
center
orient
#solvation in 1 nm by 2 nm by 3 nm box
solvate box 1.0 2.0 3.0
#the whole ethanol is declared quantum now
modify segment 1 quantum
update lists
ignore
write etl_ref.rst
write etl_ref.pdb
end
task prepare
```

Fixing atoms outside a certain distance from the QM region can also be accomplished using prepare module. These constraints will then be permanently embedded in the resulting restart file, which may be advantageous for certain types of QM/MM simulations. The actual format for the constraint directive to fix whole residues is

```
fix segments beyond  <real radius>  <integer residue number>:<string atom name>
```

or to fix on atom basis

```
fix atoms beyond  <real radius>  <integer residue number>:<string atom name>
```

This example illustrates the use of permanent fix directives during preparation stage

```
start etl
prepare
source etl0.pdb
new_top new_seq
new_rst
center
orient
#solvation in 40 A cubic box
solvate cube 4.0
modify segment 1 quantum
#fix residues more than 20 A away from ethanol oxygen atom
fix segments beyond 2.0 1:_O
update lists
ignore
write etl_ref.rst
write etl_ref.pdb
end
task prepare
```

13.4.4 QMMM Input File

13.4.4.1 QM/MM Input file

The input file for QM/MM calculations contains definition of molecular mechanics parameters, quantum mechanical parameters, and QM/MM interface parameters.

- QM/MM QM Parameters
- QM/MM MM Parameters
- QM/MM QM/MM Parameters

13.4.4.2 QM_Parameters

The parameters defining calculation of the QM region (including basis sets) must be present in the traditional NWChem input format except for the geometry block.

The geometrical information will be constructed automatically using information contained in the restart file.

The molecular mechanics parameters are given in the form of standard MD input block as used by the MD module. At the basic level the molecular mechanics input block specifies the restart and topology file that were generated during QM/MM preparation stage. It also contains information relevant to the calculation of the classical region (e.g. cutoff distances, constraints, optimization and dynamics parameters, etc) in the system. In this input block one can also set fixed atom constraints on classical atoms. Continuing with our prepare example for ethanol molecule here is a simple input block that may be used for this system.

```
 md
# this specifies that etl_md.rst will be used as a restart file
#  and etl.top will be a topology file
  system etl_md
# if we ever wanted to fix C1 atom
  fix solute 1 _C1
  noshake solute
 end
```

The *noshake solute*, shown in the above example is a recommended directive for QM/MM simulations that involve optimizations. Otherwise user has to ensure that the optimization method for classical solute atoms is a steepest descent

13.4.4.4 QM/MM parameters

The QM/MM interface parameters define the interaction between classical and quantum regions.

```
qmmm
[ [eref] <double precision default 0.0d0>]
[ [bqzone] <double precision default 9.0d0>]
[ [mm_charges] [exclude <(none||all||linkbond||linkbond_H) default none>]
      [ expand  <none||all||solute||solvent> default none]
      [ update  <integer default 0>]
[ [link_atoms] <(hydrogen||halogen) default hydrogen>]
[ [link_ecp]  <(auto||user) default auto>]
[ [region]   < [region1]  [region2]  [region3] > ]
[ [method]   [method1]  [method2]  [method3]  ]
[ [maxiter]  [maxiter1] [maxiter2] [maxiter3] ]
[ [ncycles]  < [number] default 1 > ]
[ [density]  [espfit] [static] [dynamical] ]
[ [xyz]  ]
[ [convergence] <double precision default 1.0d-4>] ]
[ [load] ]
[ [nsamples] ]
end
```

Detailed explanation of the subdirectives in the QM/MM input block is given below:

13.4.4.4.0.1 QM/MM eref

```
eref <double precision default 0.0d0>
```

This directive sets the relative zero of energy for the QM component of the system. The need for this directive arises from different definitions of zero energy for QM and MM methods. In QM methods the zero of energy for the system is typically vacuum. The zero of energy for the MM system is by definition of most parameterized force fields the separated atom energy. Therefore in many cases the energetics of the QM system will likely overshadow the MM component of the system. This imbalance can be corrected by suitably chosen value of `eref`. In most cases *IT IS OK* to leave `eref` at its default value of zero.

13.4.4.4.0.2 QM/MM bqzone

```
bqzone <double precision default 9.0d0>
```

This directive defines the radius of the zone (in angstroms) around the quantum region where classical residues/segments will be allowed to interact with quantum region both electrostatically and through Van der Waals interactions. It should be noted that classical atoms interacting with quantum region via bonded interactions are always included (this is true even if bqzone is set to 0). In addition, even if one atom of a given charged group is in the bqzone (residues are typically treated as one charged group) then the whole group will be included.

13.4.4.4.0.3 QM/MM mm_charges

```
mm_charges [exclude <(none||all||linkbond||linkbond_H) default none>]
      [expand  <none||all||solute||solvent> default none]
      [update  <integer default 0>]
```

This directive controls treatment of classical point (MM) charges that are interacting with QM region. For most QM/MM applications the use of directive will be not be necessary. Its absence would be simply mean that all MM charges within the cuttof distance (as specified by cutoff) as well those belonging to the charges groups directly bonded to QM region will be allowed to interact with QM region.

Keyword `exclude` specifies the subset MM charges that will be specifically excluded from interacting with QM region.

- `none` default value reverts to the original set of MM charges as described above.
- `all` excludes all MM charges from interacting with QM region (“gas phase” calculation).
- `linkbond` excludes MM charges that are connected to a quantum region by at most two bonds,
- `linkbond_H` similar to `linkbond` but excludes only hydrogen atoms.

Keyword expand expands the set MM charges interacting with QM region beyond the limits imposed by cutoff value.

- none default value reverts to the original set of MM charges
- solute expands electrostatic interaction to all solute MM charges
- solvent expands electrostatic interaction to all solvent MM charges
- all expands electrostatic interaction to all MM charges

Keyword update specifies how often list of MM charges will be updated in the course of the calculation. Default behavior is not to update.

13.4.4.4.0.4 QM/MM link_atoms

link_atoms <(hydrogen||halogen) default halogen>

This directive controls the treatment of bonds crossing the boundary between quantum and classical regions. The use of hydrogen keyword will trigger truncation of such bonds with hydrogen link atoms. The position of the hydrogen atom will be calculated from the coordinates of the quantum and classical atom of the truncated bond using the following expression

$$R_{hlink} = (1-g)R_{quant} + g*R_{class}$$

where *g* is the scale factor set at 0.709

Setting link_atoms to halogen will result in the modification of the quantum atom of the truncated bond to the fluoride atom. This fluoride atom will typically carry an effective core potential (ECP) basis set as specified in link_ecp directive.

13.4.4.4.0.5 QM/MM link_ecp

link_ecp <(auto||user)default auto>

This directive specifies ECP basis set on fluoride link atoms. If set toauto the ECP basis set given by Zhang, Lee, Yang[1] for 6-31G* basis will be used. Strictly speaking, this implies the use of 6-31G* spherical basis as the main basis set. If other choices are desired then keyword user should be used and ECP basis set should be entered separatelly following the format given in section dealing with ECPs . The name tag for fluoride link atoms is F_L .

13.4.4.4.0.6 QM/MM region

region < [region1] [region2] [region3] >

This directive specifies active region(s) for optimization, dynamics, frequency, and free energy calculations. Up to three regions can be specified, those are limited to

- qm - all quantum atoms
- qmlink - quantum and link atoms
- mm_solute - all classical solute atoms excluding link atoms
- solute - all solute atoms including quantum
- solvent all solvent atoms
- mm all classical solute and solvent atoms, excluding link atoms
- all all atoms

Only the first region will be used in dynamics, frequency, and free energy calculations. In the geometry optimizations, all three regions will be optimized using the following optimization methods

```
if (region.eq."qm") then
  method = "bfgs"
 else if (region.eq."qmlink") then
  method = "bfgs"
 else if (region.eq."mm_solute") then
  method = "lbfgs"
 else if (region.eq."mm") then
  method = "sd"
 else if (region.eq."solute") then
  method = "sd"
 else if (region.eq."solvent") then
  method = "sd"
 else if (region.eq."all") then
  method = "sd"
 end if
```

where “bfgs” stands for Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization method, “lbfgs” limited memory version of quasi-newton, and “sd” simple steepest descent algorithm. These assignments can be potentially altered using method directive.

13.4.4.4.0.7 QM/MM method

```
method   [method1]  [method2]  [method3]
```

This directive controls which optimization algorithm will be used for the regions as defined by [[qmmm_region|Qmmm_region]] directive. The allowed values are `bfgs` aka DRIVER, `lbfgs` limited memory version of quasi-newton, and `sd` simple steepest descent algorithm. The use of this directive is not recommended in all but special cases. In particular, `bfgs` should be used for QM region if there are any constraints, `sd` method should always be used for classical solute and solvent atoms with shake constraints.

13.4.4.4.0.8 QM/MM maxiter

```
maxiter  [maxiter1] [maxiter2] [maxiter3]
```

This directive controls maximum number of iterations for the optimizations of regions as defined by by regions directive. User is strongly encouraged to set this directive explicitly as the default value shown below may not be appropriate in all the cases:

```
if(region.eq."qm") then
  maxiter = 20
 else if (region.eq."qmlink") then
  maxiter = 20
 else if (region.eq."mm") then
  maxiter = 100
 else if (region.eq."solvent") then
  maxiter = 100
 else
  maxiter = 50
 end if
```

13.4.4.4.0.9 QM/MM ncycles

```
ncycles  < [number] default 1 >
```

This directive controls the number of optimization cycles where the defined regions will be optimized in succession, or number of sampling cycles in free energy calculations.

13.4.4.4.0.10 QM/MM density

```
density  [espfit] [static] [dynamical] default dynamical
```

This directive controls the electrostatic representation of*fixed* QM region during optimization/dynamics of classical regions.

It has no effect when position of QM atoms are changing.

- **dynamical** is an option where QM region is treated the standard way, through the recalculation of the wavefunction calculated and the resulting electron density is used to compute electrostatic interactions with classical atoms. This option is the most expensive one and becomes impractical for large scale optimizations.
- **static** option will not recalculate QM wavefunction but will still use full electron density in the computations of electrostatic interactions.
- **espfit** option will not recalculate QM wavefunction nor it will use full electron density. Instead, a set of ESP charges for QM region will be calculated and used to compute electrostatic interactions with the MM regions. This option is the most efficient and is strongly recommended for large systems.

Note that both “static” and “espfit” options do require the presence of the movecs file. It is typically available as part as part of calculation process. Otherwise it can be generated by running qmmm energy calculation.

In most calculations default value for **density** would **dynamical**, with the exception of free energy calculations where default setting **espfit** will be used.

13.4.4.4.0.11 QM/MM load

```
load < esp > [<filename>]
```

This directive instructs to load external file (located in permanent directory) containing esp charges for QM region. If filename is not provided it will be constructed from the name of the restart file by replacing “.rst” suffix with “.esp”. Note that file containing esp charges is always generated whenever esp charge calculation is performed

13.4.4.4.0.12 QM/MM convergence

```
convergence  < double precision etol default 1.0d-4>
```

This directive controls convergence of geometry optimization. The optimization is deemed converged if absolute difference in total energies between consecutive optimization cycles becomes less than *etol*.

13.4.4.4.0.13 QM/MM nsamples

```
nsamples
```

This directive is required for free energy calculations and defines number of samples for averaging during single cycle.

13.4.4.4.1 References

1. Zhang, Y.; Lee, T.-S.; Yang, W. A Pseudobond Approach to Combining Quantum Mechanical and Molecular Mechanical Methods. *The Journal of Chemical Physics* **1999**, *110* (1), 46–54. https://doi.org/10.1063/1.478083

13.4.5 QMMMM Single Point Calculations

13.4.5.1 QMMM Single Point Calculations

The task directive for QM/MM single point energy and gradient calculations is given by

```
task qmmm <qmtheory> energy
```

or

```
task qmmm <qmtheory> gradient [numerical]
```

where `qmtheory` refers to the level of QM theory (e.g. dft, tce, mp2, …).

The ground state QM/MM energy calculations should be possible with all QM descriptions available in NWChem, however most of testing was performed using core QM methods (scf,dft,mp2,tce). The ground state QM/MM gradient calculations can be performed analytically with scf,dft,mp2 levels of theory and numerically for all the others.

The relevant settings for QM/MM interface block for energy and gradient calculations include

- bqzone
- mm_charges
- link_atoms
- link_ecp.

qmmm_example3

The excited state QM/MM energy calculations can be performed with TCE

```
task qmmm tce energy
```

or TDDFT

```
task qmmm tddft energy
```

levels of theory. The excited state QM/MM gradient energy calculations can be performed only numerically.

13.4.5.3 QM/MM properties

A number of electronic structure properties can be calculated with QM/MM using capabilities provided by property, esp, and dplot modules.

The example below illustrates dipole property QM/MM DFT/B3LYP calculation for quantum water molecule embedded into 20 angstrom box of classical SPCE/E water molecules.

The preparation stage that involves definition of the QM region and solvation is performed as part of the calculation. Note that water fragment file wtr.frg is required in this calculation. Prepare run will generate restart file (wtr_ref.rst) and topology file (wtr.top)

In the QM/MM interface block the use of `bq_zone` value of 3.0 Angstrom is specified.

```
start wtr

permanent_dir ./perm
scratch_dir ./data

prepare
source wtr0.pdb
new_top new_seq
new_rst
modify segment 1  quantum
center
orient
solvate box 3.0
update lists
ignore
write wtr_ref.rst
write wtr_ref.pdb
end

task prepare

md
system wtr_ref
end

basis
* library "6-31G"
end

dft
xc b3lyp
end

qmmm
bq_zone 3.0
end

property
 dipole
end

task qmmm dft property
```

- Example QM/MM ESP Calculation:

The example below illustrates dipole property QM/MM DFT/B3LYP calculation for quantum water molecule embedded into 20 angstrom box of classical SPCE/E water molecules.

The preparation stage that involves definition of the QM region and solvation is performed as part of the calculation. Note that water fragment file wtr.frg is required in this calculation. Prepare run will generate restart file (wtr_ref.rst) and topology file (wtr.top)

In the QM/MM interface block the use of bq_zone value of 3.0 Angstrom is specified.

```
start wtr

permanent_dir ./perm
scratch_dir ./data

prepare
source wtr0.pdb
new_top new_seq
new_rst
modify segment 1  quantum
center
orient
solvate box 3.0
update lists
ignore
write wtr_ref.rst
write wtr_ref.pdb
end

task prepare

md
system wtr_ref
end

basis
* library "6-31G"
end

dft
xc b3lyp
end

qmmm
bq_zone 3.0
end

property
 dipole
end

task qmmm dft property
```

13.4.6 QMMM Potential Energy Surface Analysis

QM/MM optimization is based on multi-region optimization methodology and is invoked by

```
task qmmm <qmtheory> optimize
```

The overall algorithm involves alternating optimizations of QM and MM regions until convergence is achieved. This type of approach offers substantial savings compared to direct optimization of the entire system as a whole. In the simplest case of two regions (QM and MM) the algorithm is comprised of the following steps:

1. Optimization of the QM region keeping MM region fixed
2. Calculation of reduced electrostatic representation for the QM region (e.g. ESP charges)
3. Optimization of MM region keeping QM region fixed
4. Repeat from Step 1 until converged

The optimization process is controlled by the following keywords:

- **region** - **required** keyword which specifies which regions will optimized and in which order.
- **maxiter** - number of optimizations steps for each region within single optimization pass
- **ncycles** - number of optimization cycles
- **density** - electrostatic representation of the QM region during MM optimization
- **xyz** - output of xyz structure files
- **convergence** - convergence criteria

Here is an example QM/MM block that provides practical illustration of all these keywords for a generic optimization case where QM molecule(s) are embedded in the solvent

```
qmmm
  [region] qm   solvent
  [maxiter] 10   3000
  [ncycles] 5
  [density] espfit
  [xyz] foo
end
```

We have two regions in the system “qm” and “solvent” and we would like to optimize them both, thus the line

```
[region] qm   solvent
```

Our QM region is presumably small and the maximum number of iterations (within a single optimization pass) is set to 10. The solvent region is typically much larger (thousands of atoms) and the maximum number of iterations is set to a much large number 3000:

```
[maxiter] 10   3000
```

We would like to perform a total of 5 optimization passes, giving us a total of 5*10=50 optimization steps for QM region and 5*3000=15000 optimization steps for solvent region:

```
[ncycles] 5
```

We are requesting QM region to be represented by point ESP charges during the solvent optimization:

```
[density] espfit
```

Finally we are requesting that the coordinates of the first region to be saved in the form of numbered xyz files:

```
[xyz] foo
```

QM/MM transition states calculations for qm or qmlink regions can be performed using

```
task qmmm  saddle
```

The overall algorithm is very similar to QM/MM optimization calculations, but instead of optimization, transition state search will be performed for qm or qmlink region for specified number of steps (as defined bymaxiter keyword). The remaining classical regions (if any) will be optimized following the standard optimization protocol, which may involve, if specified, ESP charge representation of the QM atoms (a recommended option).

The success transition state calculations is strongly dependent on the initial guess. User may consider generation of the latter using QM/MM reaction pathway calculation. Another useful strategy involves precalculation of the Hessian. Following the example presented above one could have precalculated numerical Hessian for the qm region

```
...
qmmm
region qm
end

freq
animate
end

task qmmm dft freq
```

and then used this information in the TS calculation

```
...
```

```
driver
clear
inhess 2  #read in hessian from perm directory
moddir 1  #follow 1st mode
end

qmmm
bqzone 15.0
region  qm solvent
xyz  ts
maxiter 10 1000
ncycles 2
density espfit
end

task qmmm dft saddle
```

13.4.6.3 QM/MM hessian and frequency calculations

13.4.6.3.0.1 Setup

QM/MM hessian and frequency calculations are invoked though the following task directives

```
task qmmm `<qmtheory>` hessian
```

or

```
task qmmm `<qmtheory>` freq
```

Only numerical implementation are supported at this point and will be used even in the absence of “numerical” keyword. Other than standard QM/MM directives no additional QM/MM input is required for default hessian/frequency for all the QM atoms. Using region keyword(first region only), hessian/frequency calculations can also be performed for classical solute atoms. If only classical atoms are involved density keyword can be utilized to enable frozen density or ESP charge representation for fixed QM region. Hessian/frequency calculations for solvent are not possible.

13.4.6.3.0.2 Examples

Example of QM/MM frequency calculation for classical region

This example illustrates QM/MM frequency calculation for Ala-Ser-Ala system. In this case instead of default QM region (see prepare block), the calculation is performed on classical solute part of the system as defined by region directive in QM/MM block. The electrostatic field from fixed QM region is represented by point ESP charges (see density directive). These ESP charges are calculated from wavefunction generated as a result of energy calculation.

```
memory total 800 Mb

start asa

permanent_dir ./perm
scratch_dir ./data

#this will generate topology file ([asa.top](asa.top)), restart ([asa_ref.rst](asa_ref.rst)), and pdb ([asa_ref.pdb](asa_ref.pdb)) files.
prepare
  source [asa.pdb](asa.pdb)
  new_top new_seq
  new_rst
  modify atom 2:_CB quantum
  modify atom 2:2HB quantum
  modify atom 2:3HB quantum
  modify atom 2:_OG quantum
  modify atom 2:_HG quantum
  center
  orient
  solvate
  update lists
  ignore
  write [asa_ref.rst](asa_ref.rst)
  write [asa_ref.pdb](asa_ref.pdb)   # Write out PDB file to check structure
end
task prepare

md
  system asa_ref
end

basis "ao basis"
  * library "6-31G*"
end

dft
 print low
 iterations 500
end

qmmm
region mm_solute
density espfit
end

# run energy calculation to generate wavefunction file for subsequent ESP charge generation
task qmmm dft energy
task qmmm dft freq
```

Experimental implementation of Nudged Elastic Band (NEB) method is available for reaction pathway calculations with QM/MM. The actual pathway/beads construction involves (by default) only the region containing QM and link atoms (referred to as qmlink). The rest of the system plays a passive role and is quenched/optimized each time a gradient on a bead is calculated.

The initial guess for NEB pathway can be generated using geometries of the starting and ending point provided by the .rst files. These are set in the input using the following directive

```
set qmmm:neb_path_limits xxx_start.rst xxx_end.rst
```

where xxx_start.rst xxx_end.rst refers to starting and ending point of the NEB pathway. Both rst files have to be present at the **top level** directory. It should be noted that only coordinates of qmlink region will be used from these two files. The initial coordinates for the rest of the system come from reference rst file provided in the MD block

```
md
  system xxx_ref
  ...
end
```

Typically this reference restart file (xxx_ref.rst) would be a copy of a restart file for starting or ending point.

The number of beads in the NEB pathway, initial optimization step size, and number of optimization steps are set using the following directives

```
set neb:nbeads 10
set neb:stepsize 10
set neb:steps 20
```

The calculation starts by constructing initial guess for the pathway (consisting of a sequence of numbered rst files) by combining linearly interpolated coordinates of the qmlink regions from starting and ending rst files and classical coordinates from the reference file. Next phase involves calculation of the gradients on qmlink region atoms for each of the beads. This involves two steps. First classical region around the qmlink region is relaxed following standard QM/MM optimization protocol. Aside the fact that optimization region cannot be qmlink, all other optimization directives apply and should be set in the QM/MM block following standard convention, e.g.

```
qmmm
 region  solvent
 maxiter 1000
 ncycles 1
 density espfit
end
```

or

```
qmmm
 region  mm_solute solvent
 maxiter 300         1000
 ncycles 3
 density espfit
end
```

In both examples presented above we utilized espfit option for density to speed up calculations.**Note that optimization region cannot be qmlink!**

After the optimization has been performed the gradient on qmlink region is calculated. The procedure is repeated for all the beads. After that the bead coordinates will be advanced following NEB protocol and the entire cycle will be repeated again.

In addition to interpolated initial guess, one can also specify custom initial path represented by numbered sequence of restart files stored in the **perm** directory. This behavior will be triggered automatically in the absence of *qmmm:neb_path_limits* directive. The default naming of the custom initial path is of the formXXX.rst, where is the prefix of reference restart file as defined in MD block and XXX is the 3-digit integer counter with zero blanks (001,002, …, 010, 011, ..). If needed the prefix for the custom initial path can be adjusted using

```
set qmmm:neb_path
```

The progress of NEB calculation can be monitored by

```
grep gnorm <output file>
```

Experience shows that the value of gnorm less or around O(10^-2) indicates converged pathway. The current pathway in the XYZ format can be found in the output file (look for XYZ FILE string) and viewed as animation in some of the molecular viewers (e.g. JMOL)

The entire example directory including the output file can be downloaded fromhere.

Dynamical simulations within QM/MM framework can be initiated using

```
task qmmm  <qmtheory>  dynamics
```

directive. User has to specify the region for which simulation will performed. If dynamics is performed only for the classical parts of the system (QM region is fixed) then ESP point charge representation (density espfit) is recommended to speed up simulations. If this option is utilized then wavefunction file (.movecs) has to be available and present in the permanent directory (this can be most easily achieved by running energy calculation prior to dynamics).

13.4.8 QMMM Free Energy

13.4.8.1 Overview

Free energy capabilities of QM/MM module are at this point restricted to calculations of free energy differences between two fixed configurations of the QM region.

Users must be warned that this the least automated QM/MM functionality containing several calculation stages. Solid understanding of free energy calculations is required to achieve a meaningful calculation.

Description of the implemented methodology can be found inthe following paper. In this approach the free energy difference between the two configurations of the QM region (e.g. A and B):

$\Delta W_{A\to B}=-1/\beta \ln \langle (e^{-\beta (E_B-E_A)}) \rangle_{A}$

is approximated as a sum of internal QM contribution and solvation:

$\Delta W_{A\to B}\approx\Delta W_{A\to B}^{int}+\Delta W_{A\to B}^{solv}$

It is presumed that structures of A and B configurations are available as restart files sharing**common** topology file.

13.4.8.2 Internal contribution

The internal QM contribution is given by the differences in the internal QM energies evaluated at the**optimized** MM environment:

$\Delta W_{A\to B}^{int}=E_{B}^{int}-E_{A}^{int}$

The internal QM energy is nothing more but a gas phase expression total energy but evaluated with the wavefunction obtained in the presence of the environment. To calculate internal QM contribution to free energy difference one has to

1. Optimize MM environment for each configuration
2. Perform total energy calculation for each configuration
3. Calculate internal energy difference

Note that internal QM energy can be found in the QM/MM output file under “quantum energy internal” name.

13.4.8.3 Solvation contribution

The solvation contribution is evaluated by averaging energy difference between A and B configurations of the QM system represented by a set of ESP charges.

$\Delta W_{A\to B}^{solv}=-1/\beta \ln \langle (e^{-\beta (E_B^{ESP}-E_A^{ESP})}) \rangle_{A}$

where E_A^{ESP} is the total energy of the system where QM region is replaced by a set of fixed point ESP charges.

In majority of cases the A and B configuration are “too far apart” and one step free energy calculation as shown above will not lead to meaningful results. One solution is to introduce intermediate points that bridge A and B configurations by linear interpolation

$R_{\lambda_i} = (1-\lambda_i)R_A + \lambda_i R_B$

$Q_{\lambda_i} = (1-\lambda_i)Q_A + \lambda_i Q_B$

where

$\lambda_i = \frac {i}{n}, \quad i=0,..,n\,\!$

The solvation free energy difference can be then written as sum of differences for the subintervals$\,\!$

[\lambda_i\to\lambda_{i+1}]\) :

\(\Delta W_{A\to B}^{\rm esp} = \sum_{i=0}^{n}\Delta W_{\lambda_i\to\lambda_{i+1}}^{\rm esp} \)

To expedite the calculation it is convenient to use a double wide sampling strategy where the free energy differences for the intervals \(\,\! [\lambda_{i-1}\to\lambda_{i}]\) and \(\,\! [\lambda_i\to\lambda_{i+1}]\) are calculated simultaneously by sampling around \(\,\!\lambda_{i}\) point. In the simplest case where we use two subintervals (n=2)

\(\Delta W_{A\to B}^{solv} \equiv \Delta W_{0\to 1}= \Delta W_{0\to 0.5}^{solv}+\Delta W_{0.5\to 1}^{solv}\)

or

\(\Delta W_{A\to B}^{\rm solv} = -\Delta W_{0.5\to 0}^{\rm solv}+\Delta W_{0.5 \to 1}^{\rm solv}\)

The following items are necessary:

1. Restart file corresponding to either A or B configuration of the QM region (sharing the same topology file)
2. ESP charges for QM region in .esp format for both configurations
3. Coordinates for QM regions in .xyzi format for both configurations

Both .esp and .xyzi files would be typically obtained during the calculation of internal free energy (see above). ESP charges would be generated in the perm directory during optimization of the MM region. The xyzi is basically xyz structure file with an extra column that allows to map coordinates of QM atoms to the overall system. The xyzi file can also be obtained as part of calculation of internal free energy by inserting

```
set qmmm:region_print .true.
```

anywhere in the input file during energy calculation.**Both xyzi and esp files should be placed into the perm directory!!!**

In the input file the restart file is specified in the MD block following the standard notation

```
md
 system < name of rst file without extension>
 ...
end
```

while coordinates of QM region (xyzi files) and ESP charges (esp files) are set using the following directives (at the top level outside of any input blocks)

```
set qmmm:fep_geom xxx_A.xyzi xxx_B.xyzi
set qmmm:fep_esp  xxx_A.esp xxx_B.esp
```

The current interpolation interval \(\,\! [\lambda_i\to\lambda_{i+1}]\)
for which free energy difference is calculated is defined as

```
set qmmm:fep_lambda lambda_i lambda_i+1
```

To enable double wide sampling use the following directive

```
set qmmm:fep_deriv .true.
```

If set, the above directive will perform both \(\,\! [\lambda_i\to\lambda_{i+1}]\) and \(\,\! [\lambda_i\to\lambda_{i-1}]\) calculations, where

\(\,\! \lambda_{i-1}=\lambda_i - (\lambda_{i+1}-\lambda_i)\)

The calculation proceeds in cycles, each cycle consisting of two phases. First phase is generation of classical MD trajectory around \(\,\! \lambda_i\) point. Second phase is processing of the generated trajectory to calculate averages of relevant energy differences. The number of MD steps in the first phase is controlled by the QM/MM directive

- **nsamples**

This is a **required** directive for QM/MM free energy calculations.

Number of overall cycles is defined by the QM/MM directive

- **ncycles**

In most cases explicit definition of QM/MM **density** and **region** should not be required. The QM/MM **density** will automatically default to **espfit** and **region** to **mm**.

Prior to data collection for free energy calculations user may want to prequilibrate the system, which can be achieved by **equil** keyword in the MD block:

```
md
 ...
 equil <number of equilibration steps>
end
```

Other parameters (e.g. temperature and pressure can be also set in the MD block.

The actual QM/MM solvation free energy calculation is invoked through the following task directive

```
task qmmm fep
```

The current value of solvation free energy differences may be tracked though

```
grep free <name of the output file>
```

The first number is a forward (\(\,\! [\lambda_i\to\lambda_{i+1}]\)) free energy difference and second number is backward (\(\,\! [\lambda_i\to\lambda_{i-1}]\)) free energy difference, both in kcal/mol. The same numbers can also be found in the 4th and 6th columns of .thm file but this time in **atomic units**.

The same .thm file can also be used to continue from the prior calculation. This will require the presence of

```
set qmmm:extend .true.
```

directive, the .thm file, and the appropriate rst file.

Here is an example of the input file for QM/MM solvation free energy calculation

13.4.9 QMMM Apppendix

13.4.9.1 Preparing QM/MM calculations from scratch

Setting up QM/MM calculations for a new system for which classical force analog is not readily available would typically involve the following steps

1. Generation of the proper PDB file
2. Construction the fragment file(s)
3. Creation of parameter file if new atom types were defined

13.4.9.2 Generation of the proper PDB file

It is often the case that the input structure for the system comes in the form of the xyz file. Let us take a concrete example of N3O3- molecule, which we would like to embed in classical solvent and perform QM/MM calculations. Here is the structure of just N3O3- in xyz format (generated in course of gas phase optimizations)

```
 6
geometry
N              1.31562667    0.93574165   -0.42424728
O              1.56161766    0.18015298   -1.36827899
N              2.36373381    1.02559495    0.48834956
O              3.47240000    0.42852552    0.42137570
N              1.95804013    1.90608355    1.48799418
O              2.81393172    2.06788134    2.36142683
```

We cannot use this file as is in the QM/MM simulations, and it has to be converted into PDB format. This is needed even if we plan to treat this molecule quantum mechanically. There is more than way to do it. For example, we could use Babel https://openbabel.org, which will generate the PDB file as

```
COMPND    geometry
AUTHOR    GENERATED BY OPEN BABEL 2.3.0
HETATM    1  O   LIG     1       1.562   0.180  -1.368  1.00  0.00          O
HETATM    2  N   LIG     1       1.316   0.936  -0.424  1.00  0.00          N
HETATM    3  N   LIG     1       2.364   1.026   0.488  1.00  0.00          N
HETATM    4  O   LIG     1       3.472   0.429   0.421  1.00  0.00          O
HETATM    5  N   LIG     1       1.958   1.906   1.488  1.00  0.00          N
HETATM    6  O   LIG     1       2.814   2.068   2.361  1.00  0.00          O
CONECT    1    2
CONECT    2    1    3
CONECT    3    2    4    5
CONECT    4    3
CONECT    5    3    6
CONECT    6    5
MASTER        0    0    0    0    0    0    0    0    6    0    6    0
END
```

which after stripping nonessential information becomes

```
HETATM    1  O   LIG     1       1.562   0.180  -1.368  1.00  0.00          O
HETATM    2  N   LIG     1       1.316   0.936  -0.424  1.00  0.00          N
HETATM    3  N   LIG     1       2.364   1.026   0.488  1.00  0.00          N
HETATM    4  O   LIG     1       3.472   0.429   0.421  1.00  0.00          O
HETATM    5  N   LIG     1       1.958   1.906   1.488  1.00  0.00          N
HETATM    6  O   LIG     1       2.814   2.068   2.361  1.00  0.00          O
END
```

This is not yet the format we want. So what we need to do is

1. *replace HETATM field by ATOM preserving the format (column locations) of the rest of the file*

I would typically use sed for this purpose

```
sed 's/HETATM/ATOM  /' n3o3-bad.pdb > n3o3-step1.pdb
```

where `n3o3-bad.pdb` is the original pdb file from Babel and `n3o3-step1.pdb` is the converted one as shown below

```
ATOM      1  O   LIG     1       1.562   0.180  -1.368  1.00  0.00          O
ATOM      2  N   LIG     1       1.316   0.936  -0.424  1.00  0.00          N
ATOM      3  N   LIG     1       2.364   1.026   0.488  1.00  0.00          N
ATOM      4  O   LIG     1       3.472   0.429   0.421  1.00  0.00          O
ATOM      5  N   LIG     1       1.958   1.906   1.488  1.00  0.00          N
ATOM      6  O   LIG     1       2.814   2.068   2.361  1.00  0.00          O
END
```

1. *define residues (aka molecules):*

In our case Babel did this for us and the entire system is defined as one residue with the name LIG (see columns 4 and 5). We can leave it as, but I will redefine residue name to NN3 (keep it to 3 characters !). Again running `sed 's/LIG/NN3/' n3o3-step1.pdb > n3o3-step2.pdb`

```
ATOM      1  O   NN3     1       1.562   0.180  -1.368  1.00  0.00          O
ATOM      2  N   NN3     1       1.316   0.936  -0.424  1.00  0.00          N
ATOM      3  N   NN3     1       2.364   1.026   0.488  1.00  0.00          N
ATOM      4  O   NN3     1       3.472   0.429   0.421  1.00  0.00          O
ATOM      5  N   NN3     1       1.958   1.906   1.488  1.00  0.00          N
ATOM      6  O   NN3     1       2.814   2.068   2.361  1.00  0.00          O
END
```

You could have also broken it up into several residues as

```
ATOM      1  O   NN2     1       1.562   0.180  -1.368  1.00  0.00           O
ATOM      2  N   NN2     1       1.316   0.936  -0.424  1.00  0.00           N
ATOM      3  N   NN2     1       2.364   1.026   0.488  1.00  0.00           N
ATOM      4  O   NN2     1       3.472   0.429   0.421  1.00  0.00           O
ATOM      5  N   NN1     2       1.958   1.906   1.488  1.00  0.00           N
ATOM      6  O   NN1     2       2.814   2.068   2.361  1.00  0.00           O
END
```

where I defined two residues NN2 and NN1 (note changes in columns 4 and 5). To keep things simple I will stay with one residue version for now

1. *make unique atom names*:

This has to do with the requirement that all atoms names have to be unique within a given residue. So if we take our one residue version it could be modified as (notice column 3)

```
ATOM      1  O1  NN3     1       1.562   0.180  -1.368  1.00  0.00           O
ATOM      2  N1  NN3     1       1.316   0.936  -0.424  1.00  0.00           N
ATOM      3  N2  NN3     1       2.364   1.026   0.488  1.00  0.00           N
ATOM      4  O2  NN3     1       3.472   0.429   0.421  1.00  0.00           O
ATOM      5  N3  NN3     2       1.958   1.906   1.488  1.00  0.00           N
ATOM      6  O3  NN3     2       2.814   2.068   2.361  1.00  0.00           O
END
```

Now we have a PDB file for our system in “proper” PDB format. Before we move to next step, I should mention that the last 3 columns are not necessary and could have been removed at any point leading to n3o3.pdb

```
ATOM      1  O1  NN3     1       1.562   0.180  -1.368
ATOM      2  N1  NN3     1       1.316   0.936  -0.424
ATOM      3  N2  NN3     1       2.364   1.026   0.488
ATOM      4  O2  NN3     1       3.472   0.429   0.421
ATOM      5  N3  NN3     2       1.958   1.906   1.488
ATOM      6  O3  NN3     2       2.814   2.068   2.361
END
```

13.4.9.3 Generation of new fragment files

While setting up QM/MM calculations is often necessary to generate new fragment files for the molecules that are not available as part of standard set. The IMPORTANT assumption here is that these new molecules/residues will be part of OQM region, and as a result only minimum information needs to be provided to include them in QM/MM calculations.

First we must ensure that we have a proper PDB format for our as discussed in the Generation of the proper PDB file section. As a concrete example we will start with N3O3 example that was discussed there as well

```
ATOM      1  O1  NN3     1       1.562   0.180  -1.368
ATOM      2  N1  NN3     1       1.316   0.936  -0.424
ATOM      3  N2  NN3     1       2.364   1.026   0.488
ATOM      4  O2  NN3     1       3.472   0.429   0.421
ATOM      5  N3  NN3     1       1.958   1.906   1.488
ATOM      6  O3  NN3     1       2.814   2.068   2.361
END
```

We have residue named NN3 and therefore looking to construct fragment file `NN3.frg` . The best way to do it is to run a simple prepare job

```
start n3o3

prepare
source n3o3.pdb
new_top new_seq
new_rst
modify segment 1 quantum
update lists
ignore
write n3o3_ref.pdb
write n3o3_ref.rst
end

task prepare
```

This prepare job will necessarily fail because NN3.frg is not yet available:

```
Created fragment                ./NN3.frg_TMP

Unresolved atom types in fragment NN3


 **********
 *   0: pre_mkfrg failed 9999
 **********
```

As part of this process skeleton fragment file (NN3.frg_TMP) will be generated that can be modified into the final correct form. Let us take a look at NN3.frg_TMP

```
# This is an automatically generated fragment file
# Atom types and connectivity were derived from coordinates
# Atomic partial charges are crude estimates
# 00/00/00  00:00:00
#
$NN3
   6   1   1   0
NN3
   1 O1          0   0   0   1   1    0.000000    0.000000
   2 N1          0   0   0   1   1    0.000000    0.000000
   3 N2          0   0   0   1   1    0.000000    0.000000
   4 O2          0   0   0   1   1    0.000000    0.000000
   5 N3          0   0   0   1   1    0.000000    0.000000
   6 O3          0   0   0   1   1    0.000000    0.000000
   1   2
   2   3
   3   4
   3   5
   5   6
```

The main problem with this fragment file is that there are no atom types to be found in column 12. What atom type does is to identify what classical parameters should be assigned to it. Since, as mentioned in the beginning, we are planing to treat this residue/molecule as part of QM region, the only classical information needed is VDW parameters. We will assume that all nitrogens atoms in our molecule can use the same set of parameters, and the same for oxygens. Therefore we will define two atom types NX and OX

```
$NN3
   6   1   1   0
NN3
   1 O1   OX      0   0   0   1   1    0.000000    0.000000
   2 N1   NX      0   0   0   1   1    0.000000    0.000000
   3 N2   NX      0   0   0   1   1    0.000000    0.000000
   4 O2   OX      0   0   0   1   1    0.000000    0.000000
   5 N3   NX      0   0   0   1   1    0.000000    0.000000
   6 O3   OX      0   0   0   1   1    0.000000    0.000000
   1   2
   2   3
   3   4
   3   5
   5   6
```

and rename resulting file as NN3.frg. Please note that the overall format of the fragment file was preserved and atom types were entered starting at column 12. Rerunning the same prepare job moves as a bit further this time

```
modify segment     1 set 0 quantum
Parameter file                 /Users/marat/opt/codes/nwchem-new/src/data/amber_s/amber.par
Parameter file                 /Users/marat/opt/codes/nwchem-new/src/data/amber_q/amber.par
Parameter file                 /Users/marat/opt/codes/nwchem-new/src/data/amber_x/amber.par
Parameter file                 /Users/marat/opt/codes/nwchem-new/src/data/amber_u/amber.par

Undetermined force field parameters

Parameters could not be found for atom type     OX   Q
Parameters could not be found for atom type     NX   Q

**********
*   0: pre_check failed 9999
**********
```

complaining now that atom types OX and NX are not defined. This brings us to the next step of defining new parameter file for our calculation.

13.4.9.4 Generation of new parameter files

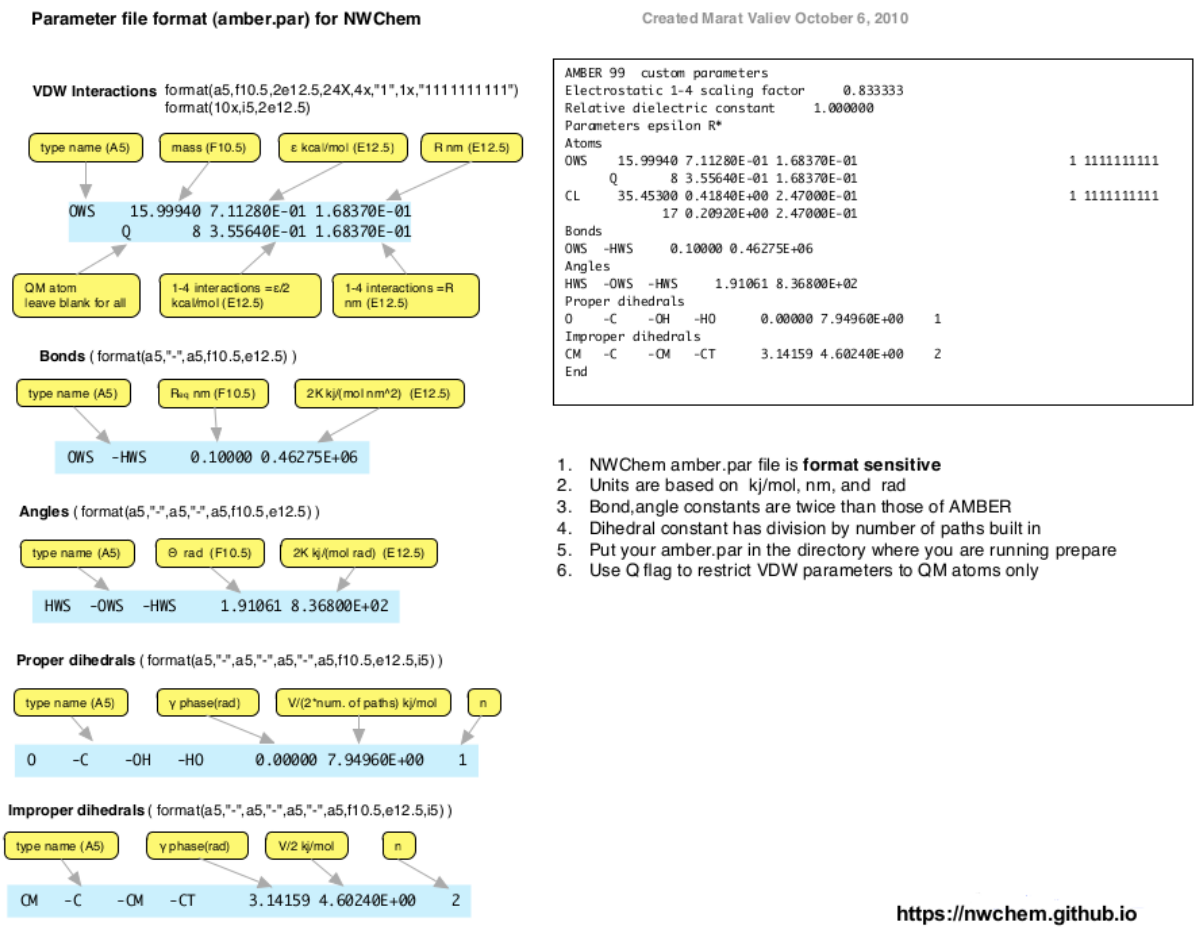
Continuing with our fragment construction inGeneration of new fragment filessection, we now need to define VDW parameters for our new atom types NX and OX. The best way to do it is to create `amber.par` file in the directory where you plan to rerun final prepare

```
Electrostatic 1-4 scaling factor     0.833333
Relative dielectric constant     1.000000
Parameters epsilon R*
#
Atoms
NX     14.01000 7.11280E-01 1.82400E-01                    1 1111111111
     Q       7 3.55640E-01 1.82400E-01
OX     16.00000 6.35968E-01 1.76830E-01                    1 1111111111
     Q       8 3.17984E-01 1.76830E-01
End
```

The format of this file is documented inFormat of NWChem parameter file. How to actually choose the appropriate values for VDW parameters is a whole new subject, which I do not think anybody yet fully addressed. The practical strategy is to copy from known atom types, which are chemically similar to the ones in your system. In the case above I copied parameters from standard AMBER atom types N and OW.

13.4.9.5 Format of NWChem parameter file

The format of NWChem parameter is illustrated on the figure below and also available aspdf file.



13.4.9.6 Conversion of standard AMBER program parameter files

Fortran code that performs conversion from AMBER program parameter file format to NWChem can be foundhere. It works by parsing out free format AMBER style parameter file contained in amber.in

```
MASS
C  12.01
CA    12.01
BOND
#this is a comment
C -CA  469.0    1.409          this is also a comment
C  -   CB     447.0      1.419
ANGLE
C -CA-CA    63.0      120.00 another comment
C -CB-NB    70.0      130.00
DIHEDRAL
X -C -CA-X    4   14.50       180.0           2.        intrpol.bsd.on C6H6
 X - C -  CB -X    4   12.00       180.0           2.        intrpol.bsd.on C6H6
IMPROPER
X -CT-N -CT         1.0         180.          2.          JCC,7,(1986),230
CT -O  - C -OH         10.5         180.          2.
NONBOND
 CA 1.9080  0.0860
C 1.9080  0.0860
```

to **fixed format** NWChem style amber.par file

```
#Generated amber.par file
Electrostatic 1-4 scaling factor     0.833333
Relative dielectric constant     1.000000
Parameters epsilon R*
#
Atoms
CA     12.01000 3.59824E-01 1.90800E-01                        1 1111111111
            6 1.79912E-01 1.90800E-01
C      12.01000 3.59824E-01 1.90800E-01                        1 1111111111
            6 1.79912E-01 1.90800E-01
Bonds
C    -CA      0.14090 3.92459E+05
C    -CB      0.14190 3.74050E+05
Angles
C    -CA   -CA      2.09440 5.27184E+02
C    -CB   -NB      2.26893 5.85760E+02
Proper dihedrals
     -C    -CA   -       3.14159 1.51670E+01    2
     -C    -CB   -       3.14159 1.25520E+01    2
Improper dihedrals
     -CT   -N    -CT      3.14159 4.18400E+00    2
CT   -O    -C    -OH      3.14159 4.39320E+01    2
End
```

13.5 Point charges

13.5.1 Overview

The Bq module provides a way to perform QM calculations in the presence of point charges or Bq's, (as typically referred to in quantum chemistry community). Using Bq module versus geometry block is a recommended way to include point charges in your calculations, in particular if number of charges are big.

13.5.2 BQ Input Parameters

The format for including external point charges using the Bq module is shown below, supporting both explicit charge definition in the body of the block and/or loading from external files.

```
 bq [units au|nm|pm|ang...] [namespace]
   [clear]
   [force|noforce]
   [load <file> [charges <chargefile>] [format ix iy iz iq] [units au|nm|pm|ang|...] [ scale <factor> ]]
   x  y  z  q
   ...
end
```

13.5.2.1 BQ: UNITS keyword

- **units** - specify the global units for the coordinates of point charges. Allowed values are au, bohr, nanometers, nm, pm, picometers, angstrom (note that only first 3 characters are important). The default units are angstrom

13.5.2.2 BQ: NAMESPACE keyword

- **namespace** - an optional name that can be used to distinguish between potentially several point charge sets. The default value for namespace is "default", and only the set that has this namespace value will be actually used in the calculation. If for example different namespace is used, this point charge set will be processed into the run time database but not actually used unless the following set directive is encountered

```
set bq <namespace>
```

Here is an example that illustrates this

```
...
#store point charge in namespace "foo"
bq "foo"
 ...
end

#perform calculation without actually loading charges in foo
task dft energy

#activate charges in foo
set bq foo
# now DFT calculation will performed in the presence of charges in foo
task dft energy
```

13.5.2.3 BQ: CLEAR keyword

- **clear** - this directive erases all the previously specified point charges in a given namespace, prior to new setup (if any).

13.5.2.4 BQ: FORCE|NOFORCE keyword

- **force** <output_file> | **noforce** this directive triggers|disables calculation of forces on Bq charges. Default value is**noforce**, which disables force calculation. The forces will be written to <output_file> if provided or to <prefix>.bqforce.dat file. The format of the file is

```
#comment line
fx fy fz
...
```

13.5.2.5 BQ: LOAD keyword

- **load** <file> [charges <chargefile>] [format ix iy iz iq] [units au|nm|pm|**ang**|...] [scale <factor>] - this directive allows to load point charges from external file(s). You can load charges and their coordinates from a single file, or from separate files. The files do not have to follow any specific format, but blank lines and comments (starting with #) will be ignored. The actual specification of how the coordinates/charges are laid out in in the file is given by format keyword. Multiple load directives are a supported.
- <file> - the name of the file where Bq coordinates and charges are stored.
- charges <chargefile> - this optional keyword allows to load charges (NOT the coordinates) from a separate<chargefile>. In this case, only coordinates would be loaded from <file>

13.5.2.5.1 BQ: LOAD/FORMAT keyword

- format ix iy iz iq - this optional keyword allows to set the fields (separated by blanks) where x,y,z coordinates and respective charge values are to be found. If a specified field doe nor exist or contains no numerical value, the processing will skip to the next line. The default value for format is 2 3 4 5, which will work for the following example (note that the second line will not be processed here)

```
#this is a comment
coordinates are in fields 2,3,4 and charge is field 5
O  2.384   1.738   1.380  -0.9
H  2.448   1.608   0.416   0.45
H  1.560   1.268   1.608   0.45
```

13.5.2.5.2 BQ: LOAD/UNITS keyword

- units au|nm|pm|ang|... - this optional keyword sets the local coordinate units valid only for this particular load directive. Otherwise global unit definition will apply (see above)

13.5.2.5.3 BQ: LOAD/SCALE keyword

- scale <factor>

```
geometry nocenter noautosym units angstrom noautoz print
O         2.045   1.011  -1.505
H1        1.912   0.062  -1.314
H2        1.119   1.318  -1.544
end

#example of explicit Bq input
bq
2.384   1.738   1.380  -0.9
2.448   1.608   0.416   0.45
1.560   1.268   1.608   0.45
end

task dft energy

#example of implicit Bq input using load directive
bq
load bq.xyz format 1 2 3 4
end
task dft energy

#example of loading coordinates and charges separately
bq
load   bq.xyz charges bq.xyz format 1 2 3 4
end
task dft energy

#example of loading Bq's with default format (1 2 3 4) and scaling charges (to zero)
bq
load bq.xyz scale 0.0
end
task dft energy

#example of mixed Bq input
bq
load bqO.xyz format 2 3 4 6
2.448   1.608   0.416   0.45
1.560   1.268   1.608   0.45
end
task dft energy

#example of erasing Bq's
bq
clear
end
task dft energy

#example of storing Bq's in custom namespace (not activated)
bq marat
load bq.xyz format 1 2 3 4
end
task dft energy

#example of activating Bq's stored in custom namespace
set bq marat

task dft energy
```

13.5.3 How to increasing the limit on the number of bq charges

There is an internal limit of 25000 bq charges. To increase this use the following set directive in the input file

```
set bq:max_nbq 30000
```

1-D RISM

Overview

The 1D-RISM module in NWChem provides description of solvated systems following one-dimensional reference interaction site of model of Chandler and Anderson. Similar to ab-initio density-functional theory, 1D-RISM can be thought of as an approach where discrete particle representation of solvent degrees of freedom is replaced by average density field. Unlike traditional continuum solvation model, this density based representation is inherently inhomogenous and incorporates specific molecular features of the solvent. In the current implementation, 1D-RISM is not directly coupled to

QM calculations but presumed to be used as a post processing step after QM calculations which provide ESP point charges for a given solute geometry.

Then parameters for 1D-RISM calculations are defined in the rism input block

```
rism
  solute configuration <filename>
  vdw [rule <arithmetic|geometric> ] parameters <filename>
  [temp <float default 298.15>]
  [closure <hnc|kh>]
end
```

At this point energy task is supported, which is invoked using standard directive

```
task rism energy
```

- **solute configuration** - points to the file that contains information about the solute geometry. charges, and atom type mapping. The format is similar to xyz style with additional fields that specify charge and atom type. The atom type maps back to the vdw parameter file (see below). The example file is shown below

```
7

O1       -1.092111    0.733461    1.237573  -1.104415 O
O2        0.758765   -0.201687    0.473908  -1.043019 O
C1       -0.212954    1.568653   -0.833617  -0.474263 C1
C2       -0.174205    0.630432    0.357135   1.276672 C2
H1        0.360636    1.160405   -1.668859   0.102898 H
H2        0.242419    2.521128   -0.531952   0.118979 H
H3       -1.243967    1.772778   -1.139547   0.123148 H
```

- **vdw** - defines van der waals parameters
  - **rule** - optional setting that specifies combination rule, defaults to “arithmetic”
  - **parameters** - points to the file that contains vdw parameters for the system. Example file is shown below (note comments in the file)

```
#Van der Waals parameters file for RISM
# type   sigma(Angstrom) epsilon (kj/mol)
C       0.3400E+01  0.3601E+00
H       0.2600E+01  0.0628E-00
```

- **temp** - defines temperature of the system with default value of 298.15
- **closure** - specifies choice of closure

Upon completion of the run, the resulting radial distribution functions are saved into rdf_out.data file.

The computed chemical potentials in both HNC and gaussian approximations are written in the output file.

Here is the complete example input file for solvated calculation of acetic acid.

```
echo
start rism

memory global 40 mb stack 23 mb heap 5 mb

rism
  closure kh
  temp 298
  vdw rule arithmetic parameters vdw.par
  solute configuration solute2.data
end

task energy rism
```

solute2.data file

```
8

O1       0.15566663   -0.86069508    1.9256322   -0.7323104568123959 O1
O2       2.31302544   -0.61550520    1.32869265  -0.7721248369124809 O2
C1       0.69252260   -1.26942616   -0.34814880  -0.4444201659837397 C1
C2       1.15967680   -0.88531805    1.02642840   1.004561861052242  C2
H1       0.22862001   -2.26160688   -0.31011132   0.1303963270585546 H
H2      -0.08170478   -0.56654621   -0.67834692   0.1513209389506027 H
H3       1.53138139   -1.28351200   -1.04644134   0.1454517042730594 H
H4       0.48680931   -0.66362820    2.83551288   0.5171246283741536 H4
```

vdw.par file

```
O1  3.0660  0.8809
O2  2.9600  0.8792
C1  3.4000  0.4580
C2  3.4000  0.3601
H   2.1150  0.0657
H4  0.8000  0.1926
```

14 Potential Energy Surface Analysis

14.1 Geometry Optimization

14.1.1 Geometry Optimization with DRIVER

The DRIVER module is one of two drivers (also see documentation onSTEPPER) to perform a geometry optimization function on the molecule defined by input using the GEOMETRY directive. Geometry optimization is either an energy minimization or a transition state optimization. The algorithm programmed in DRIVER is a quasi-newton optimization with line searches and approximate energy Hessian updates.

DRIVER is selected by default out of the two available modules to perform geometry optimization. In order to force use of DRIVER (e.g., because a previous optimization used STEPPER) provide a DRIVER input block (below) – even an empty block will force use of DRIVER.

Optional input for this module is specified within the compound directive,

```
DRIVER
  (LOOSE || DEFAULT || TIGHT)
  GMAX <real value>
  GRMS <real value>
  XMAX <real value>
  XRMS <real value>
  EPREC <real eprec default 1e-7>
  TRUST <real trust default 0.3>
  SADSTP <real sadstp default 0.1>
  CLEAR
  REDOAUTOZ
  INHESS <integer inhess default 0>
  (MODDIR || VARDIR) <integer dir default 0>
  (FIRSTNEG || NOFIRSTNEG)
  MAXITER <integer maxiter default 20>
  BSCALE <real BSCALE default 1.0>`
  ASCALE <real ASCALE default 0.25>
  TSCALE <real TSCALE default 0.1>
  HSCALE <real HSCALE default 1.0>
    PRINT ...
  XYZ <string xyz default *file_prefix*>]
  NOXYZ
  SOCKET (UNIX || IPI_CLIENT) <string socketname default (see input description)>
END
```

On each optimization step a line search is performed. To speed up calculations (up to two times), it may be beneficial to turn off the line search using following directive:

```
set driver:linopt 0
```

14.1.1.1 Convergence criteria

```
(LOOSE || DEFAULT || TIGHT)
GMAX <real value>
GRMS <real value>
XMAX <real value>
XRMS <real value>
```

The defaults may be used, or the directives LOOSE, DEFAULT, or TIGHT specified to use standard sets of values, or the individual criteria adjusted. All criteria are in atomic units. GMAX and GRMS control the maximum and root mean square gradient in the coordinates being used (Z-matrix, redundant internals, or Cartesian). XMAX and XRMS control the maximum and root mean square of the Cartesian step.

	LOOSE	**DEFAULT**	**TIGHT**
GMAX	0.00450	0.00045	0.000015
GRMS	0.00300	0.00030	0.00001
XMAX	0.01800	0.00180	0.00006
XRMS	0.01200	0.00120	0.00004

Note that GMAX and GRMS used for convergence of geometry may significantly vary in different coordinate systems such as Z-matrix, redundant internals, or Cartesian. The coordinate system is defined in the input file (default is Z-matrix). Therefore the choice of coordinate system may slightly affect converged energy. Although in most cases XMAX and XRMS are last to converge which are always done in Cartesian coordinates, which insures convergence to the same geometry in different coordinate systems.

The old criterion may be recovered with the input

```
gmax 0.0008; grms 1; xrms 1; xmax 1
```

14.1.1.2 Available precision

```
EPREC <real eprec default 1e-7>
```

In performing a line search the optimizer must know the precision of the energy (this has nothing to do with convergence criteria). The default value of 1e-7 should be adjusted if less, or more, precision is available. Note that the default EPREC for DFT calculations is 5e-6 instead of 1e-7.

14.1.1.3 Controlling the step length

```
TRUST <real trust default 0.3>
SADSTP <real sadstp default 0.1>
```

A fixed trust radius (trust) is used to control the step during minimizations, and is also used for modes being minimized during saddle-point searches. It defaults to 0.3 for minimizations and 0.1 for saddle-point searches. The parameter sadstp is the trust radius used for the mode being maximized during a saddle-point search and defaults to 0.1.

14.1.1.4 Maximum number of steps

```
MAXITER <integer maxiter default 40>
```

By default at most 40 geometry optimization steps will be taken, but this may be modified with this directive.

14.1.1.5 Discard restart information

```
CLEAR
```

By default Driver reuses Hessian information from a previous optimization, and, to facilitate a restart also stores which mode is being followed for a saddle-point search. This option deletes all restart data.

14.1.1.6 Regenerate internal coordinates

```
REDOAUTOZ
```

Deletes Hessian data and regenerates internal coordinates at the current geometry. Useful if there has been a large change in the geometry that has rendered the current set of coordinates invalid or non-optimal.

14.1.1.7 Initial Hessian

```
INHESS <integer inhess default 0>
```

- 0 = Default … use restart data if available, otherwise use diagonal guess.
- 1 = Use diagonal initial guess.
- 2 = Use restart data if available, otherwise transform Cartesian Hessian from previous frequency calculation.

In addition, the diagonal elements of the initial Hessian for internal coordinates may be scaled using separate factors for bonds, angles and torsions with the following

```
BSCALE <real bscale default 1.0>
ASCALE <real ascale default 0.25>
TSCALE <real tscale default 0.1>
```

These values typically give a two-fold speedup over unit values, based on about 100 test cases up to 15 atoms using 3-21g and 6-31g* SCF. However, if doing many optimizations on physically similar systems it may be worth fine tuning these parameters.

Finally, the entire Hessian from any source may be scaled by a factor using the directive

```
HSCALE <real hscale default 1.0>
```

It might be of utility, for instance, when computing an initial Hessian using SCF to start a large MP2 optimization. The SCF vibrational modes are expected to be stiffer than the MP2, so scaling the initial Hessian by a number less than one might be beneficial.

14.1.1.8 Mode or variable to follow to saddle point

```
(MODDIR || VARDIR) <integer dir default 0>
(FIRSTNEG || NOFIRSTNEG)
```

When searching for a transition state the program, by default, will take an initial step uphill and then do mode following using a fuzzy maximum overlap (the lowest eigen-mode with an overlap with the previous search direction of 0.7 times the maximum overlap is selected). Once a negative eigen-value is found, that mode is followed regardless of overlap.

The initial uphill step is appropriate if the gradient points roughly in the direction of the saddle point, such as might be the case if a constrained optimization was performed at the starting geometry. Alternatively, the initial search direction may be chosen to be along a specific internal variable (using the directive VARDIR) or along a specific eigen-mode (using MODDIR). Following a variable might be valuable if the initial gradient is either very small or very large. Note that the eigen-modes in the optimizer have next-to-nothing to do with the output from a frequency calculation. You can examine the eigen-modes

used by the optimizer with

```
driver; print hvecs; end
```

The selection of the first negative mode is usually a good choice if the search is started in the vicinity of the transition state and the initial search direction is satisfactory. However, sometimes the first negative mode might not be the one of interest (e.g., transverse to the reaction direction). If NOFIRSTNEG is specified, the code will not take the first negative direction and will continue doing mode-following until that mode goes negative.

14.1.1.9 Optimization history as XYZ files

```
XYZ [<string xyz default $fileprefix>]
NOXYZ
```

The XYZ directive causes the geometry at each step (but not intermediate points of a line search) to be output into separate files in the permanent directory in XYZ format. The optional string will prefix the filename. The NOXYZ directive turns this off.

For example, the input

```
driver; xyz test; end
```

will cause files test-000.xyz, test-001.xyz, … to be created in the permanent directory.

The script rasmolmovie in the NWChem contrib directory can be used to turn these into an animated GIF movie.

14.1.1.10 i-PI Socket communication

```
SOCKET (UNIX | IPI_CLIENT) <socketname> [RETRIES <integer>] [DELAY <float>]
```

The SOCKET directive enables NWChem to communicate with other software packages – such as i-PI or ASE or EON – via the i-PI socket protocol.

Communication is done either over Unix sockets (SOCKET UNIX) or IP sockets (SOCKET IPI_CLIENT):

- Unix sockets - NWChem will create and bind to a UNIX socket file located at /tmp/ipi_<socketname>. If not specified, <socketname> will default to nwchem.
- IP sockets - NWChem will bind to the IP address and port specified by <socketname>. If not specified, <socketname> will default to 127.0.0.1:31415.

The behavior of the client when first attempting to connect to the server can be controlled with two optional keywords:

- RETRIES <integer>: Specifies the maximum number of times the client will try to connect to the server socket before exiting with an error. If not provided, this defaults to 30 tries.
- DELAY <integer>: Specifies the time in seconds to wait between failed connection attempts. If not provided, this defaults to 2 seconds.

The SOCKET directive is only useful when used in conjunction with other software packages that support communication via the i-PI socket protocol. For more information, see the i-PI documentation.

14.1.1.11 Print options

The UNIX command

```
grep '^@' \< output
```

will extract a pretty table summarizing the optimization.

If you specify the NWChem input

```
scf; print none; end
driver; print low; end
task scf optimize
```

you'll obtain a pleasantly terse output.

For more control, these options for the standard print directive are recognized

- debug - prints a large amount of data. Don't use in parallel.
- high - print the search direction in internals
- default - prints geometry for each major step (not during the line search), gradient in internals (before and after application of constraints)
- low - prints convergence and energy information. At convergence prints final geometry, change in internals from initial geometry

and these specific print options

- finish (low) - print geometry data at end of calculation
- bonds (default) - print bonds at end of calculation
- angles (default) - print angles at end of calculation
- hvecs (never) - print eigen-values/vectors of the Hessian
- searchdir (high) - print the search direction in internals
- "internal gradient" (default) - print the gradient in internals
- sadmode (default) - print the mode being followed to the saddle point

14.1.2 Geometry Optimization with STEPPER

The STEPPER module performs a search for critical points on the potential energy surface of the molecule defined by input using the GEOMETRY directive. Since STEPPER is not the primary geometry optimization module in NWChem the compound directive is required; the DRIVER module is the default. Input for this module is specified within the compound directive,

```
STEPPER
  ...
END
```

The presence of the STEPPER compound directive automatically turns off the default geometry optimization tool DRIVER . Input specified for the STEPPER module must appear in the input file after the GEOMETRY directive, since it must know the number of atoms that are to be used in the geometry optimization. In the current version of NWChem, STEPPER can be used only with geometries that are defined in Cartesian coordinates. STEPPER removes translational and rotational components before determining the step direction (5 components for linear systems and 6 for others) using a standard Eckart algorithm. The default initial guess nuclear Hessian is the identity matrix.

The default in STEPPER is to minimize the energy as a function of the geometry with a maximum of 20 geometry optimization iterations. When this is the desired calculation, no input is required other than the STEPPER compound directive. However, the user also has the option of defining different tasks for the STEPPER module, and can vary the number of iterations and the convergence criteria from the default values. The input for these options is described in the following sections.

14.1.2.1 MIN and TS: Minimum or transition state search

The default is for STEPPER to minimize the energy with respect to the geometry of the system. This default behavior may be forced with the directive

```
MIN
```

STEPPER can also be used to find the transition state by following the lowest eigenvector of the nuclear Hessian. This is usually invoked by using the saddle keyword on the TASK directive, but it may also be selected by specifying the directive

```
TS
```

in the STEPPER input.

14.1.2.2 TRACK: Mode selection

STEPPER has the ability to *track* a specific mode during an optimization for a transition state search, the user can also have the module track the eigenvector corresponding to a specific mode. This is done by specifying the directive

```
TRACK [nmode <integer nmode default 1>]
```

The keyword TRACK tells STEPPER to track the eigenvector corresponding to the integer value of during a transition state walk. (Note: this input is invalid for a minimization walk since following a specific eigenvector will not necessarily give the desired local minimum.) The step is constructed to go up in energy along the nmode eigenvector and down in all other degrees of freedom.

14.1.2.3 MAXITER: Maximum number of steps

In most applications, 20 stepper iterations will be sufficient to obtain the energy minimization. However, the user has the option of specifying the maximum number of iterations allowed, using the input line,

```
MAXITER <integer maxiter default 20>
```

The value specified for the integer defines the maximum number of geometry optimization steps. The geometry optimization will restart automatically.

14.1.2.4 TRUST: Trust radius

The size of steps that can be taken in STEPPER is controlled by the trust radius which has a default value of 0.1. Steps are constrained to be no larger than the trust radius. The user has the option of overriding this default using the keyword TRUST, with the following input line,

```
TRUST <real radius default 0.1>
```

The larger the value specified for the variable radius, the larger the steps that can be taken by STEPPER . Experience has shown that for larger systems (i.e., those with 20 or more atoms), a value of 0.5, or greater, usually should be entered for .

14.1.2.5 CONVGGM, CONVGG and CONVGE: Convergence criteria

Three convergence criteria can be specified explicitly for the STEPPER calculations. The keyword CONVGGM allows the user to specify the convergence tolerance for the largest component of the gradient. This is the primary convergence criterion, as per the default settings, although all three criteria are in effect. this default setting is consistent with the other optimizer module DRIVER . The input line for CONVGGM has the following form,

```
CONVGGM <real convggm default 8.0d-04>
```

The keyword CONVGG allows the user to specify the convergence tolerance for the gradient norm for all degrees of freedom.

The input line is of the following form,

```
CONVGG <real convgg default 1.0d-02>
```

The entry for the real variable <convgg> should be approximately equal to the square root of the energy convergence tolerance.

The energy convergence tolerance is the convergence criterion for the energy difference in the geometry optimization in STEPPER . It can be specified by input using a line of the following form,

```
CONVGE <real convge default 1.0d-04>
```

14.1.2.6 Backstepping in STEPPER

If a step taken during the optimization is too large (e.g., the step causes the energy to go up for a minimization or down for a transition state search), the STEPPER optimizer will automatically “backstep” and correct the step based on information prior to the faulty step. If you have an optimization that “backsteps” frequently then the initial trust radius should most likely be decreased.

14.1.2.7 Initial Nuclear Hessian Options

Stepper uses a modified Fletcher-Powell algorithm to find the transition state or energy minimum on the potential energy hypersurface.
There are two files left in the user’s permanent directory that are used to provide an initial hessian to the critical point search algorithm. If these files do not exist then the default is to use a unit matrix as the initial hessian.
Once Stepper executes it generates a binary dump file by the name of name.stpr41 which will be used on all subsequent stepper runs and modified with the current updated hessian. The default file prefix is the “name” that is used (see START). It also stores the information for the last valid step in case the algorithm must take a “backstep”. This file is the working data store for all stepper-based optimizations. This file is never deleted by default and is the first source of an initial hessian.
The second source of an initial hessian is an ASCII file that contains the lower triangular values of the initial hessian. This is stored in file name.hess , where “name” is again the default file prefix. This is the second source of an initial hessian and is the method used to incorporate an initial hessian from any other source (e.g., another ab initio code, a molecular mechanics code, etc.,). To get a decent starting hessian at a given point you can use the task specification task scf hessian, with a smaller basis set, which will by default generate the name.hess file. Then you may define your basis set of choice and proceed with the optimization you desire.

14.2 Hessians

14.2.1 Overview

This section relates to the computation of analytic hessians which are available for open and closed shell SCF, except ROHF and for closed shell and unrestricted open shell DFT [1]. Analytic hessians are not currently available for SCF or DFT calculations relativistic all-electron methodologies or for charge fitting with DFT. The current algorithm is fully in-core and does not use symmetry.

There is no required input for the Hessian module. This module only impacts the hessian calculation. For options for calculating the frequencies, please see the Vibrational module.

14.2.2 Hessian Module Input

All input for the Hessian Module is optional since the default definitions are usually correct for most purposes. The generic

module input begins with hessian and has the form:

```
hessian
  thresh <real tol default 1d-6>
  print ...
  profile
end
```

14.2.2.1 Defining the wavefunction threshold

You may modify the default threshold for the wavefunction. This keyword is identical toTHRESH in the SCF, and the CONVERGENCE gradient in the DFT. The usual defaults for the convergence of the wavefunction for single point and gradient calculations is generally not tight enough for analytic hessians. Therefore, the hessian, by default, tightens these up to 1d-6 and runs an additional energy point if needed. If, during an analytic hessian calculation, you encounter an error:

```
cphf_solve:the available MOs do not satisfy the SCF equations
```

the convergence criteria of the wavefunction generally needs to be tightened.

14.2.2.2 Profile

The PROFILE keyword provides additional information concerning the computation times of different sections of the hessian code. Summary information is given about the maximum, minimum and average times that a particular section of the code took to complete. This is normally only useful for developers.

14.2.2.3 Print Control

Known controllable print options are shown in the table below:

Name	Print Level	Description
hess_follow	high	more information about where the calculation is
cphf_cont	debug	detailed CPHF information
nucdd_cont	debug	detailed nuclear contribution information
onedd_cont	debug	detailed one electron contribution information
twodd_cont	debug	detailed two electron contribution information
fock_xc	debug	detailed XC information during the fock builds

14.2 Vibrational frequencies

The nuclear hessian which is used to compute the vibrational frequencies can be computed by finite difference for any ab initio wave-function that has analytic gradients or by analytic methods for SCF and DFT (see Hessians for details). The appropriate nuclear hessian generation algorithm is chosen based on the user input when TASK frequencies is the task directive.

The vibrational package was integrated from the Utah Messkit and can use any nuclear hessian generated from the driver routines, finite difference routines or any analytic hessian modules. There is no required input for the “VIB” package. VIB computes the Infra Red frequencies and intensities for the computed nuclear hessian and the “projected” nuclear hessian.

The VIB module projects out the translations and rotations of the nuclear hessian using the standard Eckart projection algorithm. It also computes the zero point energy for the molecular system based on the frequencies obtained from the projected hessian.

The default mass of each atom is used unless an alternative mass is provided via thegeometry input or redefined using the vibrational module input. The default mass is the mass of the most abundant isotope of each element. If the abundance was roughly equal, the mass of the isotope with the longest half life was used.

In addition, the vibrational analysis is given at the default standard temperature of 298.15 degrees.

14.2.3 Vibrational Module Input

All input for the Vibrational Module is optional since the default definitions will compute the frequencies and IR intensities. The generic module input can begin with vib, freq, frequency and has the form:

```
{freq || vib || frequency}`
  [reuse [<string hessian_filename>]]
  [mass <integer lexical_index> <real new_mass>]
  [mass <string tag_identifier> <real new_mass>]
  [{temp || temperature} <integer number_of_temperatures>\
      <real temperature1 temperature2 ...>]
  [animate [<real step_size_for_animation>]]
  [fd_delta [<real step_size_for_fd_hessian>]]
  [filename <string file_set_name> [overwrite]]
end
```

14.2.3.1 Hessian File Reuse

By default the task frequencies directive will recompute the hessian. To reuse the previously computed hessian you need only specify reuse in the module input block. If you have stored the hessian in an alternate place you may redirect the reuse directive to that file by specifying the path to that file.

```
reuse /path_to_hessian_file
```

This will reuse your saved Hessian data but one caveat is that the geometry specification at the point where the hessian is computed must be the default “geometry” on the current run-time-data-base for the projection to work properly.

14.2.3.2 Redefining Masses of Elements

You may also modify the mass of a specific center or a group of centers via the input.

To modify the mass of a specific center you can simply use:

```
mass 3 4.00260324
```

which will set the mass of center 3 to 4.00260324 AMUs. The lexical index of centers is determined by the geometry object.

To modify all Hydrogen atoms in a molecule you may use the tag based mechanism:

```
mass hydrogen 2.014101779
```

The mass redefinitions always start with the default masses and change the masses in the order given in the input. Care must be taken to change the masses properly. For example, if you want all hydrogens to have the mass of Deuterium and the third hydrogen (which is the 6th atomic center) to have the mass of Tritium you must set the Deuterium masses first with the tag based mechanism and then set the 6th center’s mass to that of Tritium using the lexical center index mechanism.

The mass redefinitions are not fully persistent on the run-time-data-base. Each input block that redefines masses will invalidate the mass definitions of the previous input block. For example,

```
freq
  reuse
  mass hydrogen 2.014101779
end
task scf frequencies
freq
  reuse
  mass oxygen 17.9991603
end
task scf frequencies
```

will use the new mass for all hydrogens in the first frequency analysis. The mass of the oxygen atoms will be redefined in the second frequency analysis but the hydrogen atoms will use the default mass. To get a modified oxygen and hydrogen analysis you would have to use:

```
freq
  reuse
  mass hydrogen 2.014101779
end
task scf frequencies
freq
  reuse
  mass hydrogen 2.014101779
  mass oxygen 17.9991603
end
task scf frequencies
```

14.2.3.3 Temp or Temperature

The “VIB” module can generate the vibrational analysis at various temperatures other than at standard room temperature. Either temp or temperature can be used to initiate this command.

To modify the temperature of the computation you can simply use:

```
temp 4 298.15 300.0 350.0 400.0
```

At this point, the temperatures are persistant and so the user must “reset” the temperature if the standard behavior is required after setting the temperatures in a previous “VIB” command, i.e.

```
temp 1 298.15
```

14.2.3.4 Animation

The “VIB” module also can generate mode animation input files in the standard xyz file format for graphics packages like RasMol or XMol There are scripts to automate this for RasMol in `$NWCHEM_TOP/contrib/rasmolmovie` . Each mode will have 20 xyz files generated that cycle from the equilibrium geometry to 5 steps in the positive direction of the mode vector, back to 5 steps in the negative direction of the mode vector, and finally back to the equilibrium geometry. By default these files are not generated. To activate this mechanism simply use the following input directive

```
animate
```

anywhere in the frequency/vib input block.

Given an ordered list of files containing molecular coordinates in XYZ format, the rasmolmovie shell script generates an animated gif for each of the six possible views down a Cartesian axis.

It uses the free utilities

- rasmol https://www.umass.edu/microbio/rasmol/ to manipulate the molecule and generate the individual frames
- convert from ImageMagick https://www.imagemagick.org to combine the frames into an animated gif

It should be easy to modify the script to other file formats or animation tools.

14.2.3.4.1 Controlling the Step Size Along the Mode Vector

By default, the step size used is 0.15 a.u. which will give reliable animations for most systems. This can be changed via the animate input directive, e.g.

```
vib
  animate 0.20
end
```

where is the real number that is the magnitude of each step along the eigenvector of each nuclear hessian mode in atomic units.

14.2.3.5 Specifying filenames for animated normal modes

By default, normal modes will be stored in files that start with “freq.m“. This is inconvenient if more than vibrational analysis is run in a single input file. To specify different filename for a particular vibrational analysis use the directive

```
filename <file_set_name> [overwrite]
```

where is the name that will be prepended to the usual filenames. In addition the code by default requires all files to be new files. When the option “overwrite” is provided any pre-existing files will simply be overwritten.

14.2.3.6 Controlling the Step Size of the Finite difference Hessian

By default, the step size used for calculating the finite difference Hessian is 0.010 a.u. for DFT and NWPW modules, and 0.001 a.u. otherwise This can be changed via the fd_delta input directive, e.g.

```
vib
  fd_delta 0.005
end
```

where is the real number that is the magnitude of each displacement in atomic units for the calculation of the finite difference Hessian. For older versions of NWChem without the fd_delta option just set the “stpr_gen:delta” value on the runtime database, e.g.

```
set stpr_gen:delta 0.005
```

14.2.3.7 An Example Input Deck

This example input deck will optimize the geometry for the given basis set, compute the frequencies for H_2O, H_2O at different temperatures, D_2O, HDO, and TDO.

```
start  h2o
title Water
geometry units au autosym
  O     0.00000000    0.00000000    0.00000000
  H     0.00000000    1.93042809   -1.10715266
  H     0.00000000   -1.93042809   -1.10715266
end
basis noprint
  H library sto-3g
  O library sto-3g
end
scf; thresh 1e-6; end
driver; tight; end
task scf optimize

scf; thresh 1e-8; print none; end
task scf freq

freq
 reuse; temp 4 298.15 300.0 350.0 400.0
end
task scf freq

freq
 reuse; mass H 2.014101779
 temp 1 298.15
end
task scf freq

freq
 reuse; mass 2 2.014101779
end
task scf freq

freq
 reuse; mass 2 2.014101779 ; mass 3 3.01604927
end
task scf freq
```

14.2.4 References

1. Johnson, B. G.; Frisch, M. J. An Implementation of Analytic Second Derivatives of the Gradient-Corrected Density Functional Energy. *The Journal of Chemical Physics* **1994**, *100* (10), 7429–7442. https://doi.org/10.1063/1.466887

14.3 Constraints

The constraints directive allows the user to specify which constraints should be imposed on the system during the analysis of potential energy surface. Currently such constraints are limited to fixed atom positions and harmonic restraints (springs) on the distance between the two atoms. The general form of constraints block is presented below:

```
CONSTRAINTS [string name ] \
        [clear] \
        [enable||disable] \
        [fix atom <integer list>] \
        [spring bond <integer atom1> <integer atom2> <real k> <real r0> ]
        [spring dihedral <integer atom1> <integer atom2> <integer atom3> <integer atom4> <real k> <real phi0 in degrees> ]
        [spring bondings <real K0> <real gamma0> [<real ca> <integer atom1a> <integer atom2a>
                                <real cb> <integer atom1b> <integer atom2b>
                                ...]]
        [penalty pbondings <real K0> <real gcut0> <real gamma0> [<real ca> <integer atom1a> <integer atom2a>
                                        <real cb> <integer atom1b> <integer atom2b>
                                        ...]]
END
```

The keywords are described below

- name - optional keyword that associates a name with a given set of constraints. Any unnamed set of constraints will be given a name *default* and will be automatically loaded prior to a calculation. Any constraints with the name other than *default* will have to be loaded manually using the SET directive. For example,

```
CONSTRAINTS one
  spring bond 1 3 5.0 1.3
  fix atom 1
END
```

the above constraints can be loaded using the following set directive that assigns the name `one` as the current constraint

```
set constraints one
....
task ....
```

- `clear` - destroys any prior constraint information. This may be useful when the same constraints have to be redefined or completely removed from the runtime database.
- `enable||disable` - enables or disables without actually removing the information from the runtime database.
- `fix atom` - fixes atom positions during geometry optimization. This directive requires an integer list that specifies which atoms are to be fixed. This directive can be repeated within a given constraints block. To illustrate the use `fix atom` directive let us consider a situation where we would like to fix atoms 1, 3, 4, 5, 6 while performing an optimization on some hypothetical system. There are actually several ways to enter this particular constraint.
- There is a straightforward option which requires the most typing:

```
constraints
  fix atom 1 3 4 5 6
end
```

- Second method uses list input:

```
constraints
  fix atom 1 3:6
end
```

- Third approach illustrates the use of multiple `fix atom` directives:

```
constraints
  fix atom 1
  fix atom 3:6
end
```

- the `spring bond` < *i j k* r_0 > - places a spring with a spring constant *k* and equilibrium length r_0 between atoms *i* and *j* (all in atomic units). Please note that this type of constraint adds an additional term to the total energy expression

$$E = E_{total} + \frac{1}{2} k (r_{ij} - r_0)^2$$

This additional term forces the distance between atoms *i* and *j* to be in the vicinity of r_0 but never exactly that. In general the spring energy term will always have some nonzero residual value, and this has to be accounted for when comparing total energies. The `spring bond` directive can be repeated within a given constraints block. If the spring between the same pair of atoms is defined more than once, it will be replaced by the latest specification in the order it appears in the input block.

14.4 Nudged Elastic Band (NEB) method

14.4.1 Overview

The NEB module is an implementation of the nudged elastic band (NEB) method of Jonsson et al., and it is one of two drivers in NWChem that can be used to perform minimum energy path optimizations. NEB can be used at all levels of theory, including SCF, HF, DFT, PSPW, BAND, MP2, RIMP2, CCSD, TCE.

Input to the NEB modules is contained with the NEB block

```
NEB
 ...
END
```

To run a NEB calculation the following the following task directives is used

```
TASK <theory> NEB
TASK <theory> NEB ignore
```

where <theory> is SCF, HF, DFT, PSPW, BAND, MP2, CCSD, TCE, etc.. The Task directive with the ignore option is recommended, otherwise NWChem will crash if the path is not optimized in the allowed maximum number of iterations.

Optional input for this module is specified within the compound directive,

```
NEB
  NBEADS <integer nbeads default 5>
  KBEADS <float kbeads default 0.1>
  MAXITER <integer maxiter default 5>

  STEPSIZE <integer stepsize default 1.0>
  NHIST <integer nhist default 5>
  ALGORITHM <integer algorithm default 0>

  [loose | default | tight]
  GMAX <float gmax default 0.00045>
  GRMS <float grms default 0.00030>
  XMAX <float xmax default 0.00018>
  XMRS <float xmrs default 0.00012>

  [IMPOSE]
  [HASMIDDLE]
  [XYZ_PATH <string xyzfilename>]
  [RESET]
  [PRINT_SHIFT <integer print_shift default 0>]
END
```

The following list describes the input for the NEB block

- **nbeads** - number of beads (or images) used to represent the path
- **kbeads** - value for the NEB spring constant
- **maxiter** - maximum number of NEB path optimizations to be performed
- **stepsize** - value for the stepsize used in the optimization. Typically less than 1.
- **nhist** - number of histories to use for quasi-Newton optimization (algorithm =0)
- **LOOSE|DEFAULT|TIGHT** - options specifying thresholds for convergence
- **gmax** - value for the maximum gradient used to determine convergence
- **grms** - value for the root mean square gradient used to determine convergence
- **xmax** - value for the maximum cartesian step used to determine convergence
- **xrmx** - value for the root mean square cartesian step used to determine convergence
- **algorithm** - 0: quasi-Newton Fixed Point optimization, 1: dampled Verlet optimization, 2: refining conjugate gradient optimization
- **IMPOSE** - if specified causes the initial geometries used to specify the path to be aligned with one another
- **HASMIDDLE** - if specified causes the initial path to use the the “midgeom” geometry to be used as the midpoint, i.e. the initial path is defined as a linear morphing from “geometry” –> “midgeom” –> “endgeom”
- **XYZ_PATH** - if specified the initial path is defined from the sequence of geometries contained in **xyzfilename**
- **RESET** - if specified causes the NEB optimization and path to be started from scratch
- **print_shift** - setting the PRINT_SHIFT directive causes the path energies and geometries to be outputed every <print_shift> steps. The current path energies are appended to the file jobname.neb_epath and the current geometries are appended to the file jobname.nebpath_"current iteration".xyz .

14.4.2 Setting up initial path

There are three different ways to define the initial path for NEB optimization.

- Linear interpolation between two geometries

The geometries in the path are defined by

$$\vec{R}^i_{xyz} = \vec{R}^1_{xyz} + \frac{i-1}{nbeads-1} (\vec{R}^{nbeads}_{xyz}-\vec{R}^{1}_{xyz}), \ for \ i=1,nbeads$$

where the starting geometry $\vec{R}^1_{xyz}$ is entered in the geometry block labeled `geometry`, e.g.

```
geometry nocenter noautosym noautoz
O  0.00000000    -0.02293938     0.00000000
H  0.00000000     0.55046969     0.75406534
H  0.00000000     0.55046969    -0.75406534
end
```

and the last geometry in the path $\vec{R}^{nbeads}_{xyz}$ is entered in the geometry block label `endgeom`, e.g.

```
geometry endgeom nocenter noautosym noautoz
O  0.00000000     0.02293938     0.00000000
H  0.00000000    -0.55046969     0.75406534
H  0.00000000    -0.55046969    -0.75406534
end
```

- Linear interpolation between three geometries

The geometries for this path are defined by

$$\vec{R}^i_{xyz} = \vec{R}^1_{xyz} + \frac{i-1}{nbeads/2-1} (\vec{R}^{nbeads/2}_{xyz}-\vec{R}^{1}_{xyz}), \ for \ i=1,nbeads/2$$

and

$$\vec{R}^i_{xyz} = \vec{R}^{nbeads/2}_{xyz} + \frac{i-nbeads/2}{nbeads/2-1} (\vec{R}^{nbeads}_{xyz}-\vec{R}^{nbeads/2}_{xyz}), \ for \ i=nbeads/2+1,nbeads$$

where the starting $\vec{R}^1_{xyz}$, middle $\vec{R}_{xyz}^{nbeads/2}$ and last $\vec{R}^{nbeads}_{xyz}$ geometries are entered in the geometry blocks `geometry`, `midgeom` and `endgeom` respectively, e.g.

```
geometry nocenter noautosym noautoz
O  0.00000000    -0.02293938     0.00000000
H  0.00000000     0.55046969     0.75406534
H  0.00000000     0.55046969    -0.75406534
end

geometry midgeom nocenter noautosym noautoz
O  0.00000000     0.00000000     0.00000000
H  0.00000000     0.00000000     1.00000000
H  0.00000000     0.00000000    -1.00000000
end

geometry endgeom nocenter noautosym noautoz
O  0.00000000     0.02293938     0.00000000
H  0.00000000    -0.55046969     0.75406534
H  0.00000000    -0.55046969    -0.75406534
end
```

- Using `xyz_path` to explicitly input a path of geometries

The `xyz_path` option can also be used to define the initial path.

```
...
NEB
  ...
  XYZ_PATH path.xyz
END
...
```

where path.xyz contains a list of geometries in xyz format, e.g.

```
---------- path.xyz ----------
         3
energy=  -17.107207699285738
O          0.000000   -0.022939    0.000000
H          0.000000    0.550469    0.754065
H          0.000000    0.550469   -0.754065
         3
energy=  -17.094903833074170
O         -0.000003   -0.110080   -0.000000
H         -0.000000    0.273180    0.847029
H         -0.000000    0.273180   -0.847029
         3
energy=  -17.063823686395292
O         -0.000000   -0.000080   -0.000000
H          0.000000   -0.000002    0.941236
H          0.000000   -0.000002   -0.941236
         3
energy=  -17.094944036147005
O         -0.000000    0.110472   -0.000000
H         -0.000000   -0.273172    0.846957
H         -0.000000   -0.273172   -0.846957
         3
energy=  -17.107208157343706
O          0.000000    0.022939    0.000000
H          0.000000   -0.550469    0.754065
H          0.000000   -0.550469   -0.754065
---------- path.xyz ----------
```

14.4.3 Convergence criteria

The defaults may be used, or the directives LOOSE, DEFAULT, or TIGHT specified to use standard sets of values, or the individual criteria adjusted. All criteria are in atomic units. GMAX and GRMS control the maximum and root mean square gradient in the coordinates. XMAX and XRMS control the maximum and root mean square of the Cartesian step.

	LOOSE	**DEFAULT**	**TIGHT**
GMAX	0.0045d0	0.00045	0.000015
GRMS	0.0030d0	0.00030	0.00001
XMAX	0.0054d0	0.00180	0.00006
XRMS	0.0036d0	0.00120	0.00004

14.4.4 NEB Tutorial 1: H2O Inversion

(input:h2o-neb.nw, output:h2o-neb.nwout, datafiles: h2o-neb.neb_epath.dat h2o-neb.neb_final_epath.dat)

(xyzfiles: h2o-neb.nebpath_000001.xyz h2o-neb.nebpath_000005.xyz h2o-neb.nebpath_000010.xyz h2o-neb.nebpath_000020.xyz h2o-neb.nebpath_final.xyz)



```
Title "H2O inversion calculation"
echo
start h2o-neb

geometry nocenter noautosym noautoz
O  0.00000000    -0.02293938     0.00000000
H  0.00000000     0.55046969     0.75406534
H  0.00000000     0.55046969    -0.75406534
end

geometry endgeom nocenter noautosym noautoz
O  0.00000000     0.02293938     0.00000000
H  0.00000000    -0.55046969     0.75406534
H  0.00000000    -0.55046969    -0.75406534
end
### Gaussian DFT ###
basis
* library 3-21G
end

dft
  xc b3lyp
  maxiter 5001
  cgmin
end

neb
  nbeads 10
  kbeads 1.0
  maxiter 10
  stepsize 0.10
  print_shift 1
end
task dft neb ignore
neb
  # increase the number of images
  nbeads 20
  kbeads 1.0
  stepsize 1.0
  maxiter 30
  loose
end
task dft neb ignore
```

After each optimization step the path energies are outputed as follows

```
neb: Path Energy #                  9
neb:                    1  -75.970000166349976
neb:                    2  -75.973958450556779
neb:                    3  -75.973964391052448
neb:                    4  -75.973965560274110
neb:                    5  -75.973961077512683
neb:                    6  -75.973087554095144
neb:                    7  -75.965847261117744
neb:                    8  -75.950292780255126
neb:                    9  -75.932932759963109
neb:                   10  -75.921912278179292
neb:                   11  -75.921834552460439
neb:                   12  -75.932680002200939
neb:                   13  -75.949868818688529
neb:                   14  -75.965372754426866
neb:                   15  -75.972788885848303
neb:                   16  -75.973958649400714
neb:                   17  -75.973965255113598
neb:                   18  -75.973964962774133
neb:                   19  -75.973959526041568
neb:                   20  -75.970000163960066
```

Another way to keep track of the optimization process is to run the following grep command on the output file.

```
[WE24397:NWChem/NEB/Example2] bylaska% grep @ h2o-neb.nwout
@neb
@neb NEB Method
@neb algorithm      =         0
@neb maxiter        =        10
@neb nbeads         =        10
@neb nhist          =         5
@neb natoms         =         3
@neb stepsize       = 0.100E+01
@neb trust          = 0.100E+00
@neb kbeads         = 0.100E+00
@neb Gmax tolerance = 0.450E-03
@neb Grms tolerance = 0.300E-03
@neb Xmax tolerance = 0.180E-03
@neb Xrms tolerance = 0.120E-03
@neb
@neb Step    Intrinsic E    Mid-Point E     Minimum E     Maximum E   Gmax     Grms     Xrms     Xmax   Walltime
@neb ---- --------------- --------------- --------------- --------------- -------- -------- -------- -------- --------
@neb    1    -75.951572    -75.921109    -75.970632    -75.921109  0.55875  0.01606  0.14221  1.54029    454.9
@neb    2    -75.953755    -75.923180    -75.972590    -75.923177  0.38930  0.01116  0.01588  0.45644    624.4
@neb    3    -75.956726    -75.924391    -75.972861    -75.924387  0.25587  0.00961  0.03673  0.83118    805.2
@neb    4    -75.957861    -75.924279    -75.973059    -75.924275  0.23572  0.00894  0.01793  0.24399    971.8
@neb    5    -75.959613    -75.925045    -75.973869    -75.925036  0.10257  0.00464  0.03197  0.20350   1152.8
@neb    6    -75.959964    -75.925503    -75.973957    -75.925486  0.04762  0.00196  0.00905  0.10433   1316.4
@neb    7    -75.960068    -75.925822    -75.973956    -75.925791  0.03897  0.00141  0.00308  0.04432   1519.9
@neb    8    -75.960091    -75.925914    -75.973959    -75.925877  0.03707  0.00127  0.00070  0.01691   2055.8
@neb    9    -75.960129    -75.926078    -75.973962    -75.926028  0.03353  0.00108  0.00127  0.03707   2297.2
@neb   10    -75.960142    -75.926142    -75.973963    -75.926085  0.03199  0.00101  0.00054  0.00420   2756.6
@neb   NEB calculation not converged
@neb
@neb NEB Method
@neb algorithm      =         0
@neb maxiter        =        30
@neb nbeads         =        20
@neb nhist          =         5
@neb natoms         =         3
@neb stepsize       = 0.100E+01
@neb trust          = 0.100E+00
@neb kbeads         = 0.100E+01
@neb Gmax tolerance = 0.450E-02
@neb Grms tolerance = 0.300E-02
@neb Xmax tolerance = 0.540E-02
@neb Xrms tolerance = 0.360E-02
@neb
@neb Step    Intrinsic E    Mid-Point E     Minimum E     Maximum E   Gmax     Grms     Xrms     Xmax   Walltime
@neb ---- --------------- --------------- --------------- --------------- -------- -------- -------- -------- --------
@neb    1    -75.960225    -75.921704    -75.973965    -75.921669  0.24799  0.00398  0.00272  0.08741   3966.5
@neb    2    -75.960339    -75.921782    -75.973965    -75.921745  0.24794  0.00328  0.00199  0.12148   5023.2
@neb    3    -75.960424    -75.921742    -75.973965    -75.921701  0.19390  0.00286  0.00164  0.08342   5741.4
@neb    4    -75.960494    -75.921849    -75.973965    -75.921804  0.19681  0.00266  0.00143  0.09030   6079.7
@neb    5    -75.960646    -75.921874    -75.973965    -75.921820  0.17459  0.00240  0.00241  0.22047   6751.5
@neb    6    -75.960674    -75.921856    -75.973965    -75.921797  0.14246  0.00165  0.00060  0.00256   7572.3
@neb    7    -75.960724    -75.921884    -75.973966    -75.921817  0.13004  0.00153  0.00082  0.05401   7893.3
@neb    8    -75.960747    -75.921892    -75.973966    -75.921822  0.12809  0.00149  0.00038  0.00237   8631.2
@neb    9    -75.960792    -75.921912    -75.973966    -75.921835  0.12267  0.00142  0.00075  0.05081   9222.0
@neb   10    -75.960813    -75.921923    -75.973966    -75.921841  0.11902  0.00138  0.00035  0.00212  10163.2
@neb   11    -75.960834    -75.921934    -75.973966    -75.921846  0.11569  0.00135  0.00035  0.00203  10478.3
@neb   12    -75.961060    -75.922060    -75.973966    -75.921889  0.07709  0.00104  0.00365  0.30944  10863.8
@neb   13    -75.961255    -75.922186    -75.973966    -75.921919  0.04600  0.00087  0.00309  0.19999  11357.0
@neb   14    -75.961405    -75.922286    -75.973966    -75.921927  0.03549  0.00079  0.00244  0.03857  11860.0
@neb  NEB calculation converged
```

14.4 Zero Temperature String Method

14.4.5 Overview

The STRING module is an implementation of the zero temperature string method of vanden Eijden et al., and it is one of two drivers in NWChem that can be used to perform minimum energy path optimizations. STRING can be used at all levels of theory, including SCF, HF, DFT, PSPW, BAND, MP2, RIMP2, CCSD, TCE.

Input to the STRING module is contained with the STRING block

```
STRING
 ...
END
```

To run a STRING calculation the following the following task directives is used

```
TASK <theory> STRING
TASK <theory> STRING ignore
```

where <theory> is SCF, HF, DFT, PSPW, BAND, MP2, CCSD, TCE, etc.. The Task directive with the ignore option is recommended, otherwise NWChem will crash if the path is not optimized in the allowed maximum number of iterations.

Optional input for this module is specified within the compound directive,

```
STRING
  NBEADS <integer nbeads default 5>
  MAXITER <integer maxiter default 5>

  STEPSIZE <integer stepsize default 1.0>
  NHIST <integer nhist default 5>
  INTERPOL <integer algorithm default 1>

  FREEZE1 <logical freeze1 default .false.>
  FREEZEN <logical freezen default .false.>

  TOL <float tol default 0.00045>

  [IMPOSE]
  [HASMIDDLE]
  [XYZ_PATH <string xyzfilename>]
  [RESET]
  PRINT_SHIFT <integer print_shift default 0>
END
```

The following list describes the input for the STRING block

- **nbeads** - number of beads (or images) used to represent the path
- **maxiter** - maximum number of NEB path optimizations to be performed
- **stepsize** - value for the stepsize used in the optimization. Typically less than 1.
- **nhist** - number of histories to use for quasi-Newton optimization (algorithm =0)
- **tol** - value for the maximum gradient used to determine convergence
- **freeze1** - .true. : first bead of simulation frozen, .false. :first bead of simulation not frozen.
- **freezeN** - .true. :last bead of simulation frozen, .false. :last bead of simulation not frozen
- **interpol** - 1: linear, 2: spline, 3: Akima spline
- IMPOSE - if specified causes the initial geometries used to specify the path to be aligned with one another
- HASMIDDLE - if specified causes the initial path to use the the “midgeom” geometry to be used as the midpoint, i.e. the initial path is defined as a linear morphing from “geometry” –> “midgeom” –> “endgeom”
- XYZ_PATH - if specified the initial path is defined from the sequence of geometries contained in **xyzfilename**
- RESET - if specified causes the NEB optimization and path to be started from scratch
- **print_shift** - setting the PRINT_SHIFT directive causes the path energies and geometries to be outputed every <print_shift> steps. The current path energies are appended to the file jobname.neb_epath and the current geometries are appended to the file jobname.nebpath _"current iteration".xyz

14.4.6 Setting up the initial path

There are three different ways to define the initial path for NEB optimization.

- Linear interpolation between two geometries

The geometries in the path are defined by

$$\vec{R}^i_{xyz} = \vec{R}^1_{xyz} + \frac{i-1}{nbeads-1} (\vec{R}^{nbeads}_{xyz}-\vec{R}^{1}_{xyz}), \ for \ i=1,nbeads$$

where the starting geometry $\vec{R}^1_{xyz}$ is entered in the geometry block labeled geometry , e.g.

```
geometry nocenter noautosym noautoz
O  0.00000000    -0.02293938     0.00000000
H  0.00000000     0.55046969     0.75406534
H  0.00000000     0.55046969    -0.75406534
end
```

and the last geometry in the path \(\vec{R}^{nbeads}_{xyz}\) is entered in the geometry block label `endgeom`, e.g.

```
geometry endgeom nocenter noautosym noautoz
O  0.00000000     0.02293938     0.00000000
H  0.00000000    -0.55046969     0.75406534
H  0.00000000    -0.55046969    -0.75406534
end
```

- Linear interpolation between three geometries

The geometries for this path are defined by

\[\vec{R}^i_{xyz} = \vec{R}^1_{xyz} + \frac{i-1}{nbeads/2-1} (\vec{R}^{nbeads/2}_{xyz}-\vec{R}^{1}_{xyz}), \ for \ i=1,nbeads/2\]

and

\[\vec{R}^i_{xyz} = \vec{R}^{nbeads/2}_{xyz} + \frac{i-nbeads/2}{nbeads/2-1} (\vec{R}^{nbeads}_{xyz}-\vec{R}^{nbeads/2}_{xyz}), \ for \ i=nbeads/2+1,nbeads\]

where the starting \(\vec{R}^{1}_{xyz}\) , middle \(\vec{R}^{nbeads/2}_{xyz}\) and last \(\vec{R}^{nbeads}_{xyz}\)" geometries are entered in the geometry blocks `geometry`, `midgeom` and `endgeom` respectively, e.g.

```
geometry nocenter noautosym noautoz
O  0.00000000    -0.02293938     0.00000000
H  0.00000000     0.55046969     0.75406534
H  0.00000000     0.55046969    -0.75406534
end

geometry midgeom nocenter noautosym noautoz
O  0.00000000     0.00000000     0.00000000
H  0.00000000     0.00000000     1.00000000
H  0.00000000     0.00000000    -1.00000000
end

geometry endgeom nocenter noautosym noautoz
O  0.00000000     0.02293938     0.00000000
H  0.00000000    -0.55046969     0.75406534
H  0.00000000    -0.55046969    -0.75406534
end
```

- Using `xyz_path` to explicitly input a path of geometries

The `xyz_path` option can also be used to define the initial path, e.g.

```
...
STRING
  ...
 XYZ_PATH path.xyz
END
...
```

14.4.7 String Tutorial 1:HCN –> HNC path optimization

(input:HCN-string1.nw, output:HCN-string1.nwout, datafiles: HCN-string1.string_epath.dat HCN-string1.string_final_epath.dat)

(xyzfiles: HCN-string1.stringpath_000001.xyz HCN-string1.stringpath_000005.xyz HCN-string1.stringpath_000010.xyz HCN-string1.stringpath_000020.xyz HCN-string1.stringpath_000030.xyz HCN-string1.stringpath_final.xyz)

In this example, the path energy for the reaction HCN –> HNC is calculated.

```
#
# The initial path has the Carbon moving through the Nitrogen.
# So for this simulation to work that atom avoidance code needs to work.
# Because the initial path is so stiff the wavefunction optimizer needs to requires
# lots of iterations during the early stages of the path optimization.
#
#
Title "HCN --> HNC Zero-Temperature String Simulation"
echo
start hcn-hnc-dft

geometry noautoz noautosym
C        0.00000000     0.00000000    -0.49484657
N        0.00000000     0.00000000     0.64616359
H        0.00000000     0.00000000    -1.56151539
end

geometry endgeom  noautoz noautosym
C        0.00000000     0.00000000     0.73225318
N        0.00000000     0.00000000    -0.42552059
H        0.00000000     0.00000000    -1.42351006
end

### Gaussian DFT ###
basis
* library 3-21G
end

dft
  xc b3lyp
  maxiter 501
end

string
  nhist 10
  nbeads 10
  maxiter 10
  stepsize 0.10
  print_shift 1

  # don't allow the end points of the path to move
  freeze1 .true.
  freezeN .true.
end
task dft string ignore

string
  # increase the number of images
  nbeads 20
  maxiter 20

  # allow the end points of the path to move
  freeze1 .false.
  freezeN .false.
end
task dft string ignore
```

After each optimization step the path energies are outputed as follows

```
 string: Path Energy #                    2
 string:                    1  -92.906682492969779
 string:                    2  -92.743446565848473
 string:                    3  -92.751945829987775
 string:                    4  -92.756507971834026
 string:                    5  -92.726984154346979
 string:                    6  -92.701651474021503
 string:                    7  -92.672613497521183
 string:                    8  -92.825096796032099
 string:                    9  -92.716422030970662
 string:                   10  -92.881713271394148
```

Another way to keep track of the optimization process is to run the following grep command on the output file.

```
[WE24397:NWChem/NEB/Example2] bylaska% grep @ HCN-dft.out
@zts
@zts String method.
@zts Temperature         =   0.00000
@zts Covergence Tolerance =   0.00010
@zts Step Size           =   0.10000
@zts Maximum Time Steps   =       10
@zts Number of replicas   =       10
```

```
@zts Number of histories  =       10
@zts String Interpolator  =        1
@zts First Replica        = frozen
@zts Last Replica         = frozen
@zts
@zts  Step     xrms     xmax       E start      E middle         E end         E max      E average
@zts     1 0.460700 2.602234   -92.9066825   -83.4767173   -92.8817133   -83.4767173   -91.6169775
@zts     2 0.862226 5.405612   -92.9066825   -92.3028437   -92.8817133   -92.3028437   -92.6631831
@zts     3 0.105285 0.530157   -92.9066825   -92.3289676   -92.8817133   -92.3289676   -92.6702949
@zts     4 0.134687 0.740991   -92.9066825   -92.3512584   -92.8817133   -92.3512584   -92.6821949
@zts     5 0.117113 0.916210   -92.9066825   -92.3767826   -92.8817133   -92.3767826   -92.6899234
@zts     6 0.124464 0.844439   -92.9066825   -92.4195957   -92.8817133   -92.4195957   -92.7045117
@zts     7 0.092105 0.731434   -92.9066825   -92.4510785   -92.8817133   -92.4510785   -92.7156403
@zts     8 0.049227 0.330651   -92.9066825   -92.4690983   -92.8817133   -92.4690983   -92.7288274
@zts     9 0.032819 0.177356   -92.9066825   -92.4827444   -92.8817133   -92.4827444   -92.7344806
@zts    10 0.076249 0.444246   -92.9066825   -92.4930430   -92.8817133   -92.4930430   -92.7381477
@zts The string calculation failed to converge
@zts Bead number    1  Potential Energy =     -92.906682487840
@zts Bead number    2  Potential Energy =     -92.850640135623
@zts Bead number    3  Potential Energy =     -92.819370566454
@zts Bead number    4  Potential Energy =     -92.680821335407
@zts Bead number    5  Potential Energy =     -92.505231918657
@zts Bead number    6  Potential Energy =     -92.493042984646
@zts Bead number    7  Potential Energy =     -92.637367419044
@zts Bead number    8  Potential Energy =     -92.775376312982
@zts Bead number    9  Potential Energy =     -92.831230727986
@zts Bead number   10  Potential Energy =     -92.881713271394
@zts
@zts String method.
@zts Temperature          =   0.00000
@zts Covergence Tolerance =   0.00010
@zts Step Size            =   0.10000
@zts Maximum Time Steps   =       20
@zts Number of replicas   =       20
@zts Number of histories  =       10
@zts String Interpolator  =        1
@zts First Replica        = moves
@zts Last Replica         = moves
@zts
@zts  Step     xrms     xmax       E start      E middle         E end         E max      E average
@zts     1 1.039809 5.039486   -92.9071472   -92.4998400   -92.8820628   -92.4998400   -92.7500136
@zts     2 0.192562 0.999019   -92.9073958   -92.5259828   -92.8821500   -92.5259828   -92.7624061
@zts     3 0.244943 1.236459   -92.9075306   -92.5735140   -92.8821223   -92.5735140   -92.7816692
@zts     4 0.207031 1.093667   -92.9075888   -92.6229190   -92.8821177   -92.6154678   -92.7979112
@zts     5 0.056648 0.293829   -92.9075975   -92.6672565   -92.8821033   -92.6507897   -92.8101666
@zts     6 0.078950 0.555245   -92.9076044   -92.7245122   -92.8822536   -92.7014407   -92.8241914
@zts     7 0.065564 0.521110   -92.9076101   -92.7539982   -92.8822915   -92.7376310   -92.8326007
@zts     8 0.050188 0.319477   -92.9076113   -92.7695725   -92.8824219   -92.7612604   -92.8378464
@zts     9 0.055301 0.322130   -92.9076168   -92.7754581   -92.8825732   -92.7740099   -92.8408900
@zts    10 0.038769 0.195102   -92.9076177   -92.7775695   -92.8826652   -92.7775695   -92.8425440
@zts    11 0.064900 0.273480   -92.9076215   -92.7800330   -92.8827175   -92.7800330   -92.8443574
@zts    12 0.062593 0.266337   -92.9076224   -92.7823972   -92.8826993   -92.7823972   -92.8458976
@zts    13 0.205437 0.948190   -92.9076243   -92.7842034   -92.8826408   -92.7842034   -92.8469810
@zts    14 0.015025 0.068924   -92.9076247   -92.7844362   -92.8826536   -92.7844362   -92.8472227
@zts    15 0.129208 0.602636   -92.9076254   -92.7849856   -92.8826676   -92.7849856   -92.8477169
@zts    16 0.013479 0.056561   -92.9076260   -92.7855201   -92.8826783   -92.7855201   -92.8481626
@zts    17 0.472858 2.220715   -92.9076271   -92.7878088   -92.8826913   -92.7878088   -92.8497919
@zts    18 0.162617 0.766201   -92.9076273   -92.7879912   -92.8826934   -92.7879912   -92.8499197
@zts    19 0.013204 0.060562   -92.9076276   -92.7885097   -92.8826994   -92.7885097   -92.8502675
@zts    20 0.718205 3.423813   -92.9076278   -92.7905066   -92.8827009   -92.7895258   -92.8514863
@zts The string calculation failed to converge
@zts Bead number    1  Potential Energy =     -92.907627751439
@zts Bead number    2  Potential Energy =     -92.905047596626
@zts Bead number    3  Potential Energy =     -92.897944354806
@zts Bead number    4  Potential Energy =     -92.887494117302
@zts Bead number    5  Potential Energy =     -92.874059841858
@zts Bead number    6  Potential Energy =     -92.857382758537
@zts Bead number    7  Potential Energy =     -92.837207959079
@zts Bead number    8  Potential Energy =     -92.815902497386
@zts Bead number    9  Potential Energy =     -92.798474907121
@zts Bead number   10  Potential Energy =     -92.789525765222
@zts Bead number   11  Potential Energy =     -92.790506632257
@zts Bead number   12  Potential Energy =     -92.799861168980
@zts Bead number   13  Potential Energy =     -92.814252430183
@zts Bead number   14  Potential Energy =     -92.830704548760
@zts Bead number   15  Potential Energy =     -92.847248091296
@zts Bead number   16  Potential Energy =     -92.861557132126
@zts Bead number   17  Potential Energy =     -92.871838446832
@zts Bead number   18  Potential Energy =     -92.878543965696
@zts Bead number   19  Potential Energy =     -92.881844751735
@zts Bead number   20  Potential Energy =     -92.882700859222
```

A plotting program (e.g. gnuplot, xmgrace) can be used to look at final path as well as the the convergence of the path i.e.,

```
[WE24397:NEB/Example2/perm] bylaska% gnuplot

        G N U P L O T
        Version 4.6 patchlevel 0    last modified 2012-03-04
        Build System: Darwin x86_64

        Copyright (C) 1986-1993, 1998, 2004, 2007-2012
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     <http://www.gnuplot.info>
        faq, bugs, etc:   type "help FAQ"
        immediate help:   type "help"  (plot window: hit 'h')

Terminal type set to 'aqua'
gnuplot> set xlabel "Reaction Coordinate"
gnuplot> set ylabel "Energy (kcal/mol)"
gnuplot> set yrange [0:100]
gnuplot> set grid
gnuplot> set style data linespoints
gnuplot> plot "hcn-hnc-dft.string_epath" using 1:($2+92.908)*27.2116*23.06,"hcn-hnc-dft.string_final_epath" using 1:($2+92.908)*27.2116*23.06
gnuplot>
```

400px

14.4.8 String Tutorial 2:

```
Title "2SiO4H4 --> H3O3Si-O-SiO3H3 + H2O"
echo
start sio4h4-dimer

geometry noautoz noautosym
Si       -3.90592      -0.11789       0.03791
O        -2.32450      -0.24327      -0.05259
O        -4.45956      -1.13247       1.13159
O        -4.53584      -0.45118      -1.38472
O        -4.28179       1.37363       0.44838
Si        1.27960       0.06912       0.14555
O         2.85122       0.23514       0.32761
O         0.54278       0.38513       1.52092
O         0.94484      -1.42248      -0.29913
O         0.75605       1.07390      -0.97272
H        -1.66762      -0.74425      -0.29362
H        -4.05734       2.06481       0.90983
H        -4.30983      -1.85807       1.57116
H        -4.43621      -0.88060      -2.12508
H         3.59374      -0.16315       0.50572
H         0.36896       0.10990       2.31839
H         0.53993      -2.15495      -0.09488
H         0.43207       1.85525      -1.13531
end

geometry endgeom  noautoz noautosym
Si       -3.07373       0.18232      -0.24945
O        -1.50797       0.23823      -0.53062
O        -3.36758      -0.93058       0.85023
O        -3.83958      -0.20093      -1.59101
O        -3.57993       1.59735       0.27471
Si       -0.05186       0.25441       0.11277
O         0.94679      -0.58168      -0.80206
O        -0.10091      -0.40972       1.55838
O         1.41035      -3.75872       1.22931
O         0.47135       1.75206       0.24209
H         1.03624      -4.62405       0.92620
H        -3.81554       2.06192       0.96069
H        -3.97094      -1.38510       1.26383
H        -4.39754      -0.73964      -1.96563
H         1.45990      -0.57144      -1.49361
H        -0.44444      -0.37536       2.34765
H         2.15751      -4.00850       1.82933
H         0.77180       2.44229      -0.17616
end

nwpw
  simulation_cell
     SC 18.0
  end
  cutoff 30.0
  lmbfgs
end

string
  nhist 10
  nbeads 10
  maxiter 10
  stepsize 0.10
  print_shift 1

  # don't allow the end points of the path to move
  freeze1 .true.
  freezeN .true.
end
task pspw string ignore

string
  # increase the number of images
  nbeads 20
  maxiter 20

  # allow the end points of the path to move
  freeze1 .false.
  freezeN .false.
end
task pspw string ignore
```

14.4.9 String Tutorial 3: Combining NEB and String path optimizations

15 Electronic Structure Analysis

15.1 Properties

15.1.1 Overview

Properties can be calculated for both the Hartree-Fock and DFT wave functions. The properties that are available are:

- Natural bond analysis
- Dipole, quadrupole, and octupole moment
- Mulliken population analysis and bond order analysis
- Electrostatic potential (diamagnetic shielding) at nuclei
- Electric field and field gradient at nuclei
- Electric field gradients with relativistic effects
- Electron and spin density at nuclei
- NMR shielding (GIAO method)
- NMR hyperfine coupling (Fermi-Contact and Spin-Dipole expectation values)
- NMR indirect spin-spin coupling
- Gshift
- Response to electric and magnetic fields (static and dynamic)
- Raman
- Localization of molecular orbitals

The properties module is started when the task directive TASKproperty is defined in the user input file. The input format has the form:

```
PROPERTY
  [property keyword]
  [CENTER ((com || coc || origin || arb <real x y z>) default coc)]
END
```

Most of the properties can be computed for Hartree-Fock (closed-shell RHF, open-shell ROHF, and open-shell UHF), and DFT (closed-shell and open-shell spin unrestricted) wavefunctions. The NMR hyperfine and indirect spin-spin coupling require a UHF or ODFT wave function.

15.1.2 Vectors keyword

```
VECTORS [ (<string input_movecs >)]
```

The VECTORS directive allows the user to specify the input molecular orbital vectors for the property calculation

15.1.3 Property keywords

Each property can be requested by defining one of the following keywords:

```
NBOFILE
DIPOLE
QUADRUPOLE
OCTUPOLE
MULLIKEN
ESP
EFIELD
EFIELDGRAD
EFIELDGRADZ4
GSHIFT
ELECTRONDENSITY
HYPERFINE [<integer> number_of_atoms <integer> atom_list]
SHIELDING [<integer> number_of_atoms <integer> atom_list]
SPINSPIN [<integer> number_of_pairs <integer> pair_list]
RESPONSE [<integer> response_order <real> frequency]
AIMFILE
MOLDENFILE
LOCALIZATION
ALL
```

The `ALL` keyword generates all currently available properties.

15.1.3.1 NMR and EPR

Both the NMR shielding and spin-spin coupling have additional optional parameters that can be defined in the input. For the shielding the user can define the number of atoms for which the shielding tensor should be calculated, followed by the list of specific atom centers. In the case of spin-spin coupling the number of atom pairs, followed by the atom pairs, can be defined (i.e., `spinspin 1 1 2` will calculate the coupling for one pair, and the coupling will be between atoms 1 and 2).

For both the NMR spin-spin and hyperfine coupling the isotope that has the highest abundance and has spin, will be chosen for each atom under consideration.

15.1.3.1.1 Calculating EPR and paramagnetic NMR parameters

The following tutorial illustrates how to combine the hyperfine, gshift and shielding to calculate the EPR and paramagnetic NMR parameters of an open-shell system. All calculations are compatible with the ZORA model potential approach.

For theoretical and computational details, please refer to references[123].

15.1.3.1.2 NMR: Input Example

```
geometry  nocenter
  C    0.00000000    0.00000000    0.00000000
  O    1.18337200    0.00000000    0.00000000
  H    -.63151821    0.94387462    0.00000000
end

basis
 "*" library 6-311G**
end
property
 efieldgradz4 1 3
 shielding 2 1 2
 hyperfine 2 1 3
 gshift
end

relativistic
  zora on
  zora:cutoff_NMR 1d-8
  zora:cutoff 1d-30
end

dft
mult 2
xc becke88 perdew86
end

task dft property
```

15.1.3.2 Electric field

- efield computes the electric field.
- efieldgrad computes the electric field gradient.
- efieldgradZ4 computes the electric field gradient based on the two-component relativistic zeroth-order regular approximation (ZORA)

The method used in NWChem to computer electric field and electric fieled gradients in described in publication[4].

15.1.3.3 CENTER: Center of expansion for multipole calculations

The user also has the option to choose the center of expansion for the dipole, quadrupole, and octupole calculations.

```
[CENTER ((com || coc || origin || arb <real x y z>) default coc)]
```

com is the center of mass, coc is the center of charge, origin is (0.0, 0.0, 0.0) and arb is any arbitrary point which must be accompanied by the coordinated to be used. Currently the x, y, and z coordinates must be given in the same units as UNITS in GEOMETRY.

15.1.3.4 Response Calculations

Response calculations can be calculated as follows:

```
property
 response  1 7.73178E-2   # response order and frequency in Hartree energy units
 velocity                 # use modified velocity gauge for electric dipole
 orbeta                   # calculate optical rotation 'beta' directly [^5]
 giao                     # GIAO optical rotation [^6][^7][^8], forces orbeta
 bdtensor                 # calculates B-tilde of Refs. [^6][^8]
 analysis                 # analyze response in terms of MOs [^8]
 damping 0.007            # complex response functions with damping, Ref [^9]
 convergence 1e-4         # set CPKS convergence criterion (default 1e-4)
end
```

Response calculations are currently supported only for

- order 1 (linear response),
- single frequency,
- electric field,
- mixed electric-magnetic field perturbations.

The output consists of the electric polarizability and optical rotation tensors (alpha, beta for optical rotation) in atomic units.
The response keyword requires two arguments: response order and frequency in Hartree energy units (the aoresponse keyword can be used with same effect as the response keyword).
If the velocity or giao keywords are absent, the dipole-length form will be used for the dipole integrals. This is a bit faster.
The isotropic optical rotation is origin independent when using the velocity gauge (by means of velocity keyword) or with GIAOs [6] (by means of the giao keyword).
With the keyword bdtensor , a fully origin-invariant optical rotation tensor is calculated[6 8].
Note that velocity and orbeta are incompatible.
The input line

```
set prop:newaoresp 0
```

outside of the properties block forces the use of an older version of the response code, which has fewer features (in particular, no working GIAO optical rotation) but which has been tested more thoroughly. In the default newer version you may encounter undocumented features (bugs).
The keyword analysis triggers an analysis of the response tensors in terms of molecular orbitals.
If the property input block also contains the keyword pmlocalization , then the analysis is performed in terms of Pipek-Mezey localized MOs, otherwise the canonical set is used (this feature may currently not work, please check the sum of the

analysis carefully). See Ref. [6] for an example. Works with HF and density functionals for which linear response kernels are implemented in NWChem.

Please refer to papers[6][5][7][1][0][9][8] for further details:

15.1.3.5 Raman

Raman calculations can be performed by specifying the Raman block. These calculations are performed in conjunction with polarizability calculations. Detailed description of input parameters at https://pubs.acs.org/doi/10.1021/jp411039m#notes-1

```
RAMAN
 [ (NORMAL | | RESONANCE) default NORMAL ]
 [ (LORENTZIAN | | GAUSSIAN) default LORENTZIAN ]
 [ LOW <double low default 0.0> ]
 [ HIGH <double high default highest normal mode> ]
 [ FIRST <integer first default 7> ]
 [ LAST < integer last default number of normal modes > ]
 [ WIDTH <double width default 20.0> ]
 [ DQ <double dq default 0.01> ]
END
task dft raman
```

or

```
task dft raman numerical
```

Sample input block:

```
property
 response 1 8.8559E-2
 damping 0.007
end
raman
 normal
 lorentzian
end
```

15.1.3.5.1 Raman Keywords

- NORMAL and RESONANCE : Type of Raman plot to make.
- LORENTZIAN and GAUSSIAN : Generation of smoothed spectra (rather than sticks) using either a Lorentzian function or a Gaussian function. The default is LORENTZIAN .
- LOW and HIGH : The default range in which to generate the Raman spectrum plot is (0.0, highest wavenumber normal mode) cm-1. The LOW and HIGH keywords modify the frequency range.
- FIRST and LAST : The default range of indices of normal modes used in the plot is (7, number of normal modes). The FIRST and LAST keywords modify the range of indices.
- WIDTH : Controls the width in the smoothed peaks, using Lorentzians or Gaussians, in the plot. The default value for WIDTH is 20.0.
- DQ : Size of the steps along the normal modes. The default value for DQ is 0.01. It is related to the step size dR used in numerical evaluation of polarizability derivative

15.1.3.5.2 Raman Output

Raman spectrum in stick format and smoothed using Lorentzians or Gaussians stored in a filename with format [fname].normal .

The number of points is 1000 by default. This value can be changed by adding the following SET directive to the input file

```
set raman:numpts <integer>
```

15.1.3.5.3 Raman References

Please refer to papers[11][12] for further details:

15.1.3.6 Polarizability computed with the Sum over Orbitals method

As an alternative to the linear response method, the Sum over Orbitals (SOO) method is available to compute polarizabilities. Results of these method are much less accurate than linear response calculations, with values off by a factor of 2-4x. However, the qualitative nature of this results can be used to compute Raman frequencies when coupled with QMD, as described in references [13][14].

Sample input computing polarizability both with the SOO method and the linear response method:

```
property
 polfromsos
end

task dft property

property
 response 1 0
end
task dft property
```

15.1.3.7 Nbofile

The keyword NBOFILE does not execute the Natural Bond Analysis code, but simply creates an input file to be used as input to the stand-alone NBO code. All other properties are calculated upon request.

Following the successful completion of an electronic structure calculation, a Natural Bond Orbital (NBO) analysis may be carried out by providing the keyword NBOFILE in the PROPERTY directive. NWChem will query the rtdb and construct an ASCII file, *file_prefix*.gen, that may be used as input to the stand alone version of the NBO program, GenNBO*file_prefix* is equal to string following the START directive. The input deck may be edited to provide additional options to the NBO calculation, (see the NBO user's manual for details.)

Users that have their own NBO version can compile and link the code into the NWChem software. See the INSTALL file in the source for details.

15.1.3.8 Aimfile

This keyword generates AIM Wavefunction files. The resulting AIM wavefunction file (.wfn/.wfx) can be post-processed with a variety of codes, e.g.

- XAIM
- NCIPLOT
- Multiwfn
- Postg

WARNING: Since we have discovered issues in generating .WFN files with this module (e.g. systems with ECPs), the recommended method for generating .WFN file is to first generate a Molden file with the Moldenfile option, then convert the Molden file into a WFN file by using the Molden2AIM program.

15.1.3.9 Moldenfile

```
MOLDENFILE
MOLDEN_NORM (JANPA | | NWCHEM || NONE)
```

This keyword generates files using the Molden format. The resulting Molden file (.molden) should compatible with a variety of codes that can input Molden files, e.g.

- Molden
- JANPA (the nwchem2molden step is no longer required when using .molden files and the `MOLDEN_NORM JANPA` keyword)
- orbkit
- Molden2qmc
- Molden2AIM
- Multiwfn

the `MOLDEN_NORM` option allows the renormalization of the basis set coefficients. By default, the coefficient values from input are not modified. Using the `JANPA` value coefficients are normalized following JANPA's convention (where basis coefficients are normalized to unity), while the `NWCHEM` will produce coefficients normalized according to NWChem's convention. Using `MOLDEN_NORM` equal `NONE` will leave the input coefficients unmodified.
It is strongly recommended to use **spherical** basis set when using the NWChem Molden output for JANPA analysis

Example input file for a scf calculation. The resulting Molden file will be named `h2o.molden`

```
start heat

geometry; he 0. 0. 0.; end

basis spherical; * library 6-31g ; end

task scf

property
 vectors heat.movecs
 moldenfile
 molden_norm janpa
end

task scf property
```

Then, the resulting `h2o.molden` file can be post processed by Janpa with the following command

```
java -jar janpa.jar h2o.molden > h2o.janpa.txt
```

15.1.3.10 Localization

Localized molecular orbitals (LMOs) can be computed with the `localization` keyword.

```
property
 localization (( pm || boys || ibo) default pm) ((0 || 1 || 2) default 0)
end
```

The following methods are available:

- Pipek-Mezey[15], `pm` keyword (default)
- Foster-Boys[16], `boys` keyword
- IAO/IBO[17][18], `ibo` keyword

An optional integer switch may be supplied after the localization method specification. The default `0` means only occupied orbitals are localized. Use `1` to localize virtual orbitals instead, or `2` to separately localize occupied and virtual orbitals in the same run. Localization of virtual orbitals is not implemented for the Boys localization. If only the keyword `localization` is given, this defaults to a Pipek-Mezey localization of the occupied molecular orbitals.

In spin-unrestricted calculations, alpha and beta spin orbitals are localized separately. Fractional-occupation DFT and restricted open-shell Hartree-Fock or Kohn-Sham are not yet supported; attempting to use localization with fractional orbital occupations may run but will likely produce unwanted and wrong results.

A localization run prints a characterizations of the LMOs. Specifically, the PM and IBO localizations print an energy-sorted list of LMOs indicating the density weight of each LMO on the various atoms for weights exceeding a 1% threshold; the

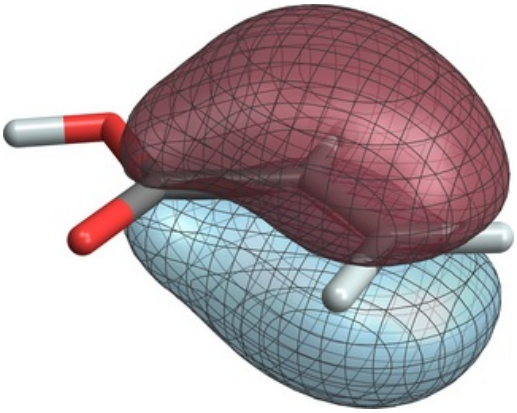
atoms are listed in terms of their input number. A Boys localization prints the LMO centroid and the expectation value of *r*^2 per LMO.

Typically, however, the user will want to visualize the LMOs. The localization run saves the LMO coefficients in a file `locorb.movecs`, which can be used as the MO vector file in an NWChemDPLOT task to generate corresponding volume data maps, for example, in the `cube' format commonly used in computational chemistry. The latter can be visualized with a considerable number of open source, free, or proprietary software packages.

NWChem test jobs that use the orbital localization functionality (under `$NWCHEM_TOP/QA/tests/`) are `localize-ibo-allyl`, `localize-pm-allyl`, and `localize-ibo-aa`. The latter is based on the acrylic acid example from the original IBO publication.[17] IBO number 19 represents the delocalized C-C pi bond LMO of the system. After running the test job, the corresponding cube file for LMO 19 can be generated, for example, with

```
dplot
vectors locorb.movecs
limitxyz
-4.3334997 4.9318898 50
-3.3065879000000002 3.718637 50
-2.0 2.0 50
gaussian
spin alpha
orbitals view;1;19
output localize-ibo-aa-00019.cube
end
task dplot
```

The resulting plot with ±0.03 isosurfaces may look like the following



A Python script is available to facilitate the creation of volume data files for visualizing multiple MOs or LMOs of a given molecule. For the acrylic acid example, after running NWChem with the `localize-ibo-aa.nw` input, the following command will create an input file with `dplot` tasks to generate cubes for the 19 occupied and 10 virtual IBOs of the molecule on a 50x50x50 grid:

```
dplot.py -i localize-ibo-aa.nw -m locorb.movecs -g 50 -l 1-29
```

The `dplot` task shown above is included in the resulting NWChem input file `dplot.nw`.

15.1.4 Gaussian Cube Files

Electrostatic potential (keyword `esp`) and the magnitude of the electric field (keyword `efield`) on the grid can be generated in the form of the Gaussian Cube File. This behavior is triggered by the inclusion of grid keyword as shown below

```
grid [pad dx [dy dz]] [rmax x y z] [rmin x y z] [ngrid nx [ny nz]] [output filename]
```

where

- `pad dx [dy dz]` - specifies amount of padding (in angstroms) in x,y, and z dimensions that will be applied in the automatic construction of the rectangular grid volume based on the geometry of the system. If only one number is provided then

the same amount of padding will be applied in all dimensions. The default setting is 4 angstrom padding in all dimensions.

- `rmin x y z` - specifies the coordinates (in angstroms) of the minimum corner of the rectangular grid volume. This will override any padding in this direction.
- `rmax x y z` - specifies the coordinates (in angstroms) of the maximum corner of the rectangular grid volume. This will override any padding in this direction.
- `ngrid nx [ny nz]` - specifies number of grid points along each dimension. If only one number is provided then the same number of grid points are assumed all dimensions. In the absence of this directive the number of grid points would be computed such that grid spacing will be close to 0.2 angstrom, but not exceeding 50 grid points in either dimension.
- `output filename` - specifies name of the output cube file. The default behavior is to use *prefix* `-elp.cube` or *prefix* `-elf.cube` file names for electrostatic potential or electric field respectively. Here *prefix* denotes the system name as specified in start directive. Note that Gaussian cube files will be written in the run directory (where the input file resides).

Example input file

```
echo
start nacl


geometry nocenter noautoz noautosym
 Na                -0.00000000     0.00000000    -0.70428494
 Cl                0.00000000    -0.00000000     1.70428494
end


basis
  * library 6-31g*
end

#electric field would be written out to nacl.elf.cube file
#with
#ngrid     : 20 20 20
#rmax      : 4.000     4.000     5.704
#rmin      :-4.000    -4.000    -4.704

property
efield
grid pad 4.0 ngrid 20
end

task dft property

#electrostatic potential would be written to esp-pad.cube file
# with the same parameters as above

property
esp
grid pad 4.0 ngrid 20 output esp-pad.cube
end

task dft property

#illustrating explicit specification of minumum box coordinates

property
esp
grid pad 4.0 rmax 4.000 4.000 5.704 ngrid 20
end

task dft property
```

15.1.5 References

1. Autschbach, J.; Patchkovskii, S.; Pritchard, B. Calculation of Hyperfine Tensors and Paramagnetic NMR Shifts Using the Relativistic Zeroth-Order Regular Approximation and Density Functional Theory. *Journal of Chemical Theory and Computation* **2011**, *7* (7), 2175–2188. https://doi.org/10.1021/ct200143w
2. Aquino, F.; Pritchard, B.; Autschbach, J. Scalar Relativistic Computations and Localized Orbital Analyses of Nuclear Hyperfine Coupling and Paramagnetic NMR Chemical Shifts. *Journal of Chemical Theory and Computation* **2012**, *8* (2), 598–609. https://doi.org/10.1021/ct2008507
3. Aquino, F.; Govind, N.; Autschbach, J. Scalar Relativistic Computations of Nuclear Magnetic Shielding and g-Shifts with the Zeroth-Order

Regular Approximation and Range-Separated Hybrid Density Functionals. *Journal of Chemical Theory and Computation* **2011**, *7* (10), 3278–3292. https://doi.org/10.1021/ct200408j

4. Aquino, F.; Govind, N.; Autschbach, J. Electric Field Gradients Calculated from Two-Component Hybrid Density Functional Theory Including Spin−orbit Coupling. *Journal of Chemical Theory and Computation* **2010**, *6* (9), 2669–2686. https://doi.org/10.1021/ct1002847
5. Autschbach. Computation of Optical Rotation Using Timedependent Density Functional Theory. *Computing Letters* **2007**, *3* (2), 131–150. https://doi.org/10.1163/157404007782913327
6. Autschbach, J. Time-Dependent Density Functional Theory for Calculating Origin-Independent Optical Rotation and Rotatory Strength Tensors. *ChemPhysChem* **2011**, *12* (17), 3224–3235. https://doi.org/10.1002/cphc.201100225
7. Krykunov, M.; Autschbach, J. Calculation of Optical Rotation with Time-Periodic Magnetic-Field-Dependent Basis Functions in Approximate Time-Dependent Density-Functional Theory. *The Journal of Chemical Physics* **2005**, *123* (11), 114103. https://doi.org/10.1063/1.2032428
8. Moore, B.; Srebro, M.; Autschbach, J. Analysis of Optical Activity in Terms of Bonds and Lone-Pairs: The Exceptionally Large Optical Rotation of Norbornenone. *Journal of Chemical Theory and Computation* **2012**, *8* (11), 4336–4346. https://doi.org/10.1021/ct300839y
9. Krykunov, M.; Kundrat, M. D.; Autschbach, J. Calculation of Circular Dichroism Spectra from Optical Rotatory Dispersion, and Vice Versa, as Complementary Tools for Theoretical Studies of Optical Activity Using Time-Dependent Density Functional Theory. *The Journal of Chemical Physics* **2006**, *125* (19), 194110. https://doi.org/10.1063/1.2363372
10. Hammond, J. R.; Govind, N.; Kowalski, K.; Autschbach, J.; Xantheas, S. S. Accurate Dipole Polarizabilities for Water Clusters n=212 at the Coupled-Cluster Level of Theory and Benchmarking of Various Density Functionals. *The Journal of Chemical Physics* **2009**, *131* (21), 214103. https://doi.org/10.1063/1.3263604
11. Mullin, J. M.; Autschbach, J.; Schatz, G. C. Time-Dependent Density Functional Methods for Surface Enhanced Raman Scattering (SERS) Studies. *Computational and Theoretical Chemistry* **2012**, *987*, 32–41. https://doi.org/10.1016/j.comptc.2011.08.027
12. Aquino, F. W.; Schatz, G. C. Time-Dependent Density Functional Methods for Raman Spectra in Open-Shell Systems. *The Journal of Physical Chemistry A* **2014**, *118* (2), 517–525. https://doi.org/10.1021/jp411039m
13. Fischer, S. A.; Ueltschi, T. W.; El-Khoury, P. Z.; Mifflin, A. L.; Hess, W. P.; Wang, H.-F.; Cramer, C. J.; Govind, N. Infrared and Raman Spectroscopy from Ab Initio Molecular Dynamics and Static Normal Mode Analysis: The C-H Region of DMSO as a Case Study. *The Journal of Physical Chemistry B* **2015**, *120* (8), 1429–1436. https://doi.org/10.1021/acs.jpcb.5b03323
14. Aprà, E.; Bhattarai, A.; Baxter, E.; Wang, S.; Johnson, G. E.; Govind, N.; El-Khoury, P. Z. Simplified Ab Initio Molecular Dynamics-Based Raman Spectral Simulations. *Applied Spectroscopy* **2020**, *74* (11), 1350–1357. https://doi.org/10.1177/0003702820923392
15. Pipek, J.; Mezey, P. G. A Fast Intrinsic Localization Procedure Applicable for Ab Initio and Semiempirical Linear Combination of Atomic Orbital Wave Functions. *The Journal of Chemical Physics* **1989**, *90* (9), 4916–4926. https://doi.org/10.1063/1.456588
16. Foster, J. M.; Boys, S. F. Canonical Configurational Interaction Procedure. *Reviews of Modern Physics* **1960**, *32* (2), 300–302. https://doi.org/10.1103/revmodphys.32.300
17. Knizia, G. Intrinsic Atomic Orbitals: An Unbiased Bridge Between Quantum Theory and Chemical Concepts. *Journal of Chemical Theory and Computation* **2013**, *9* (11), 4834–4843. https://doi.org/10.1021/ct400687b
18. Knizia, G.; Klein, J. E. M. N. Electron Flow in Reaction Mechanisms-Revealed from First Principles. *Angewandte Chemie International Edition* **2015**, *54* (18), 5518–5522. https://doi.org/10.1002/anie.201410637

15.2 Electrostatic potentials

15.2.1 Overview

The NWChem Electrostatic Potential (ESP) module derives partial atomic charges that fit the quantum mechanical electrostatic potential on selected grid points.

The ESP module is specified by the NWChem task directive

```
task esp
```

The input for the module is taken from the ESP input block

```
ESP
 ...
END
```

15.2.2 Grid specification

The grid points for which the quantum mechanical electrostatic potential is evaluated and used in the fitting procedure of

the partial atomic charges all lie outside the van der Waals radius of the atoms and within a cutoff distance from the atomic centers. The following input parameters determine the selection of grid points.

- If a grid file is found, the grid will be read from that file. If no grid file is found, or the keyword

  recalculate

is given, the grid and the electrostatic potential is recalculated.

- The extent of the grid is determined by the range keyword

  range <real rcut>

where rcut is the maximum distance in nm between a grid point and any of the atomic centers. When omitted, a default value for rcut of 0.3 nm is used.

- The grid spacing is specified by the spacing keyword

  spacing <real spac>

where spac is the grid spacing in *nm* for the regularly spaced grid points. If not specified, a default spacing of 0.05 nm is used.

- The van der Waals radius of an element can be specified by

  radius <integer iatnum> <real atrad>

where iatnum is the atomic number for which a van der Waals radius of atrad in nm will be used in the grid point determination.
Default values will be used for atoms not specified.

- The probe radius in nm determining the envelope around the molecule is specified by

  probe <real probe default 0.07>

- The distance between atomic center and probe center can be multiplied by a constant factor specified by

  factor <real factor default 1.0>

All grid points are discarded that lie within a distance factor*(radius(i)+probe) from any atom **i**.

- Schwarz screening is applied using

  screen [<real scrtol default 1.0D-5>]

15.2.3 Constraints

Additional constraints to the partial atomic charges can be imposed during the fitting procedure.

- The net charge of a subset of atoms can be constrained using

  constrain <real charge {<integer iatom>}

where charge is the net charge of the set of atoms {iatom} . A negative atom number iatom can be used to specify that the partial charge of that atom is substracted in the sum for the set.

- The net charge of a sequence of atoms can be constrained using

```
constrain <real charge> <integer iatom> through <integer jatom>
```

where charge is the net charge of the set of atoms {[iatom:jatom]} .

- A group of atoms can be constrained to have the same charge with

```
constrain equal {<integer iatom>}
```

- The individual charge of a group of atoms can be constrained to be equal to those of a second group of atoms with

```
constrain group  <integer iatom> <integer jatom>  to  <integer katom> <integer latom>
```

resulting in the same charge for atoms iatom and katom, for atoms iatom+1 and k atom+1, … for atoms jatom and latom.

- A special constraint

```
constrain xhn  <integer iatom> {<integer jatom>}
```

can be used to constrain the set {iatom,{jatom}} to zero charge, and constrain all atoms in {jatom} to have the same charge. This can be used, for example, to restrain a methyl group to zero charge, and have all hydrogen carrying identical charges.

15.2.4 Restraints

Restraints can be applied to each partial charge using the RESP charge fitting procedure.

- The directive for charge restraining is

```
restrain [hfree] (harmonic [<real scale>] | \
hyperbolic [<real scale> [<real tight>]]  \
 [maxiter <integer maxit>]  [tolerance <real toler>])
```

Here hfree can be specified to exclude hydrogen atoms from the restraining procedure. Variable scale is the strength of the restraint potential, with a default of 0.005 au for the harmonic restraint and a default value of 0.001 au for the hyperbolic restraint. For the hyperbolic restraints the tightness tight can be specified to change the default value of 0.1 e. The iteration count that needs to be carried out for the hyperbolic restraint is determined by the maximum number of allowed iterations maxiter , with a default value of 25, and the tolerance in the convergence of the partial charges toler , with a default of 0.001 e.

15.3 DPLOT

15.3.1 Overview

```
DPLOT
  ...
END
```

This directive is used to obtain the plots of various types of electron densities (or orbitals) of the molecule. The electron density is calculated on a specified set of grid points using the molecular orbitals from SCF or DFT calculation. The output file is either in MSI Insight II contour format (default) or in the Gaussian Cube format. DPLOT is not executed until the task dplot directive is given. Different sub-directives are described below.

The implementation of the dplot functionality uses mostly local memory. The quantities stored in local memory are:

- the positions of the grid points
- the property of interest expressed at the grid points

The stack memory setting in the input file must be sufficient to hold these quantities in local memory on a processor.

15.3.2 GAUSSIAN: Gaussian Cube format

```
GAUSSIAN
```

A outputfile is generate in Gaussian Cube format. You can visualize this file using gOpenMol (after converting the Gaussian Cube file with gcube2plt), Molden or Molekel.

15.3.3 TITLE: Title directive

```
TITLE <string Title default Unknown Title>
```

This sub-directive specifies a title line for the generated input to the Insight program or for the Gaussian cube file. Only one line is allowed.

15.3.4 LIMITXYZ: Plot limits

```
LIMITXYZ [units <string Units default angstroms>]
 <real X_From> <real X_To> <integer No_Of_Spacings_X>
  <real Y_From> <real Y_To> <integer No_Of_Spacings_Y>
  <real Z_From> <real Z_To> <integer No_Of_Spacings_Z>
```

This sub-directive specifies the limits of the cell to be plotted. The grid is generated using No_Of_Spacings + 1 points along each direction. The known names for Units are angstroms, au and bohr.

15.3.5 SPIN: Density to be plotted

```
SPIN <string Spin default total>
```

This sub-directive specifies what kind of density is to be computed. The known names for Spin are `total` , `alpha` , `beta` and `spindens` , the last being computed as the difference between α and β electron densities.

15.3.6 OUTPUT: Filename

```
OUTPUT <string File_Name default dplot>
```

This sub-directive specifies the name of the generated input to the Insight program or the generated Gaussian cube file. The name `OUTPUT` is reserved for the standard NWChem output.

15.3.7 VECTORS: MO vector file name

```
VECTORS <string File_Name default movecs> [<string File_Name2>]
```

This sub-directive specifies the name of the molecular orbital file. If the second file is optionally given the density is computed as the difference between the corresponding electron densities. The vector files have to match.

15.3.8 WHERE: Density evaluation

```
WHERE <string Where default grid>
```

This sub-directive specifies where the density is to be computed. The known names for `where` are `grid` (the calculation of the density is performed on the set of a grid points specified by the sub-directive `LimitXYZ` and the file specified by the sub-

directive Output is generated), nuclei (the density is computed at the position of the nuclei and written to the NWChem output) and g+n (both).

15.3.9 ORBITAL: Orbital sub-space

```
ORBITALS [<string Option default density>]
<integer No_Of_Orbitals>
<integer Orb_No_1 Orb_No_2 ...>
```

This sub-directive specifies the subset of the orbital space for the calculation of the electron density. The density is computed using the occupation numbers from the orbital file modified according to the spin directive. If the contours of the orbitals are to be plotted Option should be set to view (instead of the implicit default value density). Note, that in this case No_Of_Orbitals should be set to 1 and sub-directive where is automatically set to grid. Also specification of two orbital files conflicts with the view option. alpha orbitals are always plotted unless spin is set to beta.

15.3.10 CIVECS: CI vectors

```
CIVECS [<string Name of civecs file>]
```

15.3.11 DENSMAT: Density matrix

```
DENSMAT [<string Name of density matrix file>]
```

15.3.12 Examples

15.3.12.1 Charge Density

15.3.12.2 Example of HF charge density plot (with Gaussian Cube output)

```
start n2
geometry
  n  0 0   0.53879155
  n  0 0  -0.53879155
end
basis;  n library cc-pvdz;end
scf
vectors  output n2.movecs
end
dplot
  TITLE HOMO
  vectors n2.movecs
   LimitXYZ
 -3.0 3.0 10
 -3.0 3.0 10
 -3.0  3.0  10
  spin total
  gaussian
  output chargedensity.cube
end
task scf
task dplot
```

15.3.12.3 Example of CCSD charge density plot (with Gaussian Cube output)

```
start n2
geometry
 n 0 0  0.53879155
 n 0 0 -0.53879155
 symmetry c2v
end
basis; n library cc-pvdz;end
tce
 ccsd
 densmat n2.densmat
end
task tce energy
dplot
 TITLE HOMO
 LimitXYZ
 -3.0 3.0 10
 -3.0 3.0 10
 -3.0 3.0 10
 spin total
 gaussian
 densmat n2.densmat
 output ccsddensity.cube
end
task dplot
```

15.3.12.4 Molecular Orbital

Example of orbital plot (with Insight II contour output):

```
start n2
geometry
  n  0 0   0.53879155
  n  0 0  -0.53879155
end
basis;  n library cc-pvdz;end
scf
vectors  output n2.movecs
end
dplot
  TITLE HOMO
  vectors n2.movecs
   LimitXYZ
 -3.0 3.0 10
 -3.0 3.0 10
 -3.0  3.0  10
  spin total
  orbitals view; 1; 7
  output homo.grd
end
task scf
task dplot
```

15.3.12.5 Transition Density

TDDFT calculation followed by a calculation of the transition density for a specific excited state using the DPLOT block

```
echo
start h2o-td
title h2o-td
memory stack 400 mb heap 50 mb global 350 mb
charge 0
geometry units au noautoz nocenter
symmetry group c1
 O    0.00000000000000      0.00000000000000      0.00000000000000
 H    0.47043554760291      1.35028113274600      1.06035416576826
 H   -1.74335410533480     -0.23369304784300      0.27360785442967
end
basis "ao basis" print
 H    S
    13.0107010             0.19682158E-01
     1.9622572             0.13796524
     0.44453796            0.47831935
 H    S
     0.12194962            1.0000000
 H    P
     0.8000000             1.0000000
 O    S
  2266.1767785            -0.53431809926E-02
   340.87010191           -0.39890039230E-01
    77.363135167          -0.17853911985
    21.479644940          -0.46427684959
     6.6589433124         -0.44309745172
 O    S
     0.80975975668         1.0000000
 O    S
     0.25530772234         1.0000000
 O    P
    17.721504317           0.43394573193E-01
     3.8635505440          0.23094120765
     1.0480920883          0.51375311064
 O    P
     0.27641544411         1.0000000
 O    D
     1.2000000             1.0000000
end
dft
 xc bhlyp
 grid fine
 direct
 convergence energy 1d-5
end
tddft
 rpa
 nroots 5
 thresh 1d-5
 singlet
 notriplet
 civecs
end
task tddft energy
dplot
 civecs h2o-td.civecs_singlet
 root 2
 LimitXYZ
  -3.74335 2.47044 50
  -2.23369 3.35028 50
  -2 3.06035 50
   gaussian
   output root-2.cube
end
task dplot
```

15.3.12.6 Plot the excited state density

```
echo
start tddftgrad_co_exden
geometry
 C     0.00000000     0.00000000    -0.64628342
 O     0.00000000     0.00000000     0.48264375
 symmetry c1
end
basis spherical
 * library "3-21G"
end
dft
 xc pbe0
 direct
end
tddft
 nroots 3
 notriplet
 target 1
 civecs
 grad
  root 1
 end
end
task tddft gradient
dplot
 densmat tddftgrad_co_exden.dmat
 LimitXYZ
-4.0 4.0 50
-4.0 4.0 50
-4.0 4.0 50
 gaussian
 output co_exden.cube
end
task dplot
```

16 Other-Capabilities

16.1 Electron Transfer

The NWChem electron transfer (ET) module calculates the electronic coupling energy (also called the electron transfer matrix element) between ET reactant and product states. The electronic coupling (V_{RP}), and nuclear reorganization energy (λ) are all components of the electron transfer rate defined by Marcus' theory, which also depends on the temperature (see Reference 1 below):

$$[{k_{ET}}= \frac{2\pi}{\hbar} V_{RP}^{2} \frac{1}{\sqrt{4\pi \lambda k_{B}T}} \exp \left(\frac{- \Delta G^{*}}{k_{B} T} \right)\]$$

The ET module utilizes the method of *Corresponding Orbital Transformation* to calculate V_{RP}. The only input required are the names of the files containing the open-shell (UHF) MO vectors for the ET reactant and product states (*R* and *P*).

The basis set used in the calculation of V_{RP} must be the same as the basis set used to calculate the MO vectors of *R* and *P*. The magnitude of V_{RP} depends on the amount of overlap between *R* and *P*, which is important to consider when choosing the basis set. Diffuse functions may be necessary to fill in the overlap, particularly when the ET distance is long.

The MO's of *R* and *P* must correspond to localized states. for instance, in the reaction $A^{-} + B \rightarrow A + B^{-}$ the transferring electron is localized on A in the reactant state and is localized on B in the product state. To verify the localization of the electron in the calculation of the vectors, carefully examine the Mulliken population analysis. In order to determine which orbitals are involved in the electron transfer, use the print keyword "mulliken ao" which prints the Mulliken population of each basis function.

An effective core potential (ECP) basis can be used to replace core electrons. However, there is one caveat: the orbitals involved in electron transfer must not be replaced with ECP's. Since the ET orbitals are valence orbitals, this is not usually a problem, but the user should use ECP's with care.

Suggested references are listed below. The first two references gives a good description of Marcus' two-state ET model, and the appendix of the third reference details the method used in the ET module.

1. R.A. Marcus, N. Sutin, Biochimica Biophysica Acta 35, 437, (1985).
2. J.R. Bolton, N. Mataga, and G. McLendon in “Electron Transfer in Inorganic, Organic and Biological Systems” (American Chemical Society, Washington, D.C., 1991)
3. A. Farazdel, M. Dupuis, E. Clementi, and A. Aviram, J. Am. Chem. Soc., 112, 4206 (1990).

16.1.1 VECTORS: input of MO vectors for ET reactant and product states

```
VECTORS [reactants] <string reactants_filename>
VECTORS [products ] <string products_filename>
```

In the VECTORS directive the user specifies the source of the molecular orbital vectors for the ET reactant and product states. This is required input, as no default filename will be set by the program. In fact, this is the only required input in the ET module, although there are other optional keywords described below.

16.1.2 FOCK/NOFOCK: method for calculating the two-electron contribution to V_{RP}

```
<string (FOCK||NOFOCK) default FOCK>
```

This directive enables/disables the use of the NWChem’s Fock matrix routine in the calculation of the two-electron portion of the ET Hamiltonian. Since the Fock matrix routine has been optimized for speed, accuracy and parallel performance, it is the most efficient choice.

Alternatively, the user can calculate the two-electron contribution to the ET Hamiltonian with another subroutine which may be more accurate for systems with a small number of basis functions, although it is slower.

16.1.3 TOL2E: integral screening threshold

```
TOL2E <real tol2e default max(10e-12,min(10e-7, S(RP)*10e-7 )>
```

The variable tol2e is used in determining the integral screening threshold for the evaluation of the two-electron contribution to the Hamiltonian between the electron transfer reactant and product states. As a default, tol2e is set depending on the magnitude of the overlap between the ET reactant and product states (S_{RP}), and is not less than 1.0d-12 or greater than 1.0d-7.

The input to specify the threshold explicitly within the ET directive is, for example:

```
tol2e 1e-9
```

16.1.4 Example

The following example is for a simple electron transfer reaction, $He \rightarrow He^{+}$. The ET calculation is easy to execute, but it is crucial that ET reactant and product wavefunctions reflect localized states. This can be accomplished using either a fragment guess, or a charged atomic density guess. For self-exchange ET reactions such as this one, you can use the REORDER keyword to move the electron from the first helium to the second.

Example input :

```
basis "ao basis"
 * library aug-cc-pvtz
end

geometry
 He 0 0 0
end

charge 1

scf
 tol2e 1d-9
 uhf
 doublet
 vectors output HeP.movecs
end
task scf

charge 0

scf
 uhf
 singlet
 vectors output He.movecs
end
task scf

geometry noautosym noautoz
  He 0.0   0.0   0.0
  He 5.0   0.0   0.0
end

charge 1
#ET reactants:
scf
  doublet; uhf; vectors input fragment HeP.movecs He.movecs output HeA.movecs
end
task scf

#ET products:
scf
  doublet; uhf; vectors input HeA.movecs reorder 2 1 output HeB.movecs
end
task scf

et
 vectors reactants HeA.movecs
 vectors products HeB.movecs
end
task scf et
```

Here is what the output looks like for this example:

```
                    Electron Transfer Calculation
                    -----------------------------

  MO vectors for reactants: HeA.movecs
  MO vectors for products : HeB.movecs

  Electronic energy of reactants     H(RR)      -5.2836825646
  Electronic energy of products      H(PP)      -5.2836825646

  Reactants/Products overlap S(RP) : -4.20D-04

  Reactants/Products interaction energy:
  --------------------------------------
  One-electron contribution         H1(RP)       0.0027017960

  Beginning calculation of 2e contribution
  Two-electron integral screening (tol2e) : 4.20D-11

  Two-electron contribution         H2(RP)      -0.0004625156
  Total interaction energy          H(RP)       0.0022392804

  Electron Transfer Coupling Energy |V(RP)|      0.0000220152
                                      4.832 cm-1
                                    0.000599 eV
                                      0.014 kcal/mol
```

The overlap between the ET reactant and product states (S_{RP}) is small, so the magnitude of the coupling between the states is also small. If the fragment guess or charged atomic density guess were not used, the Mulliken spin population would be 0.5 on both He atoms, the overlap between the ET reactant and product states would be 100% and an infinite

V_{RP} would result.

18 Controlling NWChem with Python

18.1 Overview

Python programs may be embedded into the NWChem input and used to control the execution of NWChem. Python is a very powerful and widely used scripting language that provides useful things such as variables, conditional branches and loops, and is also readily extended. Example applications include scanning potential energy surfaces, computing properties in a variety of basis sets, optimizing the energy w.r.t. parameters in the basis set, computing polarizabilities with finite field, and simple molecular dynamics.

Look in the NWChem contrib directory for useful scripts and examples. Visit the Python web-site https://www.python.org for a full manual and lots of useful code and resources.

18.2 How to input and run a Python program inside NWChem

A Python program is input into NWChem inside a Python compound directive.

```
python [print|noprint]
  ...
end
```

The END directive must be flush against the left margin (see the Troubleshooting section for the reason why).

The program is by default printed to standard output when read, but this may be disabled with the noprint keyword. Python uses indentation to indicate scope (and the initial level of indentation must be zero), whereas NWChem uses optional indentation only to make the input more readable. For example, in Python, the contents of a loop, or conditionally-executed block of code must be indented further than the surrounding code. Also, Python attaches special meaning to several symbols also used by NWChem. For these reasons, the input inside a PYTHON compound directive is read verbatim except that if the first line of the Python program is indented, the same amount of indentation is removed from all subsequent lines. This is so that a program may be indented inside the PYTHON input block for improved readability of the NWChem input, while satisfying the constraint that when given to Python the first line has zero indentation.

E.g., the following two sets of input specify the same Python program.

```
python
  print ("Hello")
  print ("Goodbye")
end

python
print ("Hello")
print ("Goodbye")
end
```

whereas this program is in error since the indentation of the second line is less than that of the first.

```
python
  print ("Hello")
print ("Goodbye")
end
```

The Python program is not executed until the following directive is encountered

```
task python
```

which is to maintain consistency with the behavior of NWChem in general. The program is executed by all nodes. This enables the full functionality and speed of NWChem to be accessible from Python, but there are some gotchas

- Print statements and other output will be executed by all nodes so you will get a lot more output than probably desired unless the output is restricted to just one node (by convention node zero).
- The calls to NWChem functions are all collective (i.e., all nodes must execute them). If these calls are not made collectively your program may deadlock (i.e., cease to make progress).
- When writing to the database (`rtdb_put()`) it is the data from node zero that is written.
- NWChem overrides certain default signal handlers so care must be taken when creating processes (see Section describing a scanning example).

18.3 NWChem extensions

Since we have little experience using Python, the NWChem-Python interface might change in a non-backwardly compatible fashion as we discover better ways of providing useful functionality. We would appreciate suggestions about useful things that can be added to the NWChem-Python interface. In principle, nearly any Fortran or C routine within NWChem can be extended to Python, but we are also interested in ideas that will enable users to build completely new things. For instance, how about being able to define your own energy functions that can be used with the existing optimizers or dynamics package?

Python has been extended with a module named “nwchem” which is automatically imported and contains the following NWChem-specific commands. They all handle NWChem-related errors by raising the exception “NWChemError”, which may be handled in the standard Python manner (see Section describing handling exception).

- `input_parse(string)` – invokes the standard NWChem input parser with the data in string as input. Note that the usual behavior of NWChem will apply – the parser only reads input up to either end of input or until a `TASK` directive is encountered (the task directive is not executed by the parser).
- `task_energy(theory)` – returns the energy as if computed with the NWChem directive `TASK ENERGY <THEORY>`.
- `task_gradient(theory)` – returns a tuple (energy,gradient) as if computed with the NWChem directive `TASK GRADIENT <THEORY>`.
- `task_optimize(theory)` – returns a tuple (energy,gradient) as if computed with the NWChem directive `TASK OPTIMIZE <THEORY>`. The energy and gradient will be those at the last point in the optimization and consistent with the current geometry in the database.
- `ga_nodeid()` – returns the number of the parallel process.
- `rtdb_print(print_values)` – prints the contents of the RTDB. If `print_values` is 0, only the keys are printed, if it is 1 then the values are also printed.
- `rtdb_put(name, values)` or `rtdb_put(name, values, type)` – puts the values into the database with the given name. In the first form, the type is inferred from the first value, and in the second form the type is specified using the last argument as one of INT, DBL, LOGICAL, or CHAR.
- `rtdb_get(name)` – returns the data from the database associated with the given name.

An example below explains, in lieu of a Python wrapper for the geometry object, how to obtain the Cartesian molecular coordinates directly from the database.

18.4 Examples

Several examples will provide the best explanation of how the extensions are used, and how Python might prove useful.

18.4.1 Hello world

```
python
  print ('Hello world from process %i'  % ga_nodeid())
end

task python
```

This input prints the traditional greeting from each parallel process.

18.4.2 Scanning a basis exponent

```
geometry units au
 O 0 0 0; H 0 1.430 -1.107; H 0 -1.430 -1.107
end

python
exponent = 0.1
while (exponent <= 2.01):
   input_parse('''
   basis noprint
    H library 3-21g; O library 3-21g; O d; %f 1.0
   end
   ''' % (exponent))
   print (' exponent = %5.4f,' % exponent, ', energy = %16.10f' % task_energy('scf'))
   exponent = exponent + 0.1
end

print none

task python
```

This program augments a 3-21g basis for water with a d-function on oxygen and varies the exponent from 0.1 to 2.0 in steps of 0.1, printing the exponent and energy at each step.

The geometry is input as usual, but the basis set input is embedded inside a call to `input_parse()` in the Python program. The standard Python string substitution is used to put the current value of the exponent into the basis set (replacing the %f) before being parsed by NWChem. The energy is returned by `task_energy('scf')` and printed out. The print none in the NWChem input switches off all NWChem output so all you will see is the output from your Python program.

Note that execution in parallel may produce unwanted output since all process execute the print statement inside the Python program.

Look in the NWChem contrib directory for a routine that makes the above task easier.

18.4.3 Scanning a basis exponent revisited

```
geometry units au
O 0 0 0; H 0 1.430 -1.107; H 0 -1.430 -1.107
end


python
  if (ga_nodeid() == 0): plotdata = open("plotdata",'w')

  def energy_at_exponent(exponent):
    input_parse('''
    basis noprint
      H library 3-21g; O library 3-21g; O d; %f 1.0
    end
    ''' % (exponent))
    return task_energy('scf')

  exponent = 0.1
  while exponent <= 2.01:
    energy = energy_at_exponent(exponent)
    if (ga_nodeid() == 0):
       print (' exponent = %5.4f,' % exponent, ', energy = %16.10f' % energy)
       plotdata.write('%f %f\n' % (exponent , energy))
    exponent = exponent + 0.1

  if (ga_nodeid() == 0): plotdata.close()
end

print none
task python
```

This input performs exactly the same calculation as the previous one, but uses a slightly more sophisticated Python program, also writes the data out to a file for easy visualization with a package such as gnuplot, and protects write statements to prevent duplicate output in a parallel job. The only significant differences are in the Python program. A file called “plotdata” is opened, and then a procedure is defined which given an exponent returns the energy. Next comes the main loop that scans the exponent through the desired range and prints the results to standard output and to the file. When the loop is finished the additional output file is closed.

18.4.4 Scanning a geometric variable

```
python
   geometry = '''
     geometry noprint; symmetry d2h
        C 0 0 %f; H 0  0.916 1.224
     end
   '''
   x = 0.6
   while (x < 0.721):
     input_parse(geometry % x)
     energy = task_energy('scf')
     print (' x = %5.2f   energy = %10.6f' % (x, energy))
     x = x + 0.01
end

basis; C library 6-31g*; H library 6-31g*; end

print none

task python
```

This scans the bond length in ethene from 1.2 to 1.44 in steps of 0.2 computing the energy at each geometry. Since it is using D_{2h} symmetry the program actually uses a variable (x) that is half the bond length.

Look in the NWChem contrib directory for a routine that makes the above task easier.

18.4.5 Scan using the BSSE counterpoise corrected energy

```
basis spherical
  Ne library cc-pvdz; BqNe library Ne cc-pvdz
  He library cc-pvdz; BqHe library He cc-pvdz
end

mp2; tight; freeze core atomic; end

print none

python
  supermolecule = 'geometry noprint;   Ne 0 0 0;   He 0 0 %f; end\n'
  fragment1     = 'geometry noprint;   Ne 0 0 0; BqHe 0 0 %f; end\n'
  fragment2     = 'geometry noprint; BqNe 0 0 0;   He 0 0 %f; end\n'

  def energy(geometry):
    input_parse(geometry + 'scf; vectors atomic; end\n')
    return task_energy('mp2')

  def bsse_energy(z):
    return energy(supermolecule % z) - \
        energy(fragment1 % z) - \
        energy(fragment2 % z)
  z = 3.3
  while (z < 4.301):
    e = bsse_energy(z)
    if (ga_nodeid() == 0): print (' z = %5.2f   energy = %10.7f ' % (z, e))
    z = z + 0.1
end

task python
```

This example scans the He–Ne bond-length from 3.3 to 4.3 and prints out the BSSE counterpoise corrected MP2 energy.

The basis set is specified as usual, noting that we will need functions on ghost centers to do the counterpoise correction. The Python program commences by defining strings containing the geometry of the super-molecule and two fragments, each having one variable to be substituted. Next, a function is defined to compute the energy given a geometry, and then a function is defined to compute the counterpoise corrected energy at a given bond length. Finally, the bond length is scanned and the energy printed. When computing the energy, the atomic guess has to be forced in the SCF since by default it will attempt to use orbitals from the previous calculation which is not appropriate here.

Since the counterpoise corrected energy is a linear combination of other standard energies, it is possible to compute the analytic derivatives term by term. Thus, combining this example and the next could yield the foundation of a BSSE corrected geometry optimization package.

18.4.6 Scan the geometry and compute the energy and gradient

```
basis noprint; H library sto-3g; O library sto-3g; end

python
 print ('   y     z     energy                gradient')
 print (' ----- ----- ---------- ----------------------------------')
 y = 1.2
 while y <= 1.61:
    z = 1.0
    while z <= 1.21:
       input_parse('''
         geometry noprint units atomic
           O 0   0   0
           H 0  %f -%f
           H 0 -%f -%f
         end
       ''' % (y, z, y, z))

       (energy,gradient) = task_gradient('scf')

       print (' %5.2f %5.2f %9.6f' % (y, z, energy)),
       i = 0
       while (i < len(gradient)):
          print ('%5.2f' % gradient[i]),
          i = i + 1
       print ('')
       z = z + 0.1
    y = y + 0.1
end

print none

task python
```

This program illustrates evaluating the energy and gradient by calling task_gradient(). A water molecule is scanned through several C_{2v} geometries by varying the y and z coordinates of the two hydrogen atoms. At each geometry the coordinates, energy and gradient are printed.

The basis set (sto-3g) is input as usual. The two while loops vary the y and z coordinates. These are then substituted into a geometry which is parsed by NWChem using `input_parse()`. The energy and gradient are then evaluated by calling `task_gradient()` which returns a tuple containing the energy (a scalar) and the gradient (a vector or list). These are printed out exploiting the Python convention that a print statement ending in a comma does not print end-of-line.

18.4.7 Reaction energies varying the basis set

```
mp2; freeze atomic; end

print none

python
 energies = {}
 c2h4 = 'geometry noprint; symmetry d2h; \
       C 0 0 0.672; H 0 0.935 1.238; end\n'
 ch4  = 'geometry noprint; symmetry td; \
       C 0 0 0; H 0.634 0.634 0.634; end\n'
 h2   = 'geometry noprint; H 0 0 0.378; H 0 0 -0.378; end\n'

 def energy(basis, geometry):
   input_parse('''
     basis spherical noprint
       c library %s ; h library %s
     end
   ''' % (basis, basis))
   input_parse(geometry)
   return task_energy('mp2')

 for basis in ('sto-3g', '6-31g', '6-31g*', 'cc-pvdz', 'cc-pvtz'):
   energies[basis] =   2*energy(basis, ch4) \
                     - 2*energy(basis, h2) - energy(basis, c2h4)
   if (ga_nodeid() == 0): print (basis, ' %8.6f' % energies[basis])
end

task python
```

In this example the reaction energy for $2H_2 + C_2H_4 \rightarrow 2CH_4$ is evaluated using MP2 in several basis sets. The geometries are fixed, but could be re-optimized in each basis. To illustrate the useful associative arrays in Python, the reaction energies are put into the associative array energies – note its declaration at the top of the program.

18.4.8 Using the database

```
python
  rtdb_put("test_int2", 22)
  rtdb_put("test_int", [22, 10, 3],    INT)
  rtdb_put("test_dbl", [22.9, 12.4, 23.908],  DBL)
  rtdb_put("test_str", "hello", CHAR)
  rtdb_put("test_logic", [0,1,0,1,0,1], LOGICAL)
  rtdb_put("test_logic2", 0, LOGICAL)

  rtdb_print(1)

  print "test_str    = ", rtdb_get("test_str")
  print "test_int    = ", rtdb_get("test_int")
  print "test_in2    = ", rtdb_get("test_int2")
  print "test_dbl    = ", rtdb_get("test_dbl")
  print "test_logic  = ", rtdb_get("test_logic")
  print "test_logic2 = ", rtdb_get("test_logic2")
end

task python
```

This example illustrates how to access the database from Python.

18.4.9 Handling exceptions from NWChem

```
geometry; he 0 0 0; he 0 0 2; end
basis; he library 3-21g; end
scf; maxiter 1; end

python
  try:
    task_energy('scf')
  except NWChemError, message:
    print 'Error from NWChem ... ', message
end

task python
```

The above test program shows how to handle exceptions generated by NWChem by forcing an SCF calculation on $He_{2<\sub>}$ to fail due to insufficient iterations.

If an NWChem command fails it will raise the exception “NWChemError” (case sensitive) unless the error was fatal. If the exception is not caught, then it will cause the entire Python program to terminate with an error. This Python program catches the exception, prints out the message, and then continues as if all was well since the exception has been handled.

If your Python program detects an error, raise an unhandled exception. Do not call exit(1) since this may circumvent necessary clean-up of the NWChem execution environment.

18.4.10 Accessing geometry information: a temporary hack

In an ideal world the geometry and basis set objects would have full Python wrappers, but until then a back-door solution will have to suffice. We’ve already seen how to use `input_parse()` to put geometry (and basis) data into NWChem, so it only remains to get the geometry data back after it has been updated by a geometry optimzation or some other operation.

The following Python procedure retrieves the coordinates in the same units as initially input for a geometry of a given name. Its full source is included in the NWChem contrib directory.

```
def geom_get_coords(name):
  try:
    actualname = rtdb_get(name)
  except NWChemError:
    actualname = name
  coords = rtdb_get('geometry:' + actualname + ':coords')
  units  = rtdb_get('geometry:' + actualname + ':user units')
  if (units == 'a.u.'):
    factor = 1.0
  elif (units == 'angstroms'):
    factor = rtdb_get('geometry:'+actualname+':angstrom_to_au')
  else:
    raise NWChemError,'unknown units'
  i = 0
  while (i < len(coords)):
    coords[i] = coords[i] / factor
    i = i + 1
  return coords
```

A geometry with name NAME has its coordinates (in atomic units) stored in the database entry`geometry:NAME:coords` . A minor wrinkle here is that indirection is possible (and used by the optimizers) so that we must first check if NAME actually points to another name. In the program this is done in the first try…except sequence. With the actual name of the geometry, we can get the coordinates. Any exceptions are passed up to the caller. The rest of the code is just to convert back into the initial input units – only atomic units or Angstrøms are handled in this simple example. Returned is a list of the atomic coordinates in the same units as your initial input.

The routine is used as follows

```
coords = geom_get_coords('geometry')
```

or, if you want better error handling

```
try:
  coords = geom_get_coords('geometry')
except NWChemError,message:
  print 'Coordinates for geometry not found ', message
else:
  print coords
```

This is very dirty and definitely not supported from one release to another, but, browsing the output of`rtdb_print()` at the end of a calculation is a good way to find stuff. To be on safer ground, look in the programmers manual since some of the high-level routines do pass data via the database in a well-defined and supported manner. Be warned – you must be very careful if you try to modify data in the database. The input parser does many important things that are not immediately apparent (e.g., ensure the geometry is consistent with the point group, mark the SCF as not converged if the SCF options are changed, …). Where at all possible your Python program should generate standard NWChem input and pass it to `input_parse()` rather than setting parameters directly in the database.

18.4.11 Scaning a basis exponent yet again: plotting and handling child processes

```
geometry units au
  O 0 0 0; H 0 1.430 -1.107; H 0 -1.430 -1.107
end

print none

python
  import Gnuplot, time, signal

  def energy_at_exponent(exponent):
    input_parse('''
      basis noprint
        H library 3-21g; O library 3-21g; O d; %f 1.0
      end
    ''' % (exponent))
    return task_energy('scf')

  data = []
  exponent = 0.5
  while exponent <= 0.6:
    energy = energy_at_exponent(exponent)
    print ' exponent = ', exponent, ' energy = ', energy
    data = data + [exponent,energy](exponent,energy.md)
    exponent = exponent + 0.02

  if (ga_nodeid() == 0):
    signal.signal(signal.SIGCHLD, signal.SIG_DFL)
    g = Gnuplot.Gnuplot()
    g('set data style linespoints')
    g.plot(data)
    time.sleep(30)  # 30s to look at the plot

end

task python
```

This illustrates how to handle signals from terminating child processes and how to generate simple plots on UNIX systems. The scanning example is modified so that instead of writing the data to a file for subsequent visualization, it is saved for subsequent visualization with Gnuplot (you'll need both Gnuplot and the corresponding package for Python in your PYTHONPATH. Look at https://web.archive.org/web/20010717060625/http://monsoon.harvard.edu/~mhagger/download/).

The issue is that NWChem traps various signals from the O/S that usually indicate bad news in order to provide better error handling and reliable clean-up of shared, parallel resources. One of these signals is SIGCHLD which is generated whenever a child process terminates. If you want to create child processes within Python, then the NWChem handler for SIGCHLD must be replaced with the default handler. There seems to be no easy way to restore the NWChem handler after the child has completed, but this should have no serious side effect.

18.5 Troubleshooting

Common problems with Python programs inside NWChem.

1. You get the message

   ```
   0:python_input: indentation must be >= that of first line: 4
   ```

   This indicates that NWChem thinks that a line is less indented than the first line. If this is not the case then perhaps there is a tab in your input which NWChem treats as a single space character but appears to you as more spaces. Try running untabify in Emacs. The problem could also be the END directive that terminates the PYTHON compound directive, since Python also has an end statement. To avoid confusion the END directive for NWChem must be at the start of the line.
2. Your program hangs or deadlocks – most likely you have a piece of code that is restricted to executing on a subset of the processors (perhaps just node 0) but is calling (perhaps indirectly) a function that must execute on all nodes.

16.3 Quantum

The quantum features are routines for printing out one- and two-electron integrals that can be utilized in quantum algorithms. The outputs can be interpreted and formatted in YAML files that include the integrals and basic information

including initial wavefunction guesses for quantum algorithms. There are currently two options for integrals:

16.3.1 Bare Hamiltonian

To print the bare Hamiltonian, you must set up a CCSD calculation with the following parameters:
* The symmetry must be declared as ‘C1’
* In the TCE block, you must include the following:

```
2eorb
2emet 13
nroots ### (Optional, but will include leading excitations for excited-state quantum calculations)
```

You must set the printing parameters at the end of the input file:

```
set tce:print_integrals T
set tce:qorb ### (Total number of orbitals you wish to be printed)
set tce:qela ### (Number of alpha electrons)
set tce:qelb ### (number of beta electrons)
```

Notes:

- The two-electron integrals are printed out in compact Mulliken notation (each printed element corresponds to (ij|kl), (ij|lk), (ji|kl), (ji|lk), (kl|ij), (kl|ji), (lk|ij), and (lk|ji))
- The number of alpha and beta electrons that are set in the printing does not need to correspond to the total number of alpha and beta electrons. Rather it is the number of electrons included in the printing, starting from the HOMO and counting down. This allows you to ‘freeze’ core orbitals. However, this is not recommended, because unlike freezing Fock matrix elements and two-electron integrals in correlated calculations, this procedure is dropping core orbitals of the standard one- and two-electron integrals and can lead to an imbalance of the correlation energy.
- If the number of electrons and orbitals do not correspond to the total numbers of electrons and orbitals, then the calculation will perform a corresponding frozen core/frozen virtual CC/EOMCC calculation. This is to ensure that leading CC/EOMCC excitations do not correspond to orbitals outside of the printed integrals.

16.3.2 DUCC Hamiltonian

The double unitary coupled-cluster (DUCC) Hamiltonian is a way of incorporating correlation effects (mainly dynamical) into a reduced dimensionality Hamiltonian based on a defined active space. The procedure currently only reduces the virtual space and all occupied orbitals are considered active. Only the ground-state implementation is available in the current NWChem release.

To print the DUCC Hamiltonian, you must set up a CCSD calculation with the following parameters:

- The symmetry must be declared as ‘C1’
- In the TCE block, you must include the following:

```
2eorb
2emet 13
```

You must set the printing parameters at the end of the input file:

```
set tce:qducc T
set tce:nactv ### (The number of active virtual orbitals.
        Remember that all occupied orbitals are active as well)
set tce:nonhf F/T (If a non-RHF reference is used, set to T. Otherwise, keep as F.)
set tce:ducc_model ### (Determines how the similarity transformed Hamiltonian is truncated. See note below.)
```

Notes:

- Six models for the truncation of the similarity transformed are implemented and numbered 1-6. They correspond to

approximations A(2)-A(7), respectively, that are detailed in arXiv:2110.12077. Model 3 corresponds to the earliest implementation described in J. Chem. Phys. 151, 014107 (2019) and J. Chem. Phys. 151, 234114 (2019). In contrast to these early publications, the full form of the two-electron integral is printed. Model 2 is an aggressive truncation and is not recommended for use.

- The entire two-electron integral is printed out since the DUCC Hamiltonian does not have the same symmetry as the bare Hamiltonian. Instead of the 8-fold symmetry, it has dual 4-fold symmetries

```
(IJ|KL) = (JI|LK) = (KL|IJ) = (LK|JI)
Separately, (IJ|LK) = (JI|KL) = (KL|JI) = (LK|IJ)
But (IJ|KL) /= (IJ|LK)
```

- When utilizing the YAML interpreter, the full two-electron integral will be printed in the YAML file, despite it being labeled as ‘sparse’ in the YAML file.
- Frozen core calculations should not be performed. There is nothing preventing such calculation from being executed, but the integrals will be incorrect.
- Since the all-orbital coupled-cluster calculation must be performed, then there is a chance that a leading excitation outside of the active-space will be printed and therefore included in the initial wave function description in the YAML file. The user ought to be aware of this risk and make changes to the YAML file to exclude these excitations, which would include renormalizing the remaining amplitudes of the initial guess.

16.3.3 YAML Interpreter

Once the calculation is completed, the integrals and basic information including initial wavefunction guesses for quantum algorithms can be extracted into a YAML file. To generate the YAML file, executed the following command:

```
python $NWCHEM_TOP/contrib/quasar/export_chem_library_yaml.py < {NWChem output file} > {YAML file name}
```

Note: * the ‘fci_energy’ in the YAML file is taken to be the energy of the correlated method, which, in either case, is the CCSD energy.

16.3.4 Example input file for Bare Hamiltonian

```
echo

start Lih

geometry units angstroms
symmetry C1
 Li 0 0   0.000
 H  0 0   1.600
end

basis spherical
 * library sto-3g
end

scf
  singlet
  rhf
  thresh 1e-10
end

tce
  2eorb
  2emet 13
  ccsd
  thresh 1.0d-8
  maxiter 150
  nroots 2
end

set tce:print_integrals T
set tce:qorb 6
set tce:qela 2
set tce:qelb 2

task tce energy
```

16.3.5 Example input file for DUCC

```
echo

start Lih

geometry units angstroms
symmetry C1
 Li 0 0   0.000
 H  0 0   1.600
end

basis spherical
 * library sto-3g
end

scf
  singlet
  rhf
  thresh 1e-10
end

tce
  2eorb
  2emet 13
  ccsd
  thresh 1.0d-8
  maxiter 150
end

set tce:qducc T
set tce:nactv 4
set tce:nonhf F
set tce:ducc_model 3

task tce energy
```

16.4 Vibrational SCF (VSCF)

16.4.1 Overview

The VSCF module[1] can be used to calculate the anharmonic contributions to the vibrational modes of the molecule of interest. Energies are calculated on a one-dimensional grid along each normal mode, on a two-dimensional grid along each pair of normal modes, and optionally on a three-dimensional grid along each triplet of normal modes. These energies are then used to calculate the vibrational nuclear wavefunction at an SCF- (VSCF) and MP2-like (cc-VSCF) level of theory.

VSCF can be used at all levels of theory, SCF and correlated methods, and DFT. For correlated methods, only the SCF level dipole is evaluated and used to calculate the IR intensity values.

The VSCF module is started when the task directive TASK <theory> vscf is defined in the user input file. The input format has the form:

```
VSCF
  [coupling <string couplelevel default "pair">]
  [ngrid    <integer default 16 >]
  [iexcite  <integer default 1  >]
  [vcfct    <real    default 1.0>]
END
```

16.4.2 Coupling keyword

The order of coupling of the harmonic normal modes included in the calculation is controlled by the specifying:

```
coupling <string couplelevel default "pair">
```

For coupling=diagonal a one-dimensional grid along each normal mode is computed.
For coupling=pair a two-dimensional grid along each pair of normal modes is computed.
For coupling=triplet a three-dimensional grid along each triplet of normal modes is computed.

16.4.3 Ngrid keyword

The number of grid points along each normal mode, or pair of modes can be defined by specifying:

```
ngrid <integer default 16>
```

16.4.4 Iexcite keyword

This VSCF module by default calculates the ground state (ν=0), but can also calculate excited states (such as ν=1). The number of excited states calculated is defined by specifying:

```
iexcite <integer default 1>
```

With iexcite=1 the fundamental frequencies are calculated.
With iexcite=2 the first overtones are calculated.
With iexcite=3 the second overtones are calculated.

16.4.5 Vcfct keyword

In certain cases the pair coupling potentials can become larger than those for a single normal mode. In this case the pair potentials need to be scaled down. The scaling factor used can be defined by specifying:

```
vcfct <real default 1.0>
```

16.4.6 References

1. Chaban, G. M.; Jung, J. O.; Gerber, R. B. Ab Initio Calculation of Anharmonic Vibrational States of Polyatomic Systems: Electronic Structure Combined with Vibrational Self-Consistent Field. *The Journal of Chemical Physics* **1999**, *111* (5), 1823–1829. https://doi.org/10.1063/1.479452

16.5 Correlation consistent Composite Approach (ccCA)

The CCCA module calculates the total energy using the correlation consistent Composite Approach (ccCA). At present the ccCA module is designed for the study of main group species only.

$$E_{ccCA} = \Delta E_{MP2/CBS} \ \ + \Delta E_{CC} + \Delta E_{CV} + \Delta E_{SR} + \Delta E_{ZPE}$$

where EMP2/CBS is the complete basis set extrapolation of MP2 energies with the aug-cc-pVnZ (n=T,D,Q) series of basis sets, ΔE_{CC} is the correlation correction,

$$\Delta E_{CC} = E_{CCSD(T)/cc-pVTZ} \ \ \ - E_{MP2/cc-pVTZ}$$

ΔE_{CV} is the core-valence correction,

$$\Delta E_{CV} = E_{MP2(FC1)/aug-cc-pCVTZ} \ \ \ \ \ - E_{MP2/aug-cc-pVTZ}$$

where $FC1$ symbolizes that all electrons of first-row atoms are correlated, all electrons of second-row atoms are correlated except the 1s MOs, and all electrons of atoms K–Kr are correlated except the 1s2s2p MOs.
ΔE_{SR} is the scalar-relativistic correction,

$$\Delta E_{SR} = E_{MP2/cc-pVTZ-DK} \ \ \ - E_{MP2/cc-pVTZ}$$

and ΔE_{ZPE} is the zero-point energy correction or thermal correction. Geometry optimization and subsequent frequency analysis are performed with B3LYP/cc-pVTZ.

Suggested reference: N.J. DeYonker, B. R. Wilson, A.W. Pierpont, T.R. Cundari, A.K. Wilson, Mol. Phys. 107, 1107 (2009). Earlier variants for ccCA algorithms can also be found in: N. J. DeYonker, T. R. Cundari, A. K. Wilson, J. Chem. Phys. 124, 114104 (2006).

The ccCA module can be used to calculate the total single point energy for a fixed geometry and the zero-point energy correction is not available in this calculation. Alternatively the geometry optimization by B3LYP/cc-pVTZ is performed before the single point energy evaluation. For open shell molecules, the number of unpaired electrons must be given explicitly.

```
CCCA

    [(ENERGY||OPTIMIZE)   default ENERGY]
    [(DFT||DIRECT)   default DFT]
    [(MP2||MBPT2)   default MP2]
    [(RHF||ROHF||UHF)   default RHF]
    [(CCSD(T)||TCE)   default CCSD(T)]
    [NOPEN   <integer number of unpaired electrons   default   0 >]
    [(THERM||NOTHERM)   default  THERM]
    [(PRINT||NOPRINT) default NOPRINT]
    [BASIS <basis name for orbital projection guess>]
    [MOVEC <file name for orbital projection guess>]
END
```

One example of input file for single point energy evaluation is given here:

```
start h2o_ccca

title "H2O, ccCA test"

geometry units au

 H      0.0000000000   1.4140780900  -1.1031626600
 H      0.0000000000  -1.4140780900  -1.1031626600
 O      0.0000000000   0.0000000000  -0.0080100000
end

task ccca
```

An input file for the ground state of O2 with geometry optimization is given below:

```
start o2_ccca

title "O2, ccCA test"

geometry units au

 O      0.0000000000   0.0000000000  -2.0000
 O      0.0000000000   0.0000000000   0.0000

end

ccca
 optimize
 dft
 nopen 2
end

task ccca
```

16.6 Interfaces to Other Programs

NWChem has interfaces to several different packages which are listed below. In general, the NWChem authors work with the authors of the other packages to make sure that the interface works. However, any problems with the interface should be reported through the github issue page https://github.com/nwchemgit/nwchem/issues

16.6.1 DIRDYVTST – DIRect Dynamics for Variational Transition State Theory

by Bruce C. Garrett, Environmental Molecular Sciences Laboratory, Pacific Northwest Laboratory, Richland, Washington

Yao-Yuan Chuang and Donald G. Truhlar, Department of Chemistry and Super Computer Institute, University of Minnesota, MN 55455-0431

and interfaced to NWChem by

Ricky A. Kendall, Scalable Computing Laboratory, Ames Laboratory and Iowa State University, Ames, IA 50011

Theresa L. Windus, Environmental Molecular Sciences Laboratory, Pacific Northwest Laboratory, Richland, Washington

If you use the DIRDYVTST portion of NWChem, please use following citation in addition to the usual NWChem citation:

> DIRDYVTST, Yao-Yuan Chuang and Donald G. Truhlar, Department of Chemistry and Super Computer Institute, University of Minnesota; Ricky A. Kendall,Scalable Computing Laboratory, Ames Laboratory and Iowa State University; Bruce C. Garrett and Theresa L. Windus, Environmental Molecular Sciences Laboratory, Pacific Northwest Laboratory. `

16.6.1.1 Introduction

By using DIRDYVTST, a user can carry out electronic structure calculations with NWChem and use the resulting energies, gradients, and Hessians for direct dynamics calculations with POLYRATE. This program prepares the file30 input for POLYRATE from NWChem electronic structure calculations of energies, gradients and Hessians at the reactant, product, and saddle point geometries and along the minimum energy path. Cartesian geometries for the reactants, products, and saddle points need to be input to this program; optimization of geometries is not performed in this program. Note that DIRDYVTST is based on the DIRDYGAUSS program and is similar to two other programs: DDUTILITIES and GAUSSRATE. Users of this module are encouraged to read the POLYRATE manual since they will need to create the file fu5 input to run calculations with POLYRATE.

Notes about the code:

Input. The code has been written to parallel, as much as possible, the POLYRATE code.

Output. There is one default output file for each DIRDYVTST run - .file30.

Integrators for following the reaction path. Currently the Euler and three Page-McIver (PM) methods are implemented. The PM methods are the local quadratic approximation (LQA), the corrected LQA (CLQA), and the cubic (CUBE) algorithm. The PM methods are implemented so that the Hessian can be reused at intermediate steps at which only the gradient is updated.

16.6.1.2 Files

Test runs are located in directories in $NWCHEM_TOP/QA/tests. Test runs are available for two systems: H + H_2 and OH + H_2.

The H + H_2 test uses the Euler integration method at the SCF/3-21G level of theory to calculate points along the reaction path. This test is located in the $NWCHEM_TOP/QA/tests/h3tr1 directory.

The OH + H_2 test uses the Page-McIver CUBE algorithm to calculate points on the SCF/3-21G surface and does additional single point calculations at the SCF/6-31G* level of theory. This test is located in the $NWCHEM_TOP/QA/tests/oh3tr3 directory.

Note: These tests are set up with SCF, however, other levels of theory can be used. The initial hessian calculations at the reactants, products and saddle point can cause some problems when numerical hessians are required (especially when there is symmetry breaking in the wavefunction).

16.6.1.3 Detailed description of the input

The input consists of keywords for NWChem and keywords related to POLYRATE input. The first set of inputs are for NWChem with the general input block of the form:

```
DIRDYVTST [autosym [real tol default 1d-2] | noautosym]
  [THEORY <string theory> [basis <string basis default "ao basis">] \
                [ecp <string ecp>] [input <string input>]]
  [SPTHEORY <string theory> [basis <string basis default "ao basis">]
                [ecp <string ecp>] [input <string input>`]]
  ...
END
```

16.6.1.3.1 Use of symmetry

The use of symmetry in the calculation is controlled by the keyword `autosym | noautosym` which is used as described in the geometry directive. `Autosym` is on by default. A couple words of warning here. The tolerance related to `autosym` can cause problems when taking the initial step off of the transition state. If the tolerance is too large and the initial step relatively small, the resulting geometry will be close to a higher symmetry than is really wanted and the molecule will be symmetrized into the higher symmetry. To check this, the code prints out the symmetry at each geometry along the path. It is up to the user to check the symmetry and make sure that it is the required one. In perverse cases, the user may need to turn autosym off (noautosym) if changing the tolerance doesn't produce the desired results. In the case that autosym is used, the user does not need to worry about the different alignment of the molecule between NWChem and POLYRATE, this is taken care of internally in the DIRDYVTST module.

16.6.1.3.2 Basis specification

The basis name on the theory or sptheory directive is that specified on a basis set directive and not the name of a standard basis in the library. If not specified, the basis set for the sptheory defaults to the theory basis which defaults to “ao basis”.

16.6.1.3.3 Effective core potentials

If an effective core potential is specified in the usual fashion outside of the DIRDYVTST input then this will be used in all calculations. If an alternative ECP name (the name specified on the ECP directive in the same manner as done for basis

sets) is specified on one of the theory directives, then this ECP will be used in preference for that level of theory.

16.6.1.3.4 General input strings

For many purposes, the ability to specify the theory, basis and effective core potential is adequate. All of the options for each theory are determined from their independent input blocks. However, if the same theory (e.g., DFT) is to be used with different options for theory and sptheory, then the general input strings must be used. These strings are processed as NWChem input each time the theoretical calculation is invoked. The strings may contain any NWChem input, except for options pertaining to DIRDYVTST and the task directive. The intent is that the strings be used just to control the options pertaining to the theory being used.

A word of caution. Be sure to check that the options are producing the desired results. Since the NWChem database is persistent, the input strings should fully define the calculation you wish to have happen.

For instance, if the theory model is DFT/LDA/3-21g and the sptheory model is DFT/B3LYP/6-311g**, the DIRDYVTST input might look like this

```
dirdyvtst
  theory  dft basis 3-21g    input "dft\; xc\; end"
  sptheory dft basis 6-311g** input "dft\; xc b3lyp\; end"
  ....
end
```

The empty XC directive restores the default LDA exchange-correlation functional. Note that semi-colons and other quotation marks inside the input string must be preceded by a backslash to avoid special interpretation.

16.6.1.3.5 POLYRATE related options

These keyword options are simlar to the POLYRATE input format, except there are no ENERGETICS, OPTIMIZATION, SECOND, TUNNELING, and RATE sections.

16.6.1.3.5.1 GENERAL SECTION

The GENERAL section has the following format:

```
 * GENERAL

[TITLE <string title>]
ATOMS
  <integer num> <string tag> [<real mass>]
  ...
END
[SINGLEPOINT]
[SAVEFILE (vecs || hess || spc)
```

Descriptions

TITLE is a keyword that allows the user to input a description of the calculation. In this version, the user can only have a single-line description.

For example:

```
TITLE Calculation of D + HCl reaction
```

ATOMS is a list keyword that is used to input a list of the atoms. It is similar to POLYRATE in that the order of the atom and the atomic symbol are required in a single line. If isotope of the element is considered then the atomic mass is required in units of amu.

For example:

```
ATOMS
  1    H    2.014
  2    H
  3    Cl
END`
```

SINGLEPOINT is a keyword that specifies that a single point calculation is to be performed at the reactants, products and saddle point geometries. The type of single point calculation is specified in the sptheory line.

SAVEFILE is a keyword that specifies that NWChem files are to be saved. Allowed values of variable input to SAVEFILE are vecs, hess, and spc for saving the files base theory movecs, base theory hessian and singlepoint calculation movecs.

16.6.1.3.5.2 REACT1, REACT2, PROD1, PROD2, AND START SECTIONS

These sections have the following format:

```
*(REACT1 || REACT2 || PROD1 || PROD2 || START)
 GEOM
  <integer num> <real x y z>
  ...
 END
 SPECIES (ATOMIC || LINRP || NONLINRP || LINTS || NONLINTS default NONLINRP)
```

REACT1 and REACT2 are input for each of the reactants and PROD1 and PROD2 are input for each of the products. REACT1 and PROD1 are required. START is the input for the transition state if one exists, or starting point to follow downhill the MEP.

Descriptions

GEOM is a list keyword that indicates the geometry of the molecule in Cartesian coordinates with atomic unit.

For example:

```
GEOM
  1    0.0    0.0    0.0
  2    0.0    0.0    1.5
END
```

SPECIES is a variable keyword that indicates the type of the molecule. Options are: ATOMIC (atomic reactant or product), LINRP (linear reactant or product), NONLINRP (nonlinear reactant or product), LINTS (linear transition state), and NONLINTS (nonlinear transition state).

For example:

```
SPECIES atomic
```

16.6.1.3.5.3 PATH SECTION

The Path section has the format:

```
*PATH
 [SCALEMASS <real scalemass default 1.0>]
 [SSTEP <real sstep default 0.01>]
 [SSAVE <real ssave default 0.1>]
 [SHESS <real shess default SSAVE>]
 [SLP <real slp default 1.0>]
 [SLM <real slm default -1.0>]
 [SIGN (REACTANT || PRODUCT default REACTANT)]
 [INTEGRA (EULER || LQA || CLQA || CUBE default EULER)]
 [PRINTFREQ (on || off default off)]
```

Descriptions

SCALEMASS is a variable keyword that indicates the arbitrary mass (in amu) used for mass-scaled Cartesian coordinates. This is the variable called mu in published papers. Normally, this is taken as either 1.0 amu or, for bimolecular reactions, as the reduced mass of relative translation of the reactants.

SSTEP is a variable keyword that indicates the numerical step size (in bohrs) for the gradient grid. This is the step size for following the minimum energy path.

SSAVE is a variable keyword that indicates the numerical step size (in bohrs) for saving the Hessian grid. At each save point the potential and its first and second derivatives are recalculated and written to the .file30 file. For example, if

SSTEP=0.01 and SSAVE=0.1, then the potential information is written to .file30 every 10 steps along the gradient grid.

SHESS is a variable keyword that indicates the numerical step size (in bohrs) for recomputing the Hessian when using a Page-McIver integrator (e.g., LQA, CLQA, or CUBE). For Euler integration SHESS = SSAVE. For intermediate points along the gradient grid, the Hessian matrix from the last Hessian calculation is reused. For example, if SSTEP=0.01 and SHESS=0.05, then the Hessian matrix is recomputed every 5 steps along the gradient grid.

SLP is a variable keyword that indicates the positive limit of the reaction coordinate (in bohrs).

SLM is a variable keyword that indicates the negative limit of the reaction coordinate (in bohrs).

SIGN is a variable keyword used to ensure the conventional definition of the sign of s, s \< 0 for the reactant side and s > 0 for the product side, is followed. PRODUCT should be used if the eigenvector at the saddle point points toward the product side and REACTANT if the eigenvector points toward the reactant side.

INTEGRA is a variable keyword that indicates the integration method used to follow the reaction path. Options are: EULER, LQA, CLQA, and CUBE.

PRINTFREQ is a variable keyword that indicates that projected frequencies and eigenvectors will be printed along the MEP.

16.6.1.3.6 Restart

DIRDYVTST calculations should be restarted through the normal NWChem mechanism. The user needs to change the start directive to a restart directive and get rid of any information that will overwrite important information in the RTDB. The file.db and file.file30 need to be available for the calculation to restart properly.

16.6.1.3.7 Example

This is an example that creates the file30 file for POLYRATE for H + H_2. Note that the multiplicity is that of the entire supermolecule, a doublet. In this example, the initial energies, gradients, and Hessians are calculated at the UHF/3-21G level of theory and the singlepoint calculations are calculated at the MP2/cc-pVDZ level of theory with a tighter convergence threshold than the first SCF.

```
start h3test

basis
 h library 3-21G
end

basis singlepoint
 h library cc-pVDZ
end

scf
  uhf
  doublet
  thresh 1.0e-6
end

dirdyvtst autosym 0.001
  theory scf input "scf\; uhf\; doublet\; thresh 1.0e-06\; end"
  sptheory mp2 basis singlepoint input \
    "scf\; uhf\; doublet\; thresh 1.0e-07\; end"
*GENERAL
  TITLE
    Test run: H+H2 reaction, Page-McIver CLQA algorithm, no restart

  ATOMS
    1    H
    2    H
    3    H
  END

  SINGLEPOINT

*REACT1
   GEOM
     1  0.0   0.0   0.0
     2  0.0   0.0   1.3886144
    END

   SPECIES LINRP

*REACT2
  GEOM
    3    0.0   0.0 190.3612132
  END

  SPECIES  ATOMIC

*PROD2
  GEOM
    1   0.0   0.0 190.3612132
  END

  SPECIES   ATOMIC

*PROD1

  GEOM
    2  0.0   0.0   1.3886144
    3   0.0   0.0   0.0
  END

  SPECIES  LINRP

*START

  GEOM
    1  0.0   0.0  -1.76531973
    2  0.0   0.0   0.0
    3  0.0   0.0   1.76531973
  END

  SPECIES  LINTS

*PATH
  SSTEP  0.05
  SSAVE  0.05
  SLP    0.50
  SLM   -0.50
  SCALEMASS 0.6718993
  INTEGRA   CLQA
end

task dirdyvtst
```

17 Molecular Mechanics

17.1 Prepare

The prepare module is used to set up the necessary files for a molecular dynamics simulation with NWChem. User supplied coordinates can be used to generate topology and restart files. The topology file contains all static information about a molecular system, such as lists of atoms, bonded interactions and force field parameters. The restart file contains all dynamic information about a molecular system, such as coordinates, velocities and properties.

Without any input, the prepare module checks the existence of a topology and restart file for the molecular systems. If these files exist, the module returns to the main task level without action. The module will generate these files when they do not exist. Without any input to the module, the generated system will be for a non-solvated isolated solute system.

To update existing files, including solvation, the module requires input directives read from an input deck,

```
prepare
  ...
end
```

The prepare module performs three sub-tasks:

- sequence generation : This sub-task analyzes the supplied coordinates from a PDB-formatted file or from the input geometry, and generates a sequence file, containing the description of the system in terms of basic building blocks found as fragment or segment files in the database directories for the force field used. If these files do not exist, they are generated based on the supplied coordinates. This process constists of generating a fragment file with the list of atoms with their force field dependent atom types, partial atomic charges calculated from a Hartree Fock calculation for the fragment, followed by a restrained electrostatic potential fit, and a connectivity list. From the information on this fragment file the lists of all bonded interactions are generated, and the complete lists are written to a segment file.
- topology generation : Based on the generated or user-supplied e file and the force field specific segment database files, this sub-task compiles the lists of atoms, bonded interactions, excluded pairs, and substitutes the force field parameters. Special commands may be given to specify interaction parameters that will be changing in a free energy evaluation.
- restart generation : Using the user supplied coordinates and the topology file for the chemical system, this sub-task generates a restart file for the system with coordinates, velocities and other dynamic information. This step may include solvation of the chemical system and specifying periodic boundary conditions.

Files involved in the preparation phase exist in the following hierarchy:

- standards : The standard database files contain the original force field information. These files are to reside in a directory that is specified in the file $HOME/.nwchemrc . There will be such a directory for each supported force field. These directories contain fragment files (with extension frg), segment files (with extension sgm) and a parameter file (with the name of the force field and with extension par).
- extensions : These database files contain generally accepted extensions to the original force field and are to reside in a separate directory that is specified in the file $HOME/.nwchemrc . There will be such a directory for each supported force field. These directories contain fragment files (with extension frg), segment files (with extension sgm) and a parameter file (with the name of the force field and with extension par).
- contributed : These database files contain contributed definitions, also required for the quality assurance tests and are to reside in a separate directory that is specified in the file $HOME/.nwchemrc . There will be such a directory for each supported force field. These directories contain fragment files (with extension frg), segment files (with extension sgm) and a parameter file (with the name of the force field and with extension par).
- user preferences : These database files contain user preferred extensions to the original force field and are to reside in a separate directory that is specified in the file $HOME/.nwchemrc . Separate directories of this type should be defined for each supported force field. This directory may contain fragment files (with extension frg), segment files (with extension sgm) and a parameter file (with the name of the force field and with extension par).

- temporary files : Temporary database files contain user preferred extensions to the original force field and are to reside in a separate directory that is specified in the file `$HOME/.nwchemrc`. There may be such a directory for each supported force field. This directory may contain fragment files (with extension frg), segment files (with extension sgm) and a parameter file (with the name of the force field and with extension par).
- current files : Database files that contain user preferred extensions to the original force field and are to reside in a separate directory that is specified in the file `$HOME/.nwchemrc`. Typically this will be the current working directory, although it may be defined as a specific directory. This directory may contain fragment files (with extension frg), segment files (with extension sgm) and a parameter file (with the name of the force field and with extension par). If not specified, files will be taken from the current directory.

Data is taken from the database files searched in the above order. If data is specified more than once, the last found values are used. For example, if some standard segment is redefined in a temporary file, the latter one will be used. This allows the user to redefine standards or extensions without having to modify those database files, which may reside in a generally available, non-modifyable directory. If a filename is specified rather than a directory, the filename indicates the parameter file definition. All other files (frg and sgm files) will be take from the specified directory.

The most common problems with the prepare module are

1. The format of the pdb file does not conform to the pdb standard. In particular, atom names need to correspond with definitions in the fragment and segment database files, and should adhere to IUPAC recommendations as adopted by the pdb standard. If this problem occurs, the pdb file will need to be corrected.

1. Non-standard segments may contain atoms that could not be atom typed with the existing typing rules in the force field parameter files. When this happens, additional typing rules can be included in the parameter file, or the fragment file may be manually typed.

1. Parameters for atom types or bonded interactions do not exist in the force field. When this happens, additional parameters may be defined in the parameter files, or the segment file may be edited to include explicit parameters.

17.1.1 Default database directories

The file `$HOME/.nwchemrc` may contain the following entries that determine which files are used by the prepare module.

```
ffield <string ffname>
```

This entry specifies the default force field. Database files supplied with NWChem currently support values for ffname of amber, referring to AMBER95, and charmm, referring to the academic CHARMM22 force field.

```
<string ffname>_(1-9) <string ffdir>[{<string parfile>}]
```

Entries of this type specify the directory ffdir in which force field database files can be found. Optionally the parameterfile in this directory may be specified as parfile. The prepare module will only use files in directories specified here. One exception is that files in the current work directory will be used if no directory with current files is specified. The directories are read in the order 1-9 with duplicate parameters taken from the last occurrence found. Note that multiple parameter files may be specified that will be read in the order in which they are specified.

```
<string solvnam> <string solvfil>
```

This entry may be used to identify a pure solvent restart file solvfil by a name solvnam

An example file `$HOME/.nwchemrc` is:

```
ffield amber
amber_1 /soft/nwchem/share/amber/amber_s/amber99.par,spce.par
amber_2 /soft/nwchem/share/amber/amber_x/
amber_3 /usr/people/username/data/amber/amber_u/
spce /soft/nwchem/share/solvents/spce.rst
charmm_1 /soft/nwchem/share/charmm/charmm_s/
charmm_2 /soft/nwchem/share/charmm/charmm_x/
```

17.1.2 System name and coordinate source

```
system <string sys_calc>
```

The system name can be explicitly specified for the prepare module. If not specified, the system name will be taken from a specification in a previous md input block, or derived from the run time database name.

```
source ( pdb | rtdb )
```

The source of the coordinates can be explicitly specified to be from a PDB formatted file sys.pdb, or from a geometry object in the run time database. If not specified, a pdb file will be used when it exists in the current directory or the rtdb geometry otherwise.

```
model <integer modpdb default 0>
```

If a PDB formatted source file contains different MODELs, the model keyword can be used to specify which MODEL will be used to generate the topology and restart file. If not specified, the first MODEL found on the PDB file will be read.

```
altloc <character locpdb default ' '>
```

The altloc keyword may be used to specify the use of alternate location coordinates on a PDB file.

```
chain <character chnpdb default ' '>
```

The chain keyword may be used to specify the chain identifier for coordinates on a PDB file.

```
histidine ( hid | hie | hip )
```

specifies the default protonation state of histidine.

```
sscyx
```

Keyword sscyx may be used to rename cysteine residues that form sulphur bridges to CYX.

```
hbuild
```

Keyword hbuild may be used to add hydrogen atoms to the unknown segments of the structure found on the pdb file. Placement of hydrogen atoms is based on geometric criteria, and the resulting fragment and segment files should be carefully examined for correctness.

The database directories are used as specified in the file .nwchemrc . Specific definitions for the force field used may be changed in the input file using

```
directory_(1-9) <string ffdir> [<string parfile>]
```

17.1.3 e file generation

If no existing e file is present in the current directory, or if the new_seq keyword was specified in the prepare input deck, a new sequence file is generated from information from the pdb file, and the following input directives.

```
maxscf <integer maxscf default 20>
```

Variable maxscf specifies the maximum number of atoms in a segment for which partial atomic charges will be determined from an SCF calculation followed by RESP charge fitting. For larger segments a crude partial charge guestimation will be done.

```
qscale <real qscale default 1.0>
```

Variable qscale specifies the factor with which SCF/RESP determined charges will be multiplied.

```
modify sequence { <integer sgmnum>:<string sgmnam> }
```

This command specifies that segment sgmnam should be used for segment with number sgmnum. This command can be used to specify a particular protonation state. For example, the following command specifies that residue 114 is a hystidine protonated at the N_ε site and residue 202 is a hystidine protonated at the N_δ site:

```
modify sequence 114:HIE 202:HID
```

Links between atoms can be enforced with

```
link <string atomname> <string atomname>
```

For example, to link atom SG in segment 20 with atom FE in segment 55, use:

```
link 20:_SG 55:FE
```

The format of the sequence file is given inthis table. In addition to the list of segments this file also includes links between non-standard segments or other non-standard links. These links are generated based on distances found between atoms on the pdb file. When atoms are involved in such non-standard links that have not been identified in the fragment of segment files as a non-chain link atom, the prepare module will ignore these links and report them as skipped. If one or more of these links are required, the user has to include them with explicit link directives in the sequence file, making them forced links. Alternatively, these links can be made forced-links by changing link into LINK in the sequence file.

```
fraction { <integer imol> }
```

Directive fraction can be used to separate solute molecules into fractions for which energies will be separately reported during molecular dynamics simulations. The listed molecules will be the last molecule in a fraction. Up to 10 molecules may be specified in this directive.

```
counter <integer num> <string ion>
```

Directive counter adds num counter ions of type ion to the sequence file. Up to 10 counter directives may appear in the input block.

```
counter <real factor>
```

This directive scales the counter ion charge by the specified factor in the determination of counter ions positions.

17.1.4 Topology file generation

```
new_top [ new_seq ]
```

Keyword new_top is used to force the generation of a new topology file. An existing topology file for the system in the current directory will be overwritten. If keyword new_seq is also specified, an existing sequence file will also be overwritten with a newly generated file.

```
amber | charmm
```

The prepare module generates force field specific fragment, segment and topology files. The force field may be explicitly specified in the prepare input block by specifying its name. Currently AMBER and CHARMM are the supported force fields. A default force field may be specified in the file $HOME/.nwchemrc .

```
standard <string dir_s>[<string par_s>]
extensions <string dir_x>[<string par_x>]
contributed <string dir_q>[<string par_q>]
user <string dir_u>[<string par_u>]
temporary <string dir_t>[<string par_t>]
current <string dir_c>[<string par_c>]
```

The user can explicitly specify the directories where force field specific databases can be found. These include force field standards, extensions, quality assurance tests, user preferences, temporary , and current database files. Defaults for the directories where database files reside may be specified in the file $HOME/.nwchemrc for each of the supported force fields. Fragment, segment and sequence files generated by the prepare module are written in the temporary directory. When not specified, the current directory will be used. Topology and restart files are always created in the current directory.

The following directives control the modifications of a topology file. These directives are executed in the order in which they appear in the prepare input deck. The topology modifying commands are not stored on the run-time database and are, therefor, not persistent.

```
modify atom <string atomname> [set <integer mset> | initial | final] \
  ( type <string atomtyp> |  charge <real atomcharge> |  \
  polar <real atompolar> | dummy | self | quantum | quantum_high )
```

These modify commands change the atom type, partial atomic charge, atomic polarizability, specify a dummy, self-interaction and quantum atom, respectively. If mset is specified, the modification will only apply to the specified set, which has to be 1, 2 or 3. If not specified, the modification will be applied to all three sets. The quantum region in QM/MM simulations is defined by specifying atoms with the quantum or quantum_high label. For atoms defined quantum_high basis sets labeled X_H will be used. The atomnam should be specified as :, where isgm is the segment number, and name is the atom name. A leading blank in an atom name should be substituted with an underscore. The modify commands may be combined. For example, the following directive changes for the specified atom the charge and atom type in set 2 and specifies the atom to be a dummy in set 3.

```
modify atom 12:_C1 set 2 charge 0.12 type CA set 3 dummy
```

With the following directives modifications can be made for entire segments.

```
modify segment <integer isgm> \
  [protonation <integer iprot> | set <integer mset> | initial | final] \
  ( dummy | self | uncharged | quantum | quantum_high )
```

where protonation specifies a modification of the default protonation state of the segment as specified in the segment file. This option only applies to Q-HOP simulations.

Modifications to bonded interaction parameters can be made with the following modify commands.

```
modify ( bond <string atomtyp> <string atomtyp> |  \
    angle <string atomtyp> <string atomtyp> <string atomtyp> |       \
    torsion <string atomtyp> <string atomtyp> <string atomtyp>       \
       <string atomtyp> [ multiplicity <integer multip> ] |      \
    plane <string atomtyp> <string atomtyp> <string atomtyp>         \
       <string atomtyp> ) [set <integer mset> | initial | final] \
    <real value> <real forcon>
```

where atomtyp and mset are defined as above, multip is the torsion ultiplicity for which the modification is to be applied,

value is the reference bond, angle, torsion angle of out-of-plane angle value respectively, and forcon is the force constant for bond, angle, torsion angle of out-of-plane angle. When multip or mset are not defined the modification will be applied to all multiplicities and sets, respectively, for the identified bonded interaction.

After modifying atoms to quantum atoms the bonded interactions in which only quantum atoms are involved are removed from the bonded lists using

```
update lists
```

Error messages resulting from parameters not being defined for bonded interaction in which only quantum atoms are involved are ignored using

```
ignore
```

To specify that a free energy calculation will be carried out using the topology file, the following keyword needs to be specified,

```
free
```

To specify that a Q-HOP simulation will be carried out using the topology file, the following keyword needs to be specified,

```
qhop
```

To specify that only the first set of parameters should be used, even if multiple sets have been defined in the fragment or segment files, the following keyword needs to be specified,

```
first
```

Note that keywords free, qhop and qhop are mutually exclusive.

17.1.5 Appending to an existing topology file

```
noe <string atom1> <string atom3> \
 <real dist1> <real dist2>  <real dist3> <real forc1> <real forc2>
```

This directive specifies a distance restraint potential between atoms *atom1* and *atom2*, with a harmonic function with force constant *forc1* between *dist1* and *dist2*, and a harmonic function with force constant *forc2* between *dist2* and *dist3*. For distances shorter than *dist1* or larger than *dist3*, a constant force is applied such that force and energy are continuous at *dist1* and *dist3*, respectively. Distances are given in nm, force constants in\(kJ mol^{-1} nm^{-2}\).

```
select <integer isel> { <string atoms> }
```

Directive select specifies a group of atoms used in the definition of potential of mean force potentials.

The selected atoms are specified by the string atoms which takes the form

```
[{isgm [ - jsgm ] [,]} [:] [{aname[,]}]
```

For example, all carbon and oxygen atoms in segments 3 and 6 through 12 are selected for group 1 by

```
3,6-12:_C????,_O????
pmf [all] [bias] zalign <integer isel> <real forcon1> <real forcon2>
pmf [combine] [bias] xyplane <integer isel> <real forcon1> <real forcon2>
pmf [constraint] [bias] (distance | zdistance) <integer isel> <integer jsel> \
        <real dist1> <real dist2> <real forcon1> <real forcon2>
pmf [bias] angle <integer isel> <integer jsel> <integer ksel> \
        <real angle1> <real angle2> <real forcon1> <real forcon2>
pmf [bias] torsion <integer isel> <integer jsel> <integer ksel> <integer lsel> \
        <real angle1> <real angle2> <real forcon1> <real forcon2>
pmf [bias] basepair <integer isel> <integer jsel> \
        <real dist1> <real dist2> <real forcon1> <real forcon2>
pmf [bias] (zaxis | zaxis-cog) <integer isel> <integer jsel> <integer ksel> \
        <real dist1> <real dist2> <real forcon1> <real forcon2>
```

Directive pmf specifies a potential of mean force potential in terms of the specified atom selection. Option zalign specifies the atoms in the selection to be restrained to a line parallel to the z-axis. Option xyplane specifies the atoms in the selection to be restrained to a plane perpendicular to the z-axis. Options distance, angle and torsion, are defined in terms of the center of geometry of the specified atom selections. Keyword basepair is used to specify a harmonic potential between residues isel and jsel. Keywords zaxis and zaxis-cog can be used to pull atoms toward the z-axis. Option all may be specified to apply an equivalent pmf to each of the equivalent solute molecules in the system. Option combine may be specified to apply the specified pmf to the atoms in all of the equivalent solute molecules. Option constraint may be specified to a distance pmf to treat the distance as a constraint. Option bias may be specified to indicate that this function should be treated as a biasing potential. Appropriate corrections to free energy results will be evaluated.

17.1.6 Generating a restart file

```
new_rst
```

Keyword new_rst will cause an existing restart file to be overwritten with a new file.

The follwing directives control the manipulation of restart files, and are executed in the order in which they appear in the prepare input deck.

```
solvent name <string*3 slvnam default 'HOH'> \
        model <string slvmdl default 'spce'>
```

The solvent keyword can be used to specify the three letter solvent name as expected on the PDB formatted file, and the name of the solvent model for which solvent coordinates will be used.

```
solvate   [ < real rshell default 1.2 > ] \
        ( [ cube [ <real edge> ]] |  \
          [ box [ <real xedge> [ <real xedge> [ <real xedge> ]]]] | \
          [ sphere <real radius> ] |
          [ troct <real edge> ])
```

Solvation can be specified to be in a cubic box with specified edge, rectangular box with specified edges, or in a sphere with specified radius. Solvation in a cube or rectangular box will automatically also set periodic boundary conditions. Solvation in a sphere will only allow simulations without periodic boundary conditions. The size of the cubic and rectangular boxes will be expanded by a length specified by the expand variable. If no shape is specified, solvation will be done for a cubic box with an edge that leaves rshell nm between any solute atom and a periodic image of any solute atom after the solute has been centered. An explicit write is not needed to write the restart file. The solvate will write out a file sys_calc.rst. If not specified, the dimension of the solvation cell will be as large as to have at least a distance of rshell nm between any solute atom and the edge of the cell. The experimental troct directive generates a truncated octrahedral box.

```
touch <real touch default 0.23>
```

The variable touch specifies the minimum distance between a solvent and solute atom for which a solvent molecule will be accepted for solvation.

```
envelope <real xpndw default 0.0>
```

sets the expand vealues to be used in solvate operations.

```
expand <real xpndw default 0.1>
```

The variable xpndw specifies the size in nm with which the simulation volume will be increased after solvation.

```
read [rst | rst_old | pdb] <string filename>
write [rst | [solute [<integer nsolvent>]] ( [large] pdb | xyz)] <string filename>
```

These directives read and write the file filename in the specified format. The solute option instructs to write out the coordinates for solute and all, or if specified the first nsolvent, crystal solvent molecules only. If no format is specified, it will be derived from the extension of the filename. Recognized extensions are rst, rst_old (read only), pdb, xyz (write only) and pov (write only). Reading and then writing the same restart file will cause the sub-block size information to be lost. If this information needs to be retained a shell copy command needs to be used. The large keyword allows PDB files to be written with more than 9999 residues. Since the PDB file will not conform to the PDB convention, this option should only be used if required. NWChem will be able to read the resulting PDB file, but other codes may not.

```
scale <real scale default -1.0>
```

This directive scales the volume and coordinates written to povray files. A negative value of scale (default) scales the coordinates to lie in [-1:1].

```
cpk [<real cpk default 1.0>]
```

This directive causes povray files to contain cpk model output. The optional value is used to scale the atomic radii. A neagtive value of cpk resets the rendering to stick.

```
center | centerx | centery | centerz
```

These directives center the solute center of geometry at the origin, in the y-z plane, in the x-z plane or in the x-y plane, respectively.

```
orient
```

This directive orients the solute principal axes.

```
translate [atom | segment | molecule] \
   <integer itran> <integer itran> <real xtran(3)>
```

This directive translates solute atoms in the indicated range by xtran, without checking for bad contacts in the resulting structure.

```
rotate [atom | segment | molecule] \
   <integer itran> <integer itran> <real angle> <real xrot(3)>
```

This directive rotates solute atoms in the indicated range by angle around the vector given by xrot,, without checking for bad contacts in the resulting structure.

```
remove solvent [inside | outside] [x <real xmin> <real xmax>] \
[y <real ymin> <real ymax>] [z <real zmin> <real zmax>]
```

This directive removes solvent molecules inside or outside the specified coordinate range.

```
periodic
```

This directive enables periodic boundary conditions.

```
vacuo
```

This directive disables periodic boundary conditions.

```
grid <integer mgrid default 24> <real rgrid default 0.2>
```

This directive specifies the grid size of trial counter-ion positions and minimum distance between an atom in the system and a counter-ion.

```
crop
```

prints minimum and maximum solute coordinates.

```
boxsize
```

specifies to redetermine the box size.

```
cube
```

specifies to redetermine the smallest cubic box size.

```
box <real xsize> <real ysize>  <real zsize>
```

The box directive resets the box size.

```
align <string atomi> <string atomj> <string atomk>
```

The align directive orients the system such that atomi and atomj are on the z-axis, and atomk in the x=y plane.

```
repeat [randomx | randomy | randomz] [chains | molecules | fractions ] \
 <integer nx> <integer ny> <integer nz> [<real dist>] [<real zdist>]
```

The repeat directive causes a subsequent write pdb directive to write out multiple copies of the system, with nx copies in the x, ny copies in the y, and nz copies in the z-direction, with a minimum distance of dist between any pair of atoms from different copies. If nz is -2, an inverted copy is placed in the z direction, with a separation of zdist nm. If dist is negative, the box dimensions will be used. For systems with solvent, this directive should be used with a negative dist. Optional keywords chains, molecules and fractions specify to write each repeating solute unit as a chain, to repeat each solute molecule, or each solute fraction separately. Optional keywords randomx, randomy, and randomz can be used to apply random rotations for each repeat unit around a vector through the center of geometry of the solute in the x, y or z direction.

```
skip <integer ix> <integer iy> <integer iz>
```

The skip directive can be used to skip single repeat unit from the repeat directive. Up to 100 skip directives may be specified, and will only apply to the previously specified repeat directive.

```
(collapsexy | collapsez) [ <integer nmoves>]
```

specifies to move all solute molecules toward the z-axis or x=y-plane, respectively, to within a distance of touch nm between any pair of atoms from different solute molecules. Parameter nmoves specifies the number of collapse moves that will be made. Monatomic ions will move with the nearest multi-atom molecule.

```
collapse_group <integer imol> <integer jmol>
```

specifies that molecule jmol will move together with molecule imol in collapse operations.

```
merge <real xtran(3)> <string pdbfile>
```

specifies to merge the coordinates found on the specified pdb file into the current structure after translation by xtran(3).

17.2 Molecular dynamics

17.2.1 Introduction

17.2.1.1 Spacial decomposition

The molecular dynamics module of NWChem uses a distribution of data based on a spacial decomposition of the molecular system, offering an efficient parallel implementation in terms of both memory requirements and communication costs, especially for simulations of large molecular systems.

Inter-processor communication using the global array tools and the design of a data structure allowing distribution based on spacial decomposition are the key elements in taking advantage of the distribution of memory requirements and computational work with minimal communication.

In the spacial decomposition approach, the physical simulation volume is divided into rectangular cells, each of which is assigned to a processor. Depending on the conditions of the calculation and the number of available processors, each processor contains one or more of these spacially grouped cells. The most important aspects of this decomposition are the dependence of the cell sizes and communication cost on the number of processors and the shape of the cells, the frequent reassignment of atoms to cells leading to a fluctuating number of atoms per cell, and the locality of communication which is the main reason for the efficiency of this approach for very large molecular systems.

To improve efficiency, molecular systems are broken up into separately treated solvent and solute parts. Solvent molecules are assigned to the domains according to their center of geometry and are always owned by a one node. This avoids solvent-solvent bonded interactions crossing node boundaries. Solute molecules are broken up into segments, with each segment assigned to a processor based on its center of geometry. This limits the number of solute bonded interactions that cross node boundaries. The processor to which a particular cell is assigned is responsible for the calculation of all interactions between atoms within that cell. For the calculation of forces and energies in which atoms in cells assigned to different processors are involved, data are exchanged between processors. The number of neighboring cells is determined by the size and shape of the cells and the range of interaction. The data exchange that takes place every simulation time step represents the main communication requirements. Consequently, one of the main efforts is to design algorithms and data structures to minimize the cost of this communication. However, for very large molecular systems, memory requirements also need to be taken into account.

To compromise between these requirements exchange of data is performed in successive point to point communications rather than using the shift algorithm which reduces the number of communication calls for the same amount of communicated data.

For inhomogeneous systems, the computational load of evaluating atomic interactions will generally differ between cell pairs. This will lead to load imbalance between processors. Two algorithms have been implemented that allow for dynamically balancing the workload of each processor. One method is the dynamic resizing of cells such that cells gradually become smaller on the busiest node, thereby reducing the computational load of that node. Disadvantages of this method are that the efficiency depends on the solute distribution in the simulation volume and the redistribution of work depends on the number of nodes which could lead to results that depend on the number of nodes used. The second method is based on the dynamic redistribution of intra-node cell-cell interactions. This method represents a more coarse load balancing scheme, but does not have the disadvantages of the cell resizing algorithm. For most molecular systems the cell pair redistribution is the more efficient and preferred method.

The description of a molecular system consists of static and dynamic information. The static information does not change during a simulation and includes items such as connectivity, excluded and third neighbor lists, equilibrium values and force constants for all bonded and non-bonded interactions. The static information is called the topology of the molecular system, and is kept on a separate topology file. The dynamic information includes coordinates and velocities for all atoms in the molecular system, and is kept in a so-called restart file.

17.2.1.2 Topology

The static information about a molecular system that is needed for a molecular simulation is provided to the simulation module in a topology file. Items in this file include, among many other things, a list of atoms, their non-bonded parameters

for van der Waals and electrostatic interactions, and the complete connectivity in terms of bonds, angles and dihedrals.

In molecular systems, a distinction is made between solvent and solute, which are treated separately. A solvent molecule is defined only once in the topology file, even though many solvent molecules usually are included in the actual molecular system. In the current implementation only one solvent can be defined. Everything that is not solvent in the molecular system is solute. Each solute atom in the system must be explicitly defined in the topology.

Molecules are defined in terms of one or more segments. Typically, repetitive parts of a molecule are each defined as a single segment, such as the amino acid residues in a protein. Segments can be quite complicated to define and are, therefore, collected in a set of database files. The definition of a molecular system in terms of segments is a sequence.

Topology files are created using the prepare module.

17.2.1.3 Files

File names used have the form *system_calc.ext*, with exception of thetopology file, which is named *system*.top. Anything that refers to the definition of the chemical system can be used for *system*, as long as no periods or underlines are used. The identifier *calc* can be anything that refers to the type of calculation to be performed for the system with the topology defined. This file naming convention allows for the creation of a single topology file *system*.top that can be used for a number of different calculations, each identified with a different *calc*. For example, if crown.top is the name of the topology file for a crown ether, crown_em, crown_md, crown_ti could be used with appropriate extensions for the filenames for energy minimization, molecular dynamics simulation and multi-configuration thermodynamic integration, respectively. All of these calculations would use the same topology file crown.top.

The extensions <ext> identify the kind of information on a file, and are pre-determined.

dbg	debug file
frg	fragment file
gib	free energy data file
mri	free energy multiple run input file
mro	free energy multiple run output file
nw	**NWChem** input file
nwout	**NWChem** output file
out	molecular dynamics output file
pdb	PDB formatted coordinate file
prp	property file
qrs	quenched restart file, resulting from an energy minimization
rst	restart file, used to start or restart a simulation
seq	sequence file, describing the system in segments
sgm	segment file, describing segments
syn	synchronization time file
tst	test file
tim	timing analysis file
top	topology file, contains the static description of a system
trj	trajectory file

17.2.1.4 Databases

Database file supplied with NWChem and used by the prepare module are found in directories with name ffield*level*, where *ffield* is any of the supported force fields. The source of the data is identified by *level*, and can be

level	Description
s	original published data
x	additional published data
q	contributed data
u	user preferred data
t	user defined temporary data

The user is can replace these directories or add additional database files by specifying them in the*nwchemrc* file. or in the prepare input file.

The extension 1-9 defines the priority of database file.

frg	fragments
par	parameters
seq	sequences
sgm	segments

17.2.1.5 Force fields

Force fields recognized are

Keyword	Force field	Status
amber	AMBER99	AMBER95,GLYCAM also available
charmm	CHARMM	

17.2.2 Format of fragment files

Fragment files contain the basic information needed to specify all interactions that need to be considered in a molecular simulation. Normally these files are created by the prepare module. Manual editing is needed when, for example, the prepare module could not complete atom typing, or when modified charges are required.

The formats of files used in NWChem are listedhere.

17.2.3 Creating segment files

The prepare module is used to generate segment files from corresponding fragment files. A segment file contains all

information for the calculation of bonded and non-bonded interactions for a given chemical system using a specific force field.

Which atoms form a fragment is specified in the coordinate file, currently only in PDB format. The segment entries define three sets of parameters for each interaction.

Free energy perturbations can be performed using set 1 for the generation of the ensemble while using sets 2 and/or 3 as perturbations. Free energy multiconfiguration thermodynamic integration and multistep thermodynamic perturbation calculations are performed by gradually changing the interactions in the system from parameter set 2 to parameter set 3. These modifications can be edited into the segment files manually, or introduced directly into the topology file using the modify commands in the input for the prepare module.

17.2.4 Creating sequence files

A sequence file describes a molecular system in terms of segments. This file is generated by the prepare module for the molecular system provided on a PDB-formatted coordinate file

17.2.5 Creating topology files

The topology describes all static information that describes a molecular system. This includes the connectivity in terms of bond-stretching, angle-bending and torsional interactions, as well as the non-bonded van der Waals and Coulombic interactions.

The topology of a molecular system is generated by the prepare module from the sequence in terms of segments as specified on the PDB file. For each unique segment specified in this file the segment database directories are searched for the segment definition. For segments not found in one of the database directories a segment definition is generated in the temporary directory if a fragment file was found. If a fragment file could not be found, it is generated by the prepare module base on what is found on the PDB file.

When all segments are found or created, the parameter substitutions are performed, using force field parameters taken from the parameter databases. After all lists have been generated the topology is written to a local topology file <system>.top.

17.2.6 Creating restart files

Restart files contain all dynamical information about a molecular system and are created by the prepare module if a topology file is available. The prepare module will automatically generate coordinates for hydrogen atoms and monatomic counter ions not found on the PDB formatted coordinate file, if no fragment or segment files were generated using that PDB file.

The prepare module has a number of other optional input command, including solvation.

17.2.7 Molecular simulations

The type of molecular dynamics simulation is specified by the NWChem task directive.

```
task md [ energy | optimize | dynamics | thermodynamics ]
```

where the theory keyword md specifies use of the molecular dynamics module, and the operation keyword is one of

- *energy* for single configuration energy evaluation
- *optimize* for energy minimization
- *dynamics* for molecular dynamics simulations and single step thermodynamic perturbation free energy molecular

dynamics simulations

- *thermodynamics* for combined multi-configuration thermodynamic integration and multiple step thermodynamic perturbation free energy molecular dynamics simulations.

17.2.8 System specification

The chemical system for a calculation is specified in the topology and restart files. These files should be created using the utilities nwtop and nwrst before a simulation can be performed. The names of these files are determined from the required system directive.

```
system <string systemid>_<string calcid>
```

where the strings systemid and calcid are user defined names for the chemical system and the type of calculation to ber performed, respectively. These names are used to derive the filenames used for the calculation. The topoly file used will be systemid.top, while all other files are named systemid_calcid.ext.

17.2.9 Restarting and continuing simulations

```
finish
```

Specifies that the current job will finish a previous, incomplete simulation, using the input data that have been recorded by that previous run in the restart file. Most of the input in the current md input block will be ignored.

```
resume
```

Specifies that the current job will be an extension of a previous simulation, using most of the input data that have been recorded by that previous run in the restart file. Typically the input in the current md input block defines a larger number of steps than the previous job.

17.2.10 Parameter set

```
set <integer iset>
```

Specifies the use of parameter set <iset> for the molecular dynamics simulation. The topology file contains three separate parameters sets that can be used. The default for <iset> is 1.

```
lambda <integer ilambda> <integer ilambda>
```

Specifies the use of parameter set for the ilambda-th of mlambda steps.

```
pset <integer isetp1> [<integer isetp2>]
```

Specifies the parameter sets to be used as perturbation potentials in single step thermodynamic perturbation free energy evaluations, where <isetp1> specifies the first perturbation parameter set and <isetp2> specifies the second perturbation parameter set. Legal values for <isetp1> are 2 and 3. Legal value for <isetp2> is 3, in which case <isetp1> can only be 2. If specified, <iset> is automatically set to 1.

```
pmf [ equilharm <integer npmfc> | scale <real facpmf>]
```

Specifies that any potential of mean force functions defined in the topology files are to be used. If equilharm is specified, the first npmfc dynamics steps will use a harmonic potential in stead of any pmf constraint. If scale is specified, all pmf force constants are scaled by a factor facpmf.

```
distar [draver [<integer ndaver default 1>]]
  [scale <real drsscl>]
  [after <integer nfdrss>]
```

Specifies that any distance restraint functions defined in the topology files are to be used.

```
qhop [<integer nfhop default 10>]
  [<real rhop default 0.35>]
  [<real thop default 0.02>]
```

Specifies that a Q-HOP simulation is to be carried out with attempted proton hops every*nfhop* steps, a cutoff for the donor-acceptor pair distance of *rhop* nm, and a minimum time before back hopping can occur of*thop* ps.

17.2.11 Energy minimization algorithms

The energy minimization of the system as found in the restart file is performed with the following directives. If both are specified, steepest descent energy minimization precedes conjugate gradient minimization.

```
sd <integer msdit> [init <real dx0sd>] [min <real dxsdmx>] \
        [max <real dxmsd>]
```

Specifies the variables for steepest descent energy minimizations, where <msdit> is the maximum number of steepest descent steps taken, for which the default is 100, <dx0sd> is the initial step size in nm for which the default is 0.001,<dxsdmx> is the threshold for the step size in nm for which the default is 0.0001, and <dxmsd> is the maximum allowed step size in nm for which the default is 0.05.

```
cg <integer mcgit> [init <real dx0cg>] [min <real dxcgmx>] \
        [cy <integer ncgcy>]
```

Specifies the variables for conjugate gradient energy minimizations, where <mcgit> is the maximum number of conjugate gradient steps taken, for which the default is 100, <dx0cg> is the initial search interval size in nm for which the default is 0.001, <dxcgmx> is the threshold for the step size in nm for which the default is 0.0001, and<ncgcy> is the number of conjugate gradient steps after which the gradient history is discarded for which the default is 10. If conjugate gradient energy minimization is preceded by steepest descent energy minimization, the search interval is set to twice the final step of the steepest descent energy minimization.

17.2.12 Multi-configuration thermodynamic integration

The following keywords control free energy difference simulations. Multi-configuration thermodynamic integrations are always combined with multiple step thermodynamic perturbations.

```
(forward | reverse) [[<integer mrun> of] <integer maxlam>]
```

Specifies the direction and number of integration steps in free energy evaluations, with forward being the default direction. <mrun> is the number of ensembles that will be generated in this calculation, and<maxlam> is the total number of ensembles to complete the thermodynamic integration. The default value for <maxlam> is 21. The default value of <mrun> is the value of <maxlam> .

```
error <real edacq>
```

Specifies the maximum allowed statistical error in each generated ensemble, where <edacq> is the maximum error allowed in the ensemble average derivative of the Hamiltonian with respect to λ with a default of 5.0 kJ mol^{-1}.

```
drift <real ddacq>
```

Specifies the maximum allowed drift in the free energy result, whereis the maximum drift allowed in the ensemble average derivative of the Hamiltonian with respect to λ with a default of 5.0 kJ mol^{-1} ps^{-1}.

factor <real fdacq>

Specifies the maximum allowed change in ensemble size where <fdacq> is the minimum size of an ensemble relative to the previous ensemble in the calculation with a default value of 0.75.

decomp

Specifies that a free energy decomposition is to be carried out. Since free energy contributions are path dependent, results from a decomposition analysis can no be unambiguously interpreted, and the default is not to perform this decomposition.

sss [delta <real delta>]

Specifies that atomic non-bonded interactions describe a dummy atom in either the initial or final state of the thermodynamic calculation will be calculated using separation-shifted scaling, where <delta> is the separation-shifted scaling factor with a default of 0.075 nm^2. This scaling method prevents problems associated with singularities in the interaction potentials.

new | renew | extend

Specifies the initial conditions for thermodynamic calculations. new indicates that this is an initial mcti calculation, which is the default. renew instructs to obtain the initial conditions for each λ from the mro-file from a previous mcti calculation, which has to be renamed to an mri-file. The keyword extend will extend a previous mcti calculation from the data read from an mri-file.

17.2.13 Time and integration algorithm directives

Following directives control the integration of the equations of motion.

leapfrog | leapfrog_bc

Specifies the integration algorithm, where leapfrog specifies the default leap frog integration, and leapfrog_bc specifies the Brown-Clarke leap frog integrator.

guided [<real fguide default 0.2> [<real tguide default 0.2>]]

Specifies the use of the guided molecular dynamics simulation technique. Variable*fguide* defines the fraction of the averaged forces g to be added to the forcesf^f evaluated using the force field functions to obtain the forces$f$ used to advance the coordinates.

$$f_i=f^{f}_i+fguide * g_{i-1}$$

Variable tguide defines the length of the averaging relative to the timestep Δt.

$$g_i = {\Delta t\over tguide} f_i + \left(1- {\Delta t\over tguide}\right) g_{i-1}$$

The current implementation is still under development.

equil <integer mequi>

Specifies the number of equilibration steps <mequi> , with a default of 100.

data <integer mdacq> [over <integer ldacq>]]

Specifies the number of data gathering steps <mdacq> with a default of 500. In multi-configuration thermodynamic integrations <mequi> and <mdacq> are for each of the ensembles, and variable <ldacq> specifies the minimum number of data gathering steps in each ensemble. In regular molecular dynamics simulations <ldacq> is not used. The default value for

<ldacq> is the value of <mdacq> .

time <real stime>

Specifies the initial time <stime> of a molecular simulation in ps, with a default of 0.0.

step <real tstep>

Specifies the time step <tstep> in ps, with 0.001 as the default value.

17.2.14 Ensemble selection

Following directives control the ensemble type.

```
isotherm [<real tmpext> [<real tmpext2>]] [trelax <real tmprlx> [<real tmsrlx>]] \
    [anneal [<real tann1>] <real tann2>]
```

Specifies a constant temperature ensemble using Berendsen's thermostat, where <tmpext> is the external temperature with a default of 298.15 K, and <tmprlx> and <tmsrlx> are temperature relaxation times in ps with a default of 0.1. If only<tmprlx> is given the complete system is coupled to the heat bath with relaxation time <tmprlx> . If both relaxation times are supplied, solvent and solute are independently coupled to the heat bath with relaxation times <tmprlx> and <tmsrlx> , respectively. If keyword anneal is specified, the external temperature will change from tmpext to tempext2 between simulation time tann1 and tann2

```
isobar [<real prsext>] [trelax <real prsrlx> ] \
    [compress <real compr>] [anisotropic] [xy | z | xy-z]
```

Specifies a constant pressure ensemble using Berendsen's piston, where <prsext> is the external pressure with a default of 1.025 10^5 Pa, <prsrlx> is the pressure relaxation time in ps with a default of 0.5, and<compr> is the system compressibility in \(m{^2}N^{-1}\) with a default of 4.53E-10. Optional keywords xy, z and xy-z may be used to specify that pressure scaling is to be applied in the x and y dimension only, the z dimension only, or, in all three dimensions with identical scaling in the x and y dimension. The last option requires that anisotropic is also specified.

17.2.15 Velocity reassignments

Velocities can be periodically reassigned to reflect a certain temperature.

```
vreass <integer nfgaus> <real tgauss>
    [fraction [<real frgaus default 0.5]]
    [once]
    [(first | initial)] [(last | final)]
```

Specifies that velocities will be reassigned every <nfgaus> molecular dynamics steps, reflecting a temperature of <tgauss> K. The default is not to reassign velocities, i.e. <nfgaus> is 0. Keyword fraction allows the specification of the fraction of the new velocities are random. Keyword once specifies that velocity reassignment only should be done in the first step. Keywords first or initial and last or final specify that velocity reassigment should only be applied in the first and last window of multiple run simulations.

17.2.16 Cutoff radii

Cutoff radii can be specified for short range and long range interactions.

```
cutoff [short] <real rshort> [long <real rlong>] \
    [qmmm <real rqmmm>]
```

Specifies the short range cutoff radius <rshort> , and the long range cutoff radius <rlong> in nm. If the long range cutoff radius

is larger than the short range cutoff radius the twin range method will be used, in which short range forces and energies are evaluated every molecular dynamics step, and long range forces and energies with a frequency of <nflong> molecular dynamics steps. Keyword qmmm specifies the radius of the zone around quantum atoms defining the QM/MM bare charges. The default value for <rshort> , <rlong> and <rqmmm> is 0.9 nm.

17.2.17 Polarization

First order and self consistent electronic polarization models have been implemented.

```
polar (first | scf [[<integer mpolit>] <real ptol>])
```

Specifies the use of polarization potentials, where the keyword first specifies the first order polarization model, and scf specifies the self consistent polarization field model, iteratively determined with a maximum of <mpolit> iterations to within a tolerance of <ptol> D in the generated induced dipoles. The default is not to use polarization models.

17.2.18 External electrostatic field

```
field <real xfield> [freq <real xffreq>] [vector <real xfvect(1:3)>]
```

Specifies an external electrostatic field, where <xfield> is the field strength, <xffreq> is the frequency in MHz and <xfvect> is the external field vector.

17.2.19 Constraints

Constraints are satisfied using the SHAKE coordinate resetting procedure.

```
shake [<integer mshitw> [<integer mshits>]]  \
    [<real tlwsha> [<real tlssha>]]
```

Specifies the use of SHAKE constraints, where <mshitw> is the maximum number of solvent SHAKE iterations, and <mshits> is the maximum number of solute SHAKE iterations. If only <mshitw> is specified, the value will also be used for <mshits> . The default maximum number of iterations is 100 for both. <tlwsha> is the solvent SHAKE tolerance in nm, and <tlssha> is the solute SHAKE tolerance in nm. If only <tlwsha> is specified, the value given will also be used for <tlssha> . The default tolerance is 0.001 nm for both.

```
noshake (solvent | solute)
```

Disables SHAKE and treats the bonded interaction according to the force field.

17.2.20 Long range interaction corrections

Long range electrostatic interactions are implemented using the smooth particle mesh Ewald technique, for neutral periodic cubic systems in the constant volume ensemble, using pair interaction potentials. Particle-mesh Ewald long range interactions can only be used in molecular dynamics simulations using effective pair potentials, and not in free energy simulations, QMD or QM/MM simulations.

```
pme [grid <integer ng>] [alpha <real ealpha>] \
   [order <integer morder>] [fft <integer imfft>]\
   [procs <integer nprocs>] [solvent]
```

Specifies the use of smooth particle-mesh Ewald long range interaction treatment, where ng is the number of grid points per dimension, ealpha is the Ewald coefficient in \(nm^{-1}\), with a default that leads to a tolerance of\(10^{-4}\) at the short range cutoff radius, and morder is order of the Cardinal B-spline interpolation which must be an even number and at least 4

(default value). A platform specific 3D fast Fourier transform is used, if available, when imfft is set to 2. nprocs can be used to define a subset of processors to be used to do the FFT calculations. If solvent is specified, the charge grid will be calculated from the solvent charges only.

```
react [<real dielec default 80.0>]
```

Specifies that a simple reaction field correction is used with a dielectric constant dielec. This is an experimental option that has not been well tested.

17.2.21 Fixing coordinates

Solvent or solute may be fixed using the following keywords.

```
( fix | free )
 solvent ( [<integer idfirst> [<integer idlast>]] |
          ( within | beyond) <real rfix> <string atomname> ) | \
 solute  ( [<integer idfirst> [<integer idlast>]] [ heavy | {<string atomname>}] |
          ( within | beyond) <real rfix> <string atomname> )
 [permanent]
```

For solvent the molecule numbers idfirst and idlastmay be specified to be the first and last molecule to which the directive applies. If omitted, the directive applies to all molecules. For solute, the segment numbers idfirst and idlastmay be specified to be the first and last segment to which the directive applies. If omitted, the directive applies to all segments. In addition, the keyword heavy may be specified to apply to all non hydrogen atoms in the solute, or a set of atom names may be specified in which a wildcard character ? may be used. Keyword permanent is used to keep the specification on the restart file for subsequent simulations.

17.2.22 Special options

```
import [<integer impfr default 1> [<integer impto default impfr> \
      [<integer nftri default 1>]]]
```

Specifies the import of frames impfr to impto with frequency nftri from a trajectory file with extension tri for which energies and forces are to be recalculated. This option only applied to task md energy.

```
detail
```

Specifies that moments of inertia and radii of gyration will be part of the recorded properties.

```
profile
```

Specifies that execution time profiling data will be part of the recorded properties.

```
scale <real scaleq>
```

Specifies that all charges will be scaled by the factro scaleq.

```
collapse [<real fcoll default 10.0> [ segment | z | xy ]
```

Specifies that additional forces directed to the origin of the simulation cell with strength fcoll will be applied to all solute molecules. If z or xy is specified, these forces will only apply in the specified dimension(s).

```
include fixed
```

Specifies that energies will be evaluated between fixed atoms. Normally these interactions are excluded from the pairlists.

```
eqm <real eqm>
```

Specifies the zero point of energy in QMD simulations.

```
atomlist
```

Specifies that pairlists will be atom based. Normally pairlist are charge group based.

17.2.23 Autocorrelation function

For the evaluation of the statistical error of multi-configuration thermodynamic integration free energy results a correlated data analysis is carried out, involving the calculation of the autocorrelation function of the derivative of the Hamiltonian with respect to the control variable λ.

```
auto <integer lacf> [fit <integer nfit>] [weight <real weight>]
```

Controls the calculation of the autocorrelation, where <lacf> is the length of the autocorrelation function, with a default of 1000, <nfit> is the number of functions used in the fit of the autocorrelation function, with a default of 15, and <weight> is the weight factor for the autocorrelation function, with a default value of 0.0.

17.2.24 Print options

Keywords that control print to the output file, with extension out. Print directives may be combined to a single directive.

```
print [topol [nonbond] [solvent] [solute]]     \
    [step <integer nfoutp> [extra] [energy]] \
    [stat <integer nfstat>]                  \
    [energies [<integer nfener>]]            \
    [forces [<integer nfforce>]]             \
    [matrix]                                 \
    [expect <integer npxpct>]                \
    [timing]                                 \
    [pmf [<integer iprpmf>]]                 \
    [out6]                                   \
    [dayout]
```

- Keyword topol specifies printing the topology information, where nonbond refers to the non-bonded interaction parameters, solvent to the solvent bonded parameters, and solute to the solute bonded parameters. If only topol is specified, all topology information will be printed to the output file.
- Keyword step specifies the frequency nfoutp of printing molecular dynamics step information to the output file. If the keyword extra is specified additional energetic data are printed for solvent and solute separately. If the keyword energy is specified, information is printed for all bonded solute interactions. The default for nfoutp is 0. For molecular dynamics simulations this frequency is in time steps, and for multi-configuration thermodynamic integration in λ-steps.
- Keyword stat specifies the frequency <nfstat> of printing statistical information of properties that are calculated during the simulation. For molecular dynamics simulation this frequency is in time steps, for multi-configuration thermodynamic integration in λ-steps.
- Keyword energies specifies the frequency nfener of printing solute bonded energies the output file for energy/import calculations. The default for nfener is 0.
- Keyword forces specifies the frequency nfforc of printing solute forces the output file for energy/import calculations. The default for nfforc is 0.
- Keyword matrix specifies that a solute distance matrix is to be printed.
- Keyword expect is obsolete.
- Keyword timing specifies that timing data is printed.
- Keyword pmf specifies that pmf data is printed every iprpmf steps. Keyword out6 specifies that output is written to

standard out in stead of the output file with extension out.

- Keyword dayout is obsolete.

17.2.25 Periodic updates

Following keywords control periodic events during a molecular dynamics or thermodynamic integration simulation. Update directives may be combined to a single directive.

```
update [pairs <integer nfpair default 1>]                \
    [long <integer nflong default 1>]                \
    [center <integer nfcntr default 0> [zonly | xyonly] \
        [fraction <integer idscb(1:5)>]           \
    [motion <integer nfslow default 0>]              \
    [analysis <integer nfanal default 0>]            \
    [rdf <integer nfrdf default 0>                   \
        [range <real rrdf>] [bins <integer ngl>]    \
```

- Keyword pairs specifies the frequency <nfpair> in molecular dynamics steps of updating the pair lists. The default for the frequency is 1. In addition, pair lists are also updated after each step in which recording of the restart or trajectory files is performed. Updating the pair lists includes the redistribution of atoms that changed domain and load balancing, if specified.

- Keyword long specifies the frequency <nflong> in molecular dynamics steps of updating the long range forces. The default frequency is 1. The distinction of short range and long range forces is only made if the long range cutoff radius was specified to be larger than the short range cutoff radius. Updating the long range forces is also done in every molecular dynamics step in which the pair lists are regenerated.

- Keywrod center specifies the frequency <nfcntr> in molecular dynamics steps in which the center of geometry of the solute(s) is translated to the center of the simulation volume. Optional keyword zonly or xyonly can be used to specify that centering will take place in the z-direction or in the xy-plane only. The solute fractions determining the solutes that will be centered are specified by the keyword fraction and the vector <idscb> , with a maximum of 5 entries. This translation is implemented such that it has no effect on any aspect of the simulation. The default is not to center, i.e. nfcntr is 0. The default fraction used to center solute is 1.

- Keyword motion specifies the frequency <nfslow> in molecular dynamics steps of removing the overall rotational and center of mass translational motion.

- Keyword analysis specifies the frequency <nfanal> in molecular dynamics steps of invoking the analysis module. This option is obsolete.

- Keyword rdf specifies the frequency <nfrdf> in molecular dynamics steps of calculating contributions to the radial distribution functions. The default is 0. The range of the radial distribution functions is given by <rrdf> in nm, with a default of the short range cutoff radius. Note that radial distribution functions are not evaluated beyond the short range cutoff radius. The number of bins in each radial distribution function is given by <ngl> , with a default of 1000. This option is no longer supported. If radial distribution function are to be calculated, a rdi files needs to be available in which the contributions are specified as follows.

Card	Format	Description
I-1	i	Type, 1=solvent-solvent, 2=solvent-solute, 3-solute-solute
I-2	i	Number of the rdf for this contribution
I-3	i	First atom number
I-4	i	Second atom number

17.2.26 Recording

The following keywords control recording data to file. Record directives may be combined to a single directive.

```
record [rest <integer nfrest> [keep]]     \
       [coord <integer nfcoor default 0>] \
       [wcoor <integer nfwcoo default 0>] \
       [scoor <integer nfscoo default 0>] \
       [veloc <integer nfvelo default 0>] \
       [wvelo <integer nfwvel default 0>] \
       [svelo <integer nfsvel default 0>] \
       [force <integer nfvelo default 0>] \
       [wforc <integer nfwvel default 0>] \
       [sforc <integer nfsvel default 0>] \
       [(prop | prop_average) <integer nfprop default 0>]  \
       [free <integer nffree default 1>]  \
       [sync <integer nfsync default 0>]  \
       [times <integer nftime default 0>] \
       [acf] [cnv] [fet]
       [binary] [ascii] [ecce] [argos]
```

- Keyword rest specifies the frequency <nfrest> in molecular dynamics steps of rewriting the restart file, with extension rst. For multi-configuration thermodynamic integration simulations the frequency is in steps in λ. The default is not to record. The restart file is used to start or restart simulations. The keyword keep causes all restart files written to be kept on disk, rather than to be overwritten.
- Keyword coord specifies the frequency <nfcoor> in molecular dynamics steps of writing coordinates to the trajectory file. This directive redefines previous coord, wcoor and scoor directives. The default is not to record.
- Keyword wcoor specifies the frequency <nfcoor> in molecular dynamics steps of writing solvent coordinates to the trajectory file. This keyword takes precedent over coord. This directive redefines previous coord, wcoor and scoor directives. The default is not to record.
- Keyword scoor specifies the frequency <nfscoo> in molecular dynamics steps of writing solute coordinates to the trajectory file. This keyword takes precedent over coord. This directive redefines previous coord, wcoor and scoor directives. The default is not to record.
- Keyword veloc specifies the frequency <nfvelo> in molecular dynamics steps of writing velocities to the trajectory file. This directive redefines previous veloc, wvelo and svelo directives. The default is not to record.

- Keyword wvelo specifies the frequency <nfvelo> in molecular dynamics steps of writing solvent velocitiesto the trajectory file. This keyword takes precedent over veloc. This directive redefines previous veloc, wvelo and svelo directives. The default is not to record.
- Keyword svelo specifies the frequency <nfsvel> in molecular dynamics steps of writing solute velocities to the trajectory file. This keyword takes precedent over veloc. This directive redefines previous veloc, wvelo and svelo directives. The default is not to record.
- Keyword force specifies the frequency <nfvelo> in molecular dynamics steps of writing forces to the trajectory file. This directive redefines previous vforce, wforc and sforc directives. The default is not to record.
- Keyword wforc specifies the frequency <nfvelo> in molecular dynamics steps of writing solvent forcesto the trajectory file. This keyword takes precedent over force. This directive redefines previous vforce, wforc and sforc directives. The default is not to record.
- Keyword sforc specifies the frequency <nfsvel> in molecular dynamics steps of writing solute forces to the trajectory file. This keyword takes precedent over force. This directive redefines previous vforce, wforc and sforc directives. The default is not to record.
- Keyword prop specifies the frequency <nfprop> in molecular dynamics steps of writing information to the property file, with extension prp. The default is not to record.
- Keyword prop_average specifies the frequency <nfprop> in molecular dynamics steps of writing average information to the property file, with extension prp. The default is not to record.
- Keyword free specifies the frequency <nffree> in multi-configuration thermodynamic integration steps to record data to the free energy data file, with extension gib. The default is 1, i.e. to record at every λ. This option is obsolete. All data are required to do the final analysis.
- Keyword sync specifies the frequency <nfsync> in molecular dynamics steps of writing information to the synchronization file, with extension syn. The default is not to record. The information written is the simulation time, the wall clock time of the previous MD step, the wall clock time of the previous force evaluation, the total synchronization time, the largest synchronization time and the node on which the largest synchronization time was found. The recording of synchronization times is part of the load balancing algorithm. Since load balancing is only performed when pair-lists are updated, the frequency <nfsync> is correlated with the frequency of pair-list updates <nfpair>. This directive is only needed for analysis of the load balancing performance. For normal use this directive is not used.
- Keyword times specifies the frequency <nfsync> in molecular dynamics steps of writing information to the timings file, with extension tim. The default is not to record. The information written is wall clock time used by each of the processors for the different components in the force evaluation. This directive is only needed for analysis of the wall clock time distribution. For normal use this directive is not used.
- Keywords acf, cnv and fet are obsolete.
- Keywords binary, ascii, ecce and argos are obsolete.

17.2.27 Program control options

```
load [reset]
    ( none |
      size [<real factld>] |
      sizez [<real factld>] | pairs |
     (pairs [<integer ldpair>] size [<real factld>]) )
    [last]
    [minimum]
    [average]
    [combination]
    [iotime]
    [experimental]
```

Determines the type of dynamic load balancing performed, where the default is none. Load balancing option size is resizing cells on a node, and pairs redistributes the cell-cell interactions over nodes. Keyword reset will reset the load balancing read from the restart file. The level of cell resizing can be influenced with *factld*. The cells on the busiest node are resized with a factor

$$\left(1 - factld * { {T_{sync} \over n_p} - t^{min}_{sync} \over t_{wall}} \right)^{1\over 3}$$

Where T_{sync} is the accumulated synchronization time of all nodes,n_p is the total number of nodes,t^{min}_{sync} is the synchronization time of the busiest node, andt_{wall} is the wall clock time of the molecular dynamics step. For the combined load balancing, ldpair is the number of successive pair redistribution load balancing steps in which the accumulated synchronization time increases, before a resizing load balancing step will be attempted. Load balancing is only performed in molecular dynamics steps in which the pair-list is updated. The default load balancing is equivalent to specifying

```
load pairs 10 size 0.75
```

Keyword last specifies that the load balancing is based on the synchronization times of the last step. This is the default. Keyword average specifies that the load balancing is based on the average synchronization times since the last load balancing step. Keyword minimum specifies that the load balancing is based on the minimum synchronization times since the last load balancing step. Keywords combination, iotime and experimental are experimental load balancing options that should not be used in production runs.

```
(pack | nopack)
```

Specifies if data are communicated in packed or unpacked form. The default is pack.

```
procs <integer npx> <integer npy> <integer npz>
```

Specifies the distribution of the available processors over the three Cartesian dimensions. The default distribution is chosen such that, <npx> * <npy> * <npz> = <np> and <npx> <= <npy> <= <npz> , where <npx> , <npy> and <npz> are the processors in the x, y and z dimension respectively, and <np> is the number of processors allocated for the calculation. Where more than one combination of <npx> , <npy> and <npz> are possible, the combination is chosen with the minimum value of<npx>+<npy>+<npz> . To change the default setting the following optional input option is provided.

```
cells <integer nbx> <integer nby> <integer nbz>
```

Specifies the distribution of cells, where <nbx> , <nby> and <nbz> are the number of cells in x, y and z direction, respectively. The molecular system is decomposed into cells that form the smallest unit for communication of atomic data between nodes. The size of the cells is per default set to the short-range cutoff radius. If long-range cutoff radii are used the cell size is set to half the long-range cutoff radius if it is larger than the short-range cutoff. If the number of cells in a dimension is less than the number of processors in that dimension, the number of cells is set to the number of processors.

```
extra <integer madbox>
```

Sets the number of additional cells for which memory is allocated. In rare events the amount of memory set aside per node is insufficient to hold all atomic coordinates assigned to that node. This leads to execution which aborts with the message that mwm or msa is too small. Jobs may be restarted with additional space allocated by where <madbox> is the number of additional cells that are allocated on each node. The default for <madbox> is 6. In some cases <madbox> can be reduced to 4 if memory usage is a concern. Values of 2 or less will almost certainly result in memory shortage.

```
mwm <integer mwmreq>
```

Sets the maximum number of solvent molecules <mwmreq> per node, allowing increased memory to be allocated for solvent molecules. This option can be used if execution aborted because mwm was too small.

```
msa <integer msareq>
```

Sets the maximum number of solute atoms <msareq> per node, allowing increased memory to be allocated for solute atoms. This option can be used if execution aborted because msa was too small.

mcells <integer mbbreq>

Sets the maximum number of cell pairs <mbbreq> per node, allowing increased memory to be allocated for the cell pair lists. This option can be used if execution aborted because mbbl was too small.

boxmin <real rbox>

Sets the minimum size of a cell. This directive is obsolete. The use of mcells is preferred.

segmentsize <real rsgm>

Sets the maximum size of a segment. This value is used to determine which segments at the boundary of the cutoff radius should be considered in the generation of the pairlists. This value is also determined by the prepare module and written to the restart file. Use of this directive is not needed for simulations that use the current prepare module to generate the restart file.

memory <integer memlim>

Sets a limit <memlim> in kB on the allocated amount of memory used by the molecular dynamics module. Per default all available memory is allocated. Use of this command is required for QM/MM simulations only.

expert

Enables the use of certain combinations of features that are considered unsafe. This directive should not be used for production runs.

develop <integer idevel>

Enables the use of certain development options specified by the integer idevel. This option is for development purposes only, and should not be used for production runs.

control <integer icntrl>

Enables the use of certain development options specified by the integer icntrl. This option is for development purposes only, and should not be used for production runs.

numerical

Writes out analytical and finite difference forces for test purposes.

server <string servername> <integer serverport>

Allows monitoring over a socket connection to the specified port on the named server of basic data as a simulation is running.

For development purposes debug information can be written to the debug file with extension dbg with

debug <integer idebug>

where *idebug* specifies the type of debug information being written.

For testing purposes test information can be written to the test file with extension tst with

test <integer itest>

where *itest* specifies the number of steps test information is written.

On some platforms prefetching of data can improve the efficiency. This feature can be turned on using

```
prefetch [<integer nbget>]
```

where *nbget* is the number of outstanding communication operations.

Application of periodic boundary conditions for the evaluation of forces can be controlled with

```
pbc ( atom | residue | molecule )
```

This option rarely needs to be used.

Autocorrelation functions for error analysis are controlled using

```
auto [ fit <integer iapprx> | weight <real weight> ]
```

This option is disabled in the current release.

Membrane system equilibration can be made more efficient using

```
membrane [ rotations ]
```

Constraining the center of mass of solute molecules in the xy plane is accomplished using

```
scmxy [<integer icmopt default 1>]
```

where *icmopt* determines if the constraint is mass weighted (2).

Radius of gyration calculations are enabled using

```
radius_gyration
```

Calculations of diffusion coefficients is enabled using

```
diffusion
```

This option is disabled in the current release.

```
comlim ( on | off )
```

is disabled

To limit the size of recoding files, new files are opened every *nfnewf* md steps using

```
batch <integer nfnewf>
```

17.3 Analysis

The analysis module is used to analyze molecular trajectories generated by the NWChem molecular dynamics module, or partial charges generated by the NWChem electrostatic potential fit module. This module should not de run in parallel mode.

Directives for the analysis module are read from an input deck,

```
analysis
 ...
end
```

The analysis is performed as post-analysis of trajectory files through using the task directive

```
task analysis
```

or

```
task analyze
```

17.3.1 System specification

```
system <string systemid>_<string calcid>
```

where the strings systemid and calcid are user defined names for the chemical system and the type of calculation to ber performed, respectively. These names are used to derive the filenames used for the calculation. The topoly file used will be systemid.top, while all other files are named systemid_calcid.ext.

17.3.2 Reference coordinates

Most analyses require a set of reference coordinates. These coordinates are read from a NWChem restart file by the directive,

```
reference <string filename>
```

where filename is the name of an existing restart file. This input directive is required.

17.3.3 File specification

The trajectory file(s) to be analyzed are specified with

```
file  <string filename>  [ <integer firstfile>   <integer lastfile> ]
```

where filename is an existing trj trajectory file. If firstfile and lastfile are specified, the specified filename needs to have a? wildcard character that will be substituted by the 3-character integer number from firstfile to lastfile, and the analysis will be performed on the series of files. For example,

```
file tr_md?.trj 3 6
```

will instruct the analysis to be performed on files tr_md003.trj, tr_md004.trj, tr_md005.trj and tr_md006.trj.

From the specified files the subset of frames to be analyzed is specified by

```
frames [ <integer firstframe default 1> ]  <integer lastframe>  \
    [ <integer frequency default 1> ]
```

For example, to analyze the first 100 frames from the specified trajectory files, use

```
frames 100
```

To analyze every 10-th frame between frames 200 and 400 recorded on the specified trajectory files, use

```
frames 200 400 10
```

A time offset can be specified with

```
time <real timoff>
```

Solute coordinates of the reference set and ech subsequent frame read from a trajectory file are translated to have the center of geometry of the specified solute molecule at the center of the simulation box. After this translation all molecules are folded back into the box according to the periodic boundary conditions. The directive for this operation is

```
center  <integer imol>  [ <integer jmol default imol> ]
```

Coordinates of each frame read from a trajectory file can be rotated using

```
rotate ( off | x | y | z ) <real angle units degrees>
```

If center was defined, rotation takes place after the system has been centered. The rotate directives only apply to frames read from the trajectory files, and not to the reference coordinates. Up to 100 rotate directives can be specified, which will be carried out in the order in which they appear in the input deck. rotate off cancels all previously defined rotate directives.

To perform a hydrogen bond analysis:

```
hbond [distance [[<real rhbmin default 0.0>] <real rhbmin>]] \
    [angle [<real hbdmin> [ <real hbdmax default pi>]]] \
    [solvent [<integer numwhb>]]
```

17.3.4 Selection

Analyses can be applied to a selection of solute atoms and solvent molecules. The selection is determined by

```
select ( [ super ] [ { <string atomlist> } ] |
  solvent <real range> | save <string filename> | read <string filename> )
```

where {atomlist} is the set of atom names selected from the specified residues. By default all solute atoms are selected. When keyword `super` is specified the selecion applies to the superimposition option.

The selected atoms are specified by the string atomlist which takes the form

```
[{isgm [ - jsgm ] [,]} [:] [{aname[,]}]
```

where isgm and jsgm are the first and last residue numbers, and aname is an atom name. In the atomname a question mark may be used as a wildcard character.

For example, all protein backbone atoms are selected by

```
select _N,_CA,_C
```

To select the backbone atoms in residues 20 to 80 and 90 to 100 only, use

```
select 20-80,90-100:_N,_CA,_C
```

This selection is reset to apply to all atoms after each file directive.

Solvent molecules within range nm from any selected solute atom are selected by

```
select solvent <real range>
```

After solvent selection, the solute atom selection is reset to being all selected.

The current selection can be saved to, or read from a file using the save and read keywords, respectively.

Some analysis are performed on groups of atoms. These groups of atoms are defined by

```
define <integer igroup> [<real rsel>] [solvent] { <string atomlist> }
```

The string atom in this definitions again takes the form

```
[{isgm [ - jsgm ] [,]} [:] [{aname[,]}]
```

where isgm and jsgm are the first and last residue numbers, and aname is an atom name. In the atomname a question mark may be used as a wildcard character.

Multiple define directive can be used to define a single set of atoms.

17.3.5 Coordinate analysis

To analyze the root mean square deviation from the specified reference coordinates:

```
rmsd
```

To analyze protein φ-ψ and backbone hydrogen bonding:

```
ramachandran
```

To define a distance:

```
distance <integer ibond> <string atomi> <string atomj>
```

To define an angle:

```
angle <integer iangle> <string atomi> <string atomj> <string atomk>
```

To define a torsion:

```
torsion <integer itorsion> <string atomi> <string atomj> \
                <string atomk> <string atoml>
```

To define a vector:

```
vector <integer ivector> <string atomi> <string atomj>
```

The atom string in these definitions takes the form

```
<integer segment>:<string atomname> | w<integer molecule>:<string atomname>
```

for solute and solvent atom specification, respectively.

To define charge distribution in z-direction:

```
charge_distribution <integer bins>
```

Analyses on atoms in a predefined group are specified by

```
group [<integer igroup> [periodic <integer ipbc>] \
    ( local [<real rsel default 0.0>] [<real rval default rsel>]
      <string function> )
```

where igroup specifies the group of atoms defined with a define directive. Keyword$_{\text{periodic}}$ can be used to specify the periodicity, ipbc=1 for periodicity in z, ipbc=2 for periodicity in x and y, and ipbc=3 for periodicity in x, y and z. Currently the only option is local which prints all selected solute atom with a distance between rsel and rval from the atoms defined in igroup. The actual analysis is done by the scan deirective. A formatted report is printed from group analyses using

```
report <string filename> local
```

Analyses on pairs of atoms in predefined groups are specified by

```
groups [<integer igroup> [<integer jgroup>]] [periodic [<integer ipbc default 3>]] \
    <string function> [<real value1> [<real value2>]] [<string filename>]
```

where *igroup* and *jgroup* are groups of atoms defined with a define directive. Keyword `periodic` specifies that periodic boundary conditions need to be applied in *ipbc* dimensions. The type of analysis is define by *function*, *value1* and *value2*. If *filename* is specified, the analysis is applied to the reference coordinates and written to the specified file. If no filename is given, the analysis is applied to the specified trajectory and performed as part of the scan directive. Implemented analyses defined by `<string function> [<real value1> [<real value2>]]` include

- distance to calculate the distance between the centers of geometry of the two specified groups of atoms, and
- distances to calculate all atomic distances between atoms in the specified groups that lie between *value1* and *value2*.

Coordinate histograms are specified by

```
histogram <integer idef> [<integer length>] zcoordinate <string filename>
```

where *idef* is the atom group definition number, *length* is the size of the histogram, zcoordinate is the currently only histogram option, and *filename* is the filname to which the histogram is written.

Order parameters are evalated using

```
order <integer isel> <integer jsel> <string atomi> <string atomj>
```

This is an experimental feature.

To write the average coordinates of a trajectory

```
average [super] <string filename>
```

To perform the coordinate analysis:

```
scan [ super ] <string filename>
```

which will create, depending on the specified analysis options files filename.rms and filename.ana. After the scan directive previously defined coordinate analysis options are all reset. Optional keyword `super` specifies that frames read from the trajectory file(s) are superimposed to the reference structure before the analysis is performed.

17.3.6 Essential dynamics analysis

Essential dynamics analysis is performed by

```
essential
```

This can be followed by one or more

```
project <integer vector> <string filename>
```

to project the trajectory onto the specified vector. This will create files filename with extensions frm or trj, val, vec, _min.pdb and _max.pdb, with the projected trajectory, the projection value, the eigenvector, and the minimum and maximum projection structure.

For example, an essential dynamics analysis with projection onto the first vector generating files firstvec.{trj, val, vec, _min.pdb, _max.pdb} is generated by

essential
project 1 firstvec

17.3.7 Trajectory format conversion

To write a single frame in PDB or XYZ format, use

```
write [<integer number default 1>] [super] [solute] <string filename>
```

To copy the selected frames from the specified trejctory file(s), onto a new file, use

```
copy  [solute] [rotate <real tangle>] <string filename>
```

To superimpose the selected atoms for each specified frame to the reference coordinates before copying onto a new file, use

```
super [solute] [rotate <real tangle>] <string filename>
```

The rotate directive specifies that the structure will make a full ratation every tangle ps. This directive only has effect when writing povray files.

The format of the new file is determined from the extension, which can be one of

- amb AMBER formatted trajectory file (obsolete)
- arc DISCOVER archive file
- bam AMBER unformatted trajectory file
- crd AMBER formatted trajectory file
- dcd CHARMM formatted trajectory file
- esp gOpenMol formatted electrostatic potential files
- frm ecce frames file (obsolete)
- pov povray input files
- trj NWChem trajectory file
- xyz NWChem trajectory in xyz format

If no extension is specified, a trj formatted file will be written.

A special tag can be added to frm and pov formatted files using

```
label <integer itag> <string tag>  [ <real rval default 1.0> ] \\
     [ <integer iatag> [ <integer jatag default iatag> ] [ <real rtag default 0.0> ] ]
     [ <string anam> ]
```

where tag number *itag* is set to the string *tag* for all atoms anam within a distance *rtag* from segments *iatag* through *jatag*. A question mark can be used in anam as a wild card character.

Atom rendering is specified using

```
render ( cpk | stick )  [ <real rval default 1.0> ] \\
      [ <integer iatag> [ <integer jatag default iatag> ] [ <real rtag default 0.0> ] ]
      [ <string anam> ]
```

for all atoms anam within a distance *rtag* from segments *iatag* through *jatag*, and a scaling factor of *rval*. A question mark can be used in anam as a wild card character.

Atom color is specified using

```
color ( <string color> | atom ) \\
    [ <integer iatag> [ <integer jatag default iatag> ] [ <real rtag default 0.0> ] ]
    [ <string anam> ]
```

for all atoms anam within a distance *rtag* from segments *iatag* through *jatag*. A question mark can be used in anam as a wildcard character.

For example, to display all carbon atoms in segments 34 through 45 in green and rendered cpk in povray files can be specified with

```
render cpk 34 45 _C??
color green 34 45 _C??
```

Coordinates written to a pov file can be scaled using

```
scale <real factor>
```

A zero or negative scaling factor will scale the coordinates to lie within [-1,1] in all dimensions.

The cpk rendering in povray files can be scaled by

```
cpk <real factor default 1.0>
```

The stick rendering in povray files can be scaled by

```
stick <real factor default 1.0>
```

The initial sequence number of esp related files is defined by

```
index <integer index default 1>
```

A sequence of trajectory files with unequal lengths can be converted to files with al*hclean* frames using

```
clean <integer nclean>
```

17.3.8 Electrostatic potentials

A file in plt format of the electrostatic potential resulting from partial charges generated by the ESP module is generated by the command

```
esp  [ <integer spacing default 10> ] \
    [ <real rcut default 1.0> ] [periodic [<integer iper default 3>]] \
    [ <string xfile> [ <string pltfile> ] ]
```

The input coordinates are taken from the xyzq file that can be generated from a rst by the prepare module. Parameter spacing specifies the number of gridpoints per nm, rcut specifies extent of the charge grid beyond the molecule. Periodic boundaries will be used if periodic is specified. If iper is set to 2, periodic boundary conditions are applied in x and y dimensions only. If periodic is specified, a negative value of rcut will extend the grid in the periodic dimensions by abs(rcut), otherwise this value will be ignored in the periodic dimensions. The resulting plt formatted file pltfile can be viewed with the gOpenMol program. The resulting electrostatic potential grid is in units of $kJ mol^{-1} e^{-1}$. If no files are specified, only the parameters are set. This analysis applies to solute(s) only.

The electrostatic potential at specific point are evaluated using

```
esp_points [<string filpin> [<string filhol> [<string filpou> [<string filavg>]]]]
```

17.4 Format of MD files

17.4.1 Format of fragment file

Card	Format	Description
I-1-1	a1	$ or # comments to describe fragment
I-2-1	i5	number of atoms in the fragment
I-2-2	i5	number of parameter sets
I-2-3	i5	default parameter set
I-2-4	i5	number of z-matrix definition
For each parameter		
set one card I-3		
I-3-1	a	residue name for parameter set
For each atom one deck II		
II-1-1	i5	atom sequence number
II-1-2	a6	atom name
II-1-3	a5	atom type
II-1-4	a1	dynamics type
		blank : normal
		D : dummy atom
		S: solute interactions only
		Q : quantum atom
		other : intramolecular solute interactions only
II-1-5	i5	link number
		0: no link
		1: first atom in chain
		2: second atom in chain
		3 and up: other links
II-1-6	i5	environment type
		0: no special identifier
		1: planar, using improper torsion
		2: tetrahedral, using improper torsion
		3: tetrahedral, using improper torsion

Card	Format	Description
		4: atom in aromatic ring
II-1-7	i5	
II-1-8	i5	charge group
II-1-9	i5	polarization group
II-1-10	f12.6	atomic partial charge
II-1-11	f12.6	atomic polarizability
For each additional		
parameter set on card II-2		
II-2-1	11x,a6	atom type
II-2-2	a1	dynamics type
		blank : normal
		D : dummy atom
		S : solute interactions only
		Q : quantum atom
		other : intramolecular solute interactions only
II-2-7	25x,f12.6	atomic partial charge
II-2-8	f12.6	atomic polarizability
Any number of cards in deck III to		
specify complete connectivity		
III-1-1	16i5	connectivity, duplication allowed
One blank card to signal		
the end of the connectivity list		
For each z-matrix definition one card IV		
IV-1-1	i5	atom i
IV-1-2	i5	atom j
IV-1-3	i5	atom k
IV-1-4	i5	atom l
IV-1-5	f12.6	bond length i-j
IV-1-6	f12.6	angle i-j-k
IV-1-7	f12.6	torson i-j-k-l

17.4.2 Format of segment file (1 of 7)

Card	Format	Description
I-0-1		# lines at top are comments
I-1-1	a1	$ to identify the start of a segment
I-1-2	a10	name of the segment, the tenth character
		N: identifies beginning of a chain
		C: identifies end of a chain
		blank: identifies chain fragment
		M: identifies an integral molecule
I-2-1	f12.6	version number
I-3-1	i5	number of atoms in the segment
I-3-2	i5	number of bonds in the segment
I-3-3	i5	number of angles in the segment
I-3-4	i5	number of proper dihedrals in the segment
I-3-5	i5	number of improper dihedrals in the segment
I-3-6	i5	number of z-matrix definitions
I-3-7	i5	number of parameter sets
1-3-8	i5	default parameter set
For each parameter set one card I-4		
I-4-1	f12.6	dipole correction energy

17.4.3 Format of segment file (2 of 7)

Card	Format	Description
For each atom one deck II		
II-1-1	i5	atom sequence number
II-1-2	a6	atom name
II-1-3	i5	link number
II-1-4	i5	environment type
		0: no special identifier
		1: planar, using improper torsion
		2: tetrahedral, using improper torsion
		3: tetrahedral, using improper torsion
		4: atom in aromatic ring
II-1-5	i5	
II-1-6	i5	charge group
II-1-7	i5	polarization group
For each parameter set one card II-2		
II-2-1	5x,a5	atom type
II-2-2	a1	dynamics type
		blank : normal
		D : dummy atom
		S : solute interactions only
		Q : quantum atom
		other : intramolecular solute interactions only
II-2-3	f12.6	atomic partial charge in e
II-2-4	f12.6	atomic polarizability ($4 \pi \varepsilon_o$ in nm^3)

17.4.4 Format of segment file (3 of 7)

Card	Format	Description
For each bond a deck III		
III-1-1	i5	bond sequence number
III-1-2	i5	bond atom i
III-1-3	i5	bond atom j
III-1-4	i5	bond type
		0: harmonic
		1: constrained bond
III-1-5	i5	bond parameter origin
		0: from database, next card ignored
		1: from next card
For each parameter set one card III-2		
III-2-1	f12.6	bond length in nm
III-2-2	e12.5	bond force constant in (kJ nm^{2} mol^{-1})

17.4.5 Format of segment file (4 of 7)

Card	Format	Description
For each angle a deck IV		
IV-1-1	i5	angle sequence number
IV-1-2	i5	angle atom i
IV-1-3	i5	angle atom j
IV-1-4	i5	angle atom k
IV-1-5	i5	angle type
		0: harmonic
IV-1-6	i5	angle parameter origin
		0: from database, next card ignored
		1: from next card
For each parameter set one card IV-2		
IV-2-1	f10.6	angle in radians
IV-2-2	e12.5	angle force constant in (kJ mol $^{-1}$)

17.4.6 Format of segment file (5 of 7)

Card	Format	Description
For each proper dihedral a deck V		
V-1-1	i5	proper dihedral sequence number
V-1-2	i5	proper dihedral atom i
V-1-3	i5	proper dihedral atom j
V-1-4	i5	proper dihedral atom k
V-1-5	i5	proper dihedral atom l
V-1-6	i5	proper dihedral type
		0: (*C*cos(mφ-δ)
V-1-7	i5	proper dihedral parameter origin
		0: from database, next card ignored
		1: from next card
For each parameter set one card V-2		
V-2-1	i3	multiplicity
V-2-2	f10.6	proper dihedral in radians
V-2-3	e12.5	proper dihedral force constant in (kJ mol $^{-1}$)

17.4.7 Format of segment file (6 of 7)

Card	Format	Description
For each improper dihedral a deck VI		
VI-1-1	i5	improper dihedral sequence number
VI-1-2	i5	improper dihedral atom i
VI-1-3	i5	improper dihedral atom j
VI-1-4	i5	improper dihedral atom k
VI-1-5	i5	improper dihedral atom l
VI-1-6	i5	improper dihedral type
		0: harmonic
VI-1-7	i5	improper dihedral parameter origin
		0: from database, next card ignored
		1: from next card
For each parameter set one card VI-2		
VI-2-1	3x,f10.6	improper dihedral in radians
VI-2-2	e12.5	improper dihedral force constant in (kJ mol^{-1})

17.4.8 Format of segment file (7 of 7)

Card	Format	Description
For each z-matrix definition one card VII		
VII-1-1	i5	atom i
VII-1-2	i5	atom j
VII-1-3	i5	atom k
VII-1-4	i5	atom l
VII-1-5	f12.6	bond length i-j
VII-1-6	f12.6	angle i-j-k
VII-1-7	f12.6	torson i-j-k-l

17.4.9 Format of sequence file

Card	Format	Description
I-1-1	a1	$ to identify the start of a sequence
I-1-2	a10	name of the sequence
Any number of cards 1 and 2 in deck II		
to specify the system		
II-1-1	i5	segment number
II-1-2	a10	segment name, last character will be determined from chain
II-2-1	a	break to identify a break in the molecule chain
II-2-1	a	molecule to identify the end of a solute molecule
II-2-1	a	fraction to identify the end of a solute fraction
II-2-1	a5	link to specify a link
II-2-2	i5	segment number of first link atom
II-2-3	a4	name of first link atom
II-2-4	i5	segment number of second link atom
II-2-5	a4	name of second link atom
II-2-1	a	solvent to identify solvent definition on next card
II-2-1	a	stop to identify the end of the sequence
II-2-1	a6	repeat to repeat next ncard cards ncount times
II-2-2	i5	number of cards to repeat (ncards)
II-2-3	i5	number of times to repeat cards (ncount)
Any number of cards in deck II		
to specify the system		

17.4.10 Format of trajectory file

Card	Format	Description
I-1-1	a6	keyword header
I-2-1	i10	number of atoms per solvent molecule
I-2-2	i10	number of solute atoms
I-2-3	i10	number of solute bonds
I-2-4	i10	number of solvent bonds

Card	Format	Description
I-2-5	i10	number of solvent molecules
I-2-6	i10	precision of the coordinates. 0=standard, 1=high
For each atom per solvent molecule one card I-3		
I-3-1	a5	solvent name
I-3-2	a5	atom name
For each solute atom one card I-4		
I-4-1	a5	segment name
I-4-2	a5	atom name
I-4-3	i6	segment number
I-4-4	i10	solute atom counter
I-4-5	i5	integer 1
For each solvent bond one card I-5		
I-5-1	i8	atom index i for bond between i and j
I-5-2	i8	atom index j for bond between i and j
For each solute bond one card I-6		
I-6-1	i8	atom index i for bond between i and j
I-6-2	i8	atom index j for bond between i and j
For each frame one deck II		
II-1-1	a5	keyword frame
II-2-1	f12.6	time of frame in ps
II-2-2	12.6	temperature of frame in K
II-2-3	e12.5	pressure of frame in Pa
II-2-4	a10	date
II-2-5	a10	time
II-3-1	f12.6	box dimension x
II-3-2	12x,f12.6	box dimension y
II-3-3	24x,f12.6	box dimension z
II-4-1	l1	logical lxw for solvent coordinates
II-4-2	l1	logical lvw for solvent velocities
II-4-3	l1	logical lfw for solvent forces

Card	Format	Description
II-4-4	l1	
II-4-5	l1	logical lxs for solute coordinates
II-4-6	l1	logical lvs for solute velocities
II-4-7	l1	logical lfs for solute forces
II-4-8	l1	logical lps for solute induced dipoles
II-4-5	i10	number of solvent molecules
II-4-6	i10	number of solvent atoms
II-4-7	i10	number of solute atoms
For each solvent molecule one card II-5		
for each atom, if standard precision		
II-5-1	3f8.3	solvent atom coordinates, if lxw or lvw
II-5-4	3f8.3	solvent atom velocities, if lvw
II-5-7	3f8.1	solvent atom forces, if lfw
For each solute atom one card II-6		
for each atom, if standard precision		
II-6-1	3f8.3	solute atom coordinates, if lxs or lvs
II-6-4	3f8.3	solute atom velocities, if lvs
II-6-7	3f8.1	solute atom forces, if lfs
For each solvent molecule one card II-5		
for each atom, if high precision		
II-5-1	3e12.6	solvent atom coordinates, if lxw or lvw
II-5-4	3e12.6	solvent atom velocities, if lvw
II-5-7	3e12.6	solvent atom forces, if lfw (on new card if both lxw and lvw)
For each solute atom one card II-6		
for each atom, if high precision		
II-6-1	3e12.6	solute atom coordinates, if lxs or lvs
II-6-4	3e12.6	solute atom velocities, if lvs
II-6-7	3e12.6	solute atom forces, if lfs (on new card if both lxs and lvs)

17.4.11 Format of free energy file

Card	Format	Description
For each step in λ one deck I		
I-1-1	i7	number nderiv of data summed in derivative decomposition array deriv
I-1-2	i7	length ndata of total derivative array drf
I-1-3	f12.6	current value of λ
I-1-4	f12.6	step size of λ
I-2-1	4e12.12	derivative decomposition array deriv(1:24)
I-3-1	4e12.12	total derivative array dfr(1:nda)
I-4-1	i10	size of ensemble at current λ
I-4-2	e20.12	average temperature at current λ
I-4-3	e20.12	average exponent reverse perturbation energy at current λ
I-4-4	e20.12	average exponent forward perturbation energy at current λ

17.4.12 Format of root mean square deviation file

Card	Format	Description
For each analyzed time step one card I-1		
I-1-1	f12.6	time in ps
I-1-2	f12.6	total rms deviation of the selected atoms before superimposition
I-1-3	f12.6	total rms deviation of the selected atoms after superimposition
II-1-1	a8	keyword analysis
For each solute atom one card II-2		
II-2-1	a5	segment name
II-2-2	a5	atom name
II-2-3	i6	segment number
II-2-4	i10	atom number
II-2-5	i5	selected if 1
II-2-6	f12.6	average atom rms deviation after superimposition
III-1-1	a8	keyword analysis
For each solute segment one card III-2		
III-2-1	a5	segment name
III-2-2	i6	segment number
III-2-3	f12.6	average segment rms deviation after superimposition

17.4.13 Format of property file

Card	Format	Description
1	i7	number nprop of recorded properties
I-1-2	1x,2a10	date and time
For each of the nprop properties one card I-2		
I-2-1	a50	description of recorded property
For each recorded step one deck II		
II-1-1	4(1pe12.5)	value of property

18 Controlling NWChem with Python

18.1 Overview

Python programs may be embedded into the NWChem input and used to control the execution of NWChem. Python is a very powerful and widely used scripting language that provides useful things such as variables, conditional branches and loops, and is also readily extended. Example applications include scanning potential energy surfaces, computing properties in a variety of basis sets, optimizing the energy w.r.t. parameters in the basis set, computing polarizabilities with finite field, and simple molecular dynamics.

Look in the NWChem contrib directory for useful scripts and examples. Visit the Python web-site https://www.python.org for a full manual and lots of useful code and resources.

18.2 How to input and run a Python program inside NWChem

A Python program is input into NWChem inside a Python compound directive.

```
python [print|noprint]
  ...
end
```

The END directive must be flush against the left margin (see the Troubleshooting section for the reason why).

The program is by default printed to standard output when read, but this may be disabled with the noprint keyword. Python uses indentation to indicate scope (and the initial level of indentation must be zero), whereas NWChem uses optional indentation only to make the input more readable. For example, in Python, the contents of a loop, or conditionally-executed block of code must be indented further than the surrounding code. Also, Python attaches special meaning to several symbols also used by NWChem. For these reasons, the input inside a PYTHON compound directive is read verbatim except that if the first line of the Python program is indented, the same amount of indentation is removed from all subsequent lines. This is so that a program may be indented inside the PYTHON input block for improved readability of the NWChem input, while satisfying the constraint that when given to Python the first line has zero indentation.

E.g., the following two sets of input specify the same Python program.

```
python
  print ("Hello")
  print ("Goodbye")
end

python
print ("Hello")
print ("Goodbye")
end
```

whereas this program is in error since the indentation of the second line is less than that of the first.

```
python
  print ("Hello")
print ("Goodbye")
end
```

The Python program is not executed until the following directive is encountered

```
task python
```

which is to maintain consistency with the behavior of NWChem in general. The program is executed by all nodes. This enables the full functionality and speed of NWChem to be accessible from Python, but there are some gotchas

- Print statements and other output will be executed by all nodes so you will get a lot more output than probably desired unless the output is restricted to just one node (by convention node zero).
- The calls to NWChem functions are all collective (i.e., all nodes must execute them). If these calls are not made collectively your program may deadlock (i.e., cease to make progress).
- When writing to the database (`rtdb_put()`) it is the data from node zero that is written.

- NWChem overrides certain default signal handlers so care must be taken when creating processes (see Section describing a scanning example).

18.3 NWChem extensions

Since we have little experience using Python, the NWChem-Python interface might change in a non-backwardly compatible fashion as we discover better ways of providing useful functionality. We would appreciate suggestions about useful things that can be added to the NWChem-Python interface. In principle, nearly any Fortran or C routine within NWChem can be extended to Python, but we are also interested in ideas that will enable users to build completely new things. For instance, how about being able to define your own energy functions that can be used with the existing optimizers or dynamics package?

Python has been extended with a module named “nwchem” which is automatically imported and contains the following NWChem-specific commands. They all handle NWChem-related errors by raising the exception “NWChemError”, which may be handled in the standard Python manner (see Section describing handling exception).

- `input_parse(string)` – invokes the standard NWChem input parser with the data in string as input. Note that the usual behavior of NWChem will apply – the parser only reads input up to either end of input or until a `TASK` directive is encountered (the task directive is not executed by the parser).
- `task_energy(theory)` – returns the energy as if computed with the NWChem directive `TASK ENERGY <THEORY>` .
- `task_gradient(theory)` – returns a tuple (energy,gradient) as if computed with the NWChem directive `TASK GRADIENT <THEORY>` .
- `task_optimize(theory)` – returns a tuple (energy,gradient) as if computed with the NWChem directive `TASK OPTIMIZE <THEORY>` . The energy and gradient will be those at the last point in the optimization and consistent with the current geometry in the database.
- `ga_nodeid()` – returns the number of the parallel process.
- `rtdb_print(print_values)` – prints the contents of the RTDB. If `print_values` is 0, only the keys are printed, if it is 1 then the values are also printed.
- `rtdb_put(name, values)` or `rtdb_put(name, values, type)` – puts the values into the database with the given name. In the first form, the type is inferred from the first value, and in the second form the type is specified using the last argument as one of INT, DBL, LOGICAL, or CHAR.
- `rtdb_get(name)` – returns the data from the database associated with the given name.

An example below explains, in lieu of a Python wrapper for the geometry object, how to obtain the Cartesian molecular coordinates directly from the database.

18.4 Examples

Several examples will provide the best explanation of how the extensions are used, and how Python might prove useful.

18.4.1 Hello world

```
python
  print ('Hello world from process %i'  % ga_nodeid())
end

task python
```

This input prints the traditional greeting from each parallel process.

18.4.2 Scanning a basis exponent

```
geometry units au
 O 0 0 0; H 0 1.430 -1.107; H 0 -1.430 -1.107
end

python
exponent = 0.1
while (exponent <= 2.01):
   input_parse('''
   basis noprint
    H library 3-21g; O library 3-21g; O d; %f 1.0
   end
   ''' % (exponent))
   print (' exponent = %5.4f,' % exponent, ', energy = %16.10f' % task_energy('scf'))
   exponent = exponent + 0.1
end

print none

task python
```

This program augments a 3-21g basis for water with a d-function on oxygen and varies the exponent from 0.1 to 2.0 in steps of 0.1, printing the exponent and energy at each step.

The geometry is input as usual, but the basis set input is embedded inside a call to `input_parse()` in the Python program. The standard Python string substitution is used to put the current value of the exponent into the basis set (replacing the %f) before being parsed by NWChem. The energy is returned by `task_energy('scf')` and printed out. The print none in the NWChem input switches off all NWChem output so all you will see is the output from your Python program.

Note that execution in parallel may produce unwanted output since all process execute the print statement inside the Python program.

Look in the NWChem contrib directory for a routine that makes the above task easier.

18.4.3 Scanning a basis exponent revisited

```
geometry units au
O 0 0 0; H 0 1.430 -1.107; H 0 -1.430 -1.107
end


python
  if (ga_nodeid() == 0): plotdata = open("plotdata",'w')

  def energy_at_exponent(exponent):
    input_parse('''
    basis noprint
      H library 3-21g; O library 3-21g; O d; %f 1.0
    end
    ''' % (exponent))
    return task_energy('scf')

  exponent = 0.1
  while exponent <= 2.01:
    energy = energy_at_exponent(exponent)
    if (ga_nodeid() == 0):
       print (' exponent = %5.4f,' % exponent, ', energy = %16.10f' % energy)
       plotdata.write('%f %f\n' % (exponent , energy))
    exponent = exponent + 0.1

  if (ga_nodeid() == 0): plotdata.close()
end

print none
task python
```

This input performs exactly the same calculation as the previous one, but uses a slightly more sophisticated Python program, also writes the data out to a file for easy visualization with a package such as gnuplot, and protects write statements to prevent duplicate output in a parallel job. The only significant differences are in the Python program. A file called “plotdata” is opened, and then a procedure is defined which given an exponent returns the energy. Next comes the main loop that scans the exponent through the desired range and prints the results to standard output and to the file. When the loop is finished the additional output file is closed.

18.4.4 Scanning a geometric variable

```
python
  geometry = '''
    geometry noprint; symmetry d2h
      C 0 0 %f; H 0  0.916 1.224
    end
  '''
  x = 0.6
  while (x < 0.721):
    input_parse(geometry % x)
    energy = task_energy('scf')
    print (' x = %5.2f   energy = %10.6f' % (x, energy))
    x = x + 0.01
end

basis; C library 6-31g*; H library 6-31g*; end

print none

task python
```

This scans the bond length in ethene from 1.2 to 1.44 in steps of 0.2 computing the energy at each geometry. Since it is using D_{2h} symmetry the program actually uses a variable (x) that is half the bond length.

Look in the NWChem contrib directory for a routine that makes the above task easier.

18.4.5 Scan using the BSSE counterpoise corrected energy

```
basis spherical
  Ne library cc-pvdz; BqNe library Ne cc-pvdz
  He library cc-pvdz; BqHe library He cc-pvdz
end

mp2; tight; freeze core atomic; end

print none

python
  supermolecule = 'geometry noprint;   Ne 0 0 0;   He 0 0 %f; end\n'
  fragment1     = 'geometry noprint;   Ne 0 0 0; BqHe 0 0 %f; end\n'
  fragment2     = 'geometry noprint; BqNe 0 0 0;   He 0 0 %f; end\n'

  def energy(geometry):
    input_parse(geometry + 'scf; vectors atomic; end\n')
    return task_energy('mp2')

  def bsse_energy(z):
    return energy(supermolecule % z) - \
        energy(fragment1 % z) - \
        energy(fragment2 % z)
  z = 3.3
  while (z < 4.301):
    e = bsse_energy(z)
    if (ga_nodeid() == 0): print (' z = %5.2f   energy = %10.7f ' % (z, e))
    z = z + 0.1
end

task python
```

This example scans the He–Ne bond-length from 3.3 to 4.3 and prints out the BSSE counterpoise corrected MP2 energy.

The basis set is specified as usual, noting that we will need functions on ghost centers to do the counterpoise correction. The Python program commences by defining strings containing the geometry of the super-molecule and two fragments, each having one variable to be substituted. Next, a function is defined to compute the energy given a geometry, and then a function is defined to compute the counterpoise corrected energy at a given bond length. Finally, the bond length is scanned and the energy printed. When computing the energy, the atomic guess has to be forced in the SCF since by default it will attempt to use orbitals from the previous calculation which is not appropriate here.

Since the counterpoise corrected energy is a linear combination of other standard energies, it is possible to compute the analytic derivatives term by term. Thus, combining this example and the next could yield the foundation of a BSSE corrected geometry optimization package.

18.4.6 Scan the geometry and compute the energy and gradient

```
basis noprint; H library sto-3g; O library sto-3g; end

python
 print ('   y     z     energy               gradient')
 print (' ----- ----- ---------- ----------------------------------')
 y = 1.2
 while y <= 1.61:
    z = 1.0
    while z <= 1.21:
       input_parse('''
         geometry noprint units atomic
           O 0   0   0
           H 0  %f -%f
           H 0 -%f -%f
         end
       ''' % (y, z, y, z))

       (energy,gradient) = task_gradient('scf')

       print (' %5.2f %5.2f %9.6f' % (y, z, energy)),
       i = 0
       while (i < len(gradient)):
          print ('%5.2f' % gradient[i]),
          i = i + 1
       print ('')
       z = z + 0.1
    y = y + 0.1
end

print none

task python
```

This program illustrates evaluating the energy and gradient by calling task_gradient(). A water molecule is scanned through several C_{2v} geometries by varying the y and z coordinates of the two hydrogen atoms. At each geometry the coordinates, energy and gradient are printed.

The basis set (sto-3g) is input as usual. The two while loops vary the y and z coordinates. These are then substituted into a geometry which is parsed by NWChem using `input_parse()`. The energy and gradient are then evaluated by calling `task_gradient()` which returns a tuple containing the energy (a scalar) and the gradient (a vector or list). These are printed out exploiting the Python convention that a print statement ending in a comma does not print end-of-line.

18.4.7 Reaction energies varying the basis set

```
mp2; freeze atomic; end

print none

python
  energies = {}
  c2h4 = 'geometry noprint; symmetry d2h; \
         C 0 0 0.672; H 0 0.935 1.238; end\n'
  ch4  = 'geometry noprint; symmetry td; \
         C 0 0 0; H 0.634 0.634 0.634; end\n'
  h2   = 'geometry noprint; H 0 0 0.378; H 0 0 -0.378; end\n'

  def energy(basis, geometry):
    input_parse('''
      basis spherical noprint
        c library %s ; h library %s
      end
    ''' % (basis, basis))
    input_parse(geometry)
    return task_energy('mp2')

  for basis in ('sto-3g', '6-31g', '6-31g*', 'cc-pvdz', 'cc-pvtz'):
     energies[basis] =   2*energy(basis, ch4) \
                       - 2*energy(basis, h2) - energy(basis, c2h4)
     if (ga_nodeid() == 0): print (basis, ' %8.6f' % energies[basis])
end

task python
```

In this example the reaction energy for $2H_2 + C_2H_4 \rightarrow 2CH_4$ is evaluated using MP2 in several basis sets. The geometries

are fixed, but could be re-optimized in each basis. To illustrate the useful associative arrays in Python, the reaction energies are put into the associative array energies – note its declaration at the top of the program.

18.4.8 Using the database

```
python
  rtdb_put("test_int2", 22)
  rtdb_put("test_int", [22, 10, 3],    INT)
  rtdb_put("test_dbl", [22.9, 12.4, 23.908],  DBL)
  rtdb_put("test_str", "hello", CHAR)
  rtdb_put("test_logic", [0,1,0,1,0,1], LOGICAL)
  rtdb_put("test_logic2", 0, LOGICAL)

  rtdb_print(1)

  print "test_str    = ", rtdb_get("test_str")
  print "test_int    = ", rtdb_get("test_int")
  print "test_in2    = ", rtdb_get("test_int2")
  print "test_dbl    = ", rtdb_get("test_dbl")
  print "test_logic  = ", rtdb_get("test_logic")
  print "test_logic2 = ", rtdb_get("test_logic2")
end

task python
```

This example illustrates how to access the database from Python.

18.4.9 Handling exceptions from NWChem

```
geometry; he 0 0 0; he 0 0 2; end
basis; he library 3-21g; end
scf; maxiter 1; end

python
  try:
    task_energy('scf')
  except NWChemError, message:
    print 'Error from NWChem ... ', message
end

task python
```

The above test program shows how to handle exceptions generated by NWChem by forcing an SCF calculation on $He_{2<\sub>}$ to fail due to insufficient iterations.

If an NWChem command fails it will raise the exception “NWChemError” (case sensitive) unless the error was fatal. If the exception is not caught, then it will cause the entire Python program to terminate with an error. This Python program catches the exception, prints out the message, and then continues as if all was well since the exception has been handled.

If your Python program detects an error, raise an unhandled exception. Do not call exit(1) since this may circumvent necessary clean-up of the NWChem execution environment.

18.4.10 Accessing geometry information: a temporary hack

In an ideal world the geometry and basis set objects would have full Python wrappers, but until then a back-door solution will have to suffice. We’ve already seen how to use `input_parse()` to put geometry (and basis) data into NWChem, so it only remains to get the geometry data back after it has been updated by a geometry optimzation or some other operation.

The following Python procedure retrieves the coordinates in the same units as initially input for a geometry of a given name. Its full source is included in the NWChem contrib directory.

```
def geom_get_coords(name):
  try:
    actualname = rtdb_get(name)
  except NWChemError:
    actualname = name
  coords = rtdb_get('geometry:' + actualname + ':coords')
  units  = rtdb_get('geometry:' + actualname + ':user units')
  if (units == 'a.u.'):
    factor = 1.0
  elif (units == 'angstroms'):
    factor = rtdb_get('geometry:'+actualname+':angstrom_to_au')
  else:
    raise NWChemError,'unknown units'
  i = 0
  while (i < len(coords)):
    coords[i] = coords[i] / factor
    i = i + 1
  return coords
```

A geometry with name NAME has its coordinates (in atomic units) stored in the database entry`geometry:NAME:coords` . A minor wrinkle here is that indirection is possible (and used by the optimizers) so that we must first check if NAME actually points to another name. In the program this is done in the first try…except sequence. With the actual name of the geometry, we can get the coordinates. Any exceptions are passed up to the caller. The rest of the code is just to convert back into the initial input units – only atomic units or Angstrøms are handled in this simple example. Returned is a list of the atomic coordinates in the same units as your initial input.

The routine is used as follows

```
coords = geom_get_coords('geometry')
```

or, if you want better error handling

```
try:
  coords = geom_get_coords('geometry')
except NWChemError,message:
  print 'Coordinates for geometry not found ', message
else:
  print coords
```

This is very dirty and definitely not supported from one release to another, but, browsing the output of`rtdb_print()` at the end of a calculation is a good way to find stuff. To be on safer ground, look in the programmers manual since some of the high-level routines do pass data via the database in a well-defined and supported manner. Be warned – you must be very careful if you try to modify data in the database. The input parser does many important things that are not immediately apparent (e.g., ensure the geometry is consistent with the point group, mark the SCF as not converged if the SCF options are changed, …). Where at all possible your Python program should generate standard NWChem input and pass it to `input_parse()` rather than setting parameters directly in the database.

18.4.11 Scaning a basis exponent yet again: plotting and handling child processes

```
geometry units au
  O 0 0 0; H 0 1.430 -1.107; H 0 -1.430 -1.107
end

print none

python
  import Gnuplot, time, signal

  def energy_at_exponent(exponent):
    input_parse('''
      basis noprint
        H library 3-21g; O library 3-21g; O d; %f 1.0
      end
    ''' % (exponent))
    return task_energy('scf')

  data = []
  exponent = 0.5
  while exponent <= 0.6:
    energy = energy_at_exponent(exponent)
    print ' exponent = ', exponent, ' energy = ', energy
    data = data + [exponent,energy](exponent,energy.md)
    exponent = exponent + 0.02

  if (ga_nodeid() == 0):
    signal.signal(signal.SIGCHLD, signal.SIG_DFL)
    g = Gnuplot.Gnuplot()
    g('set data style linespoints')
    g.plot(data)
    time.sleep(30)  # 30s to look at the plot

end

task python
```

This illustrates how to handle signals from terminating child processes and how to generate simple plots on UNIX systems. The scanning example is modified so that instead of writing the data to a file for subsequent visualization, it is saved for subsequent visualization with Gnuplot (you'll need both Gnuplot and the corresponding package for Python in your PYTHONPATH. Look at https://web.archive.org/web/20010717060625/http://monsoon.harvard.edu/~mhagger/download/).

The issue is that NWChem traps various signals from the O/S that usually indicate bad news in order to provide better error handling and reliable clean-up of shared, parallel resources. One of these signals is SIGCHLD which is generated whenever a child process terminates. If you want to create child processes within Python, then the NWChem handler for SIGCHLD must be replaced with the default handler. There seems to be no easy way to restore the NWChem handler after the child has completed, but this should have no serious side effect.

18.5 Troubleshooting

Common problems with Python programs inside NWChem.

1. You get the message

   ```
   0:python_input: indentation must be >= that of first line: 4
   ```

   This indicates that NWChem thinks that a line is less indented than the first line. If this is not the case then perhaps there is a tab in your input which NWChem treats as a single space character but appears to you as more spaces. Try running untabify in Emacs. The problem could also be the END directive that terminates the PYTHON compound directive, since Python also has an end statement. To avoid confusion the END directive for NWChem must be at the start of the line.
2. Your program hangs or deadlocks – most likely you have a piece of code that is restricted to executing on a subset of the processors (perhaps just node 0) but is calling (perhaps indirectly) a function that must execute on all nodes.

19 NWChem Examples

19.1 Sample input files

19.1.1 Water SCF calculation and geometry optimization in a 6-31g basis

The Getting Started input file performs a geometry optimization in a single task. A single point SCF energy calculation is performed and then restarted to perform the optimization (both could of course be performed in a single task).

19.1.1.1 Job 1. Single point SCF energy

```
start h2o
title "Water in 6-31g basis set"

geometry units au
  O     0.00000000    0.00000000    0.00000000
  H     0.00000000    1.43042809   -1.10715266
  H     0.00000000   -1.43042809   -1.10715266
end
basis
  H library 6-31g
  O library 6-31g
end
task scf
```

The final energy should be -75.983998.

19.1.1.2 Job 2. Restarting and perform a geometry optimization

```
restart h2o
title "Water geometry optimization"

task scf optimize
```

There is no need to specify anything that has not changed from the previous input deck, though it will do no harm to repeat it.

19.1.2 Compute the polarizability of Ne using finite field

19.1.2.1 Job 1. Compute the atomic energy

```
start ne
title "Neon"
geometry; ne 0 0 0; end
basis spherical
  ne library aug-cc-pvdz
end
scf; thresh 1e-10; end
task scf
```

The final energy should be -128.496350.

19.1.2.2 Job 2. Compute the energy with applied field

An external field may be simulated with point charges. The charges here apply a field of magnitude 0.01 atomic units to the atom at the origin. Since the basis functions have not been reordered by the additional centers we can also restart from the previous vectors, which is the default for a restart job.

```
restart ne
title "Neon in electric field"
geometry units atomic
  bq1 0 0 100 charge 50
  ne  0 0 0
  bq2 0 0 -100 charge -50
end
task scf
```

The final energy should be -128.496441, which together with the previous field-free result yields an estimate for the

polarizability of 1.83 atomic units. Note that by default NWChem does not include the interaction between the two point charges in the total energy.

19.1.3 SCF energy of H_2CO using ECPs for C and O

The following will compute the SCF energy for formaldehyde with ECPs on the Carbon and Oxygen centers.

```
title "formaldehyde ECP deck"

start ecpchho

geometry units au
  C        0.000000  0.000000 -1.025176
  O        0.000000  0.000000  1.280289
  H        0.000000  1.767475 -2.045628
  H        0.000000 -1.767475 -2.045628
end

basis
  C  SP
   0.1675097360D+02 -0.7812840500D-01  0.3088908800D-01
   0.2888377460D+01 -0.3741108860D+00  0.2645728130D+00
   0.6904575040D+00  0.1229059640D+01  0.8225024920D+00
  C  SP
   0.1813976910D+00  0.1000000000D+01  0.1000000000D+01
  C  D
   0.8000000000D+00  0.1000000000D+01
  C  F
   0.1000000000D+01  0.1000000000D+01
  O  SP
   0.1842936330D+02 -0.1218775590D+00  0.5975796600D-01
   0.4047420810D+01 -0.1962142380D+00  0.3267825930D+00
   0.1093836980D+01  0.1156987900D+01  0.7484058930D+00
  O  SP
   0.2906290230D+00  0.1000000000D+01  0.1000000000D+01
  O  D
   0.8000000000D+00  0.1000000000D+01
  O  F
   0.1100000000D+01  0.1000000000D+01
  H  S
   0.1873113696D+02  0.3349460434D-01
   0.2825394365D+01  0.2347269535D+00
   0.6401216923D+00  0.8137573262D+00
  H  S
   0.1612777588D+00  0.1000000000D+01
end

ecp
  C nelec 2
  C ul
       1      80.0000000      -1.60000000
       1      30.0000000      -0.40000000
       2       0.5498205      -0.03990210
  C s
       0       0.7374760       0.63810832
       0     135.2354832      11.00916230
       2       8.5605569      20.13797020
  C p
       2      10.6863587      -3.24684280
       2      23.4979897       0.78505765
  O nelec 2
  O ul
       1      80.0000000      -1.60000000
       1      30.0000000      -0.40000000
       2       1.0953760      -0.06623814
  O s
       0       0.9212952       0.39552179
       0      28.6481971       2.51654843
       2       9.3033500      17.04478500
  O p
       2      52.3427019      27.97790770
       2      30.7220233     -16.49630500
end

scf
  vectors input hcore
  maxiter 20
end

task scf
```

This should produce the following output:

```
      Final RHF  results
      ------------------

        Total SCF energy =    -22.507927218024
      One electron energy =   -71.508730162974
     Two electron energy =     31.201960019808
Nuclear repulsion energy =     17.798842925142
```

19.1.4 MP2 optimization and CCSD(T) on nitrogen

The following performs an MP2 geometry optimization followed by a CCSD(T) energy evaluation at the converged geometry. A Dunning correlation-consistent triple-zeta basis is used. The default of Cartesian basis functions must be overridden using the keyword spherical on the BASIS directive. The 1s core orbitals are frozen in both the MP2 and coupled-cluster calculations (note that these must separately specified). The final MP2 energy is -109.383276, and the CCSD(T) energy is -109.399662.

```
start n2

geometry
  symmetry d2h
  n 0 0 0.542
end

basis spherical
  n library cc-pvtz
end

mp2
  freeze core
end

task mp2 optimize

ccsd
  freeze core
end

task ccsd(t)
```

19.2 Examples of geometries using symmetry

Below are examples of the use of theSYMMETRY directive in the compound GEOMETRY directive. The *z* axis is always the primary rotation axis. When in doubt about which axes and planes are used for the group elements, the keyword print may be added to the SYMMETRY directive to obtain this information.

19.2.1 C_s methanol

The σ_h plane is the *xy* plane.

```
geometry units angstroms
  C    0.11931097   -0.66334875    0.00000000
  H    1.20599017   -0.87824237    0.00000000
  H   -0.32267592   -1.15740001    0.89812652
  O   -0.01716588    0.78143468    0.00000000
  H   -1.04379735    0.88169812    0.00000000
  symmetry cs
end
```

19.2.2 C_{2v} water

The z axis is the C_2 axis and the σ_v may be either the *xz* or the *yz* planes.

```
geometry units au
  O     0.00000000    0.00000000    0.00000000
  H     0.00000000    1.43042809   -1.10715266
  symmetry group c2v
end
```

19.2.3 D_{2h} acetylene

Although acetylene has symmetry $D_{\infty h}$ the subgroup D_{2h} includes all operations that interchange equivalent atoms which is what determines how much speedup you gain from using symmetry in building a Fock matrix.

The C_2 axes are the x, y, and z axes. The σ planes are the xy, xz and yz planes. Generally, the unique atoms are placed to use the z as the primary rotational axis and use the xz or yz planes as the σ plane.

```
geometry units au
  symmetry group d2h
  C     0.000000000    0.000000000   -1.115108538
  H     0.000000000    0.000000000   -3.106737425
end
```

19.2.4 D_{2h} ethylene

The C_2 axes are the x, y, and z axes. The σ planes are the xy, xz and yz planes. Generally, the unique atoms are placed to use the z as the primary rotational axis and use the xz or yz planes as the σ plane.

```
geometry units angstroms
  C 0 0 0.659250
  H 0 0.916366 1.224352
  symmetry d2h
end
```

19.2.5 T_d methane

For ease of use, the primary C_3 axis should be the x=y=z axis. The $3C_2$ axes are the x, y, and z.

```
geometry units au
  c   0.0000000     0.0000000     0.0000000
  h   1.1828637     1.1828637     1.1828637
  symmetry group Td
end
```

19.2.6 I_h buckminsterfullerene

One of the C_5 axes is the z axis and the point of inversion is the origin.

```
geometry units angstroms # Bonds = 1.4445, 1.3945
  symmetry group Ih
  c   -1.2287651   0.0   3.3143121
end
```

19.2.7 S_4 porphyrin

The S_4 and C_2 rotation axis is the z axis. The reflection plane for the S_4 operation is the xy plane.

```
geometry units angstroms
  symmetry group s4

  fe          0.000  0.000  0.000
  h           2.242  6.496 -3.320
  h           1.542  4.304 -2.811
  c           1.947  6.284 -2.433
  c           1.568  4.987 -2.084
  h           2.252  8.213 -1.695
  c           1.993  7.278 -1.458
  h           5.474 -1.041 -1.143
  c           1.234  4.676 -0.765
  h           7.738 -1.714 -0.606
  c           0.857  3.276 -0.417
  h           1.380 -4.889 -0.413
  c           1.875  2.341 -0.234
  h           3.629  3.659 -0.234
  c           0.493 -2.964 -0.229
  c           1.551 -3.933 -0.221
  c           5.678 -1.273 -0.198
  c           1.656  6.974 -0.144
  c           3.261  2.696 -0.100
  n           1.702  0.990 -0.035
end
```

19.2.8 D_{3h} iron penta-carbonyl

The C_3 axis is the z axis. The σ_h plane is the xy plane. One of the perpendicular C_2 axes is the x=y axis. One of the σ_v planes is the plane containing the x=y axis and the z axis. (The other axes and planes are generated by the C_3 operation.)

```
geometry units au
  symmetry group d3h

  fe      0.0        0.0        0.0

  c       0.0        0.0        3.414358
  o       0.0        0.0        5.591323

  c       2.4417087  2.4417087  0.0
  o       3.9810552  3.9810552  0.0
end
```

19.2.9 D_{3d} sodium crown ether

The C_3 axis is the z axis. The point of inversion is the origin. One of the perpendicular C_2 axes is the x=y axis. One of the σ_d planes is the plane containing the -x=y axis and the z axis.

Note that the oxygen atom is rotated in the x-y plane 15 degrees away from the y-axis so that it lies in a mirror plane. There is a total of six atoms generated from the unique oxygen, in contrast to twelve from each of the carbon and hydrogen atoms.

```
geometry units au
  symmetry D3d

 NA    .0000000000   .0000000000   .0000000000
 O    1.3384771885  4.9952647969  .1544089284
 H    6.7342048019 -0.6723850379 2.6581562148
 C    6.7599180056 -0.4844977035  .6136583870
 H    8.6497577017  0.0709194071  .0345361934

end
```

19.2.10 C_{3v} ammonia

The C_3 axis is the z axis. One of the σ_v planes is the plane containing the x=y axis and the z axis.

```
geometry units angstroms
  N 0     0     -0.055
  H 0.665 0.665 -0.481
  symmetry c3v
end
```

19.2.11 D_{6h} benzene

The C_6 axis is the z axis. The point of inversion is the origin. One of the 6 perpendicular C_2' axes is the x=y axis. (-x=y works as a C_2'' axis.) The σ_h plane is the xy plane. The σ_d planes contain the C_2'' axis and the z axis. The σ_v planes contain the C_2' axis and the z axis.

```
geometry units au
  C 1.855 1.855 0
  H 3.289 3.289 0
  symmetry D6h
end
```

19.2.12 C_{3h} H_3BO_3

The C_3 axis is the z axis. The σ_h plane is the xy plane.

```
geometry units au
  b 0 0 0
  o 2.27238285 1.19464491 0.00000000
  h 2.10895420 2.97347707 0.00000000
 symmetry C3h
end
```

19.2.13 D_{5d} ferrocene

The C_5 axis is the z axis. The center of inversion is the origin. One of the perpendicular C_2 axes is the x axis. One of the σ_d planes is the yz plane.

```
geometry units angstroms
  symmetry d5d

  fe 0 0     0
  c  0 1.194 1.789
  h  0 2.256 1.789
end
```

19.2.14 C_{4v} SF_5Cl

The C_4 axis is the z axis. The σ_v planes are the yz and the xz planes. The σ_d planes are: 1) the plane containing the x=y axis and the z axis and 2) the plane containing the -x=y axis and the z axis.

```
geometry units au
  S  0.00000000 0.00000000 -0.14917600
  Cl 0.00000000 0.00000000  4.03279700
  F  3.13694200 0.00000000 -0.15321800
  F  0.00000000 0.00000000 -3.27074500

  symmetry C4v
end
```

19.2.15 C_{2h} trans-dichloroethylene

The C_2 axis is the z axis. The origin is the inversion center. The σ_h plane is the xy plane.

```
geometry units angstroms
  C  0.65051239 -0.08305064 0
  Cl 1.75249381  1.30491767 0
  H  1.14820954 -1.04789741 0
  symmetry C2h
end
```

19.2.16 D_{2d} CH_2CCH_2

The C_2 axis is the z axis (z is also the S_4 axis). The x and y axes are the perpendicular C_2's. The σ_d planes are: 1) the plane containing the x=y axis and the z axis and 2) the plane containing the -x=y axis and the z axis.

```
geometry units angstroms
  symmetry d2d
  c 0     0     0
  c 0     0     1.300
  h 0.656 0.656 1.857
end
```

19.2.17 D_{5h} cyclopentadiene anion

The C_5 axis is the z axis (z is also the S_5 axis). The y axis is one of the perpendicular C_2 axes. The σ_h plane is the xy plane and one of the σ_d planes is the yz plane.

```
charge -1
geometry units angstroms
  symmetry d5h
  c 0 1.1853 0
  h 0 2.2654 0
end
```

19.2.18 D_{4h} gold tetrachloride

The C_4 axis is the z axis (z is also the S_4 axis). The C_2' axes are the x and y axes and the C_2'' axes are the x=y axis and the x=-y axis. The inversion center is the origin. The σ_h plane is the xy plane. The σ_v planes are the xz and yz planes and the σ_d planes are 1) the plane containing the x=-y axis and the z axis and 2) the plane containing the x=y axis and the z axis.

```
geometry units au
  Au 0 0     0
  Cl 0 4.033 0
  symmetry D4h
end
```

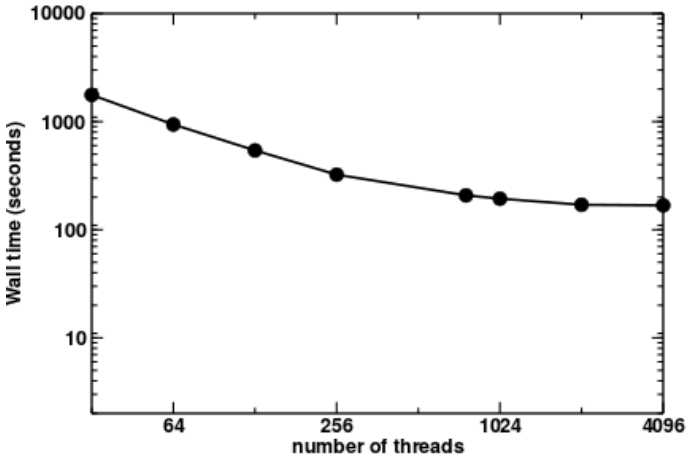
20 Benchmarks performed with NWChem

This page contains a suite of benchmarks performed with NWChem. The benchmarks include a variety of computational chemistry methods on a variety of high performance computing platforms. The list of benchmarks available will evolve continuously as new data becomes available. If you have benchmark information you would like to add for your computing system, please contact one of the developers.

20 Hybrid density functional calculation on the C_{240} Buckyball

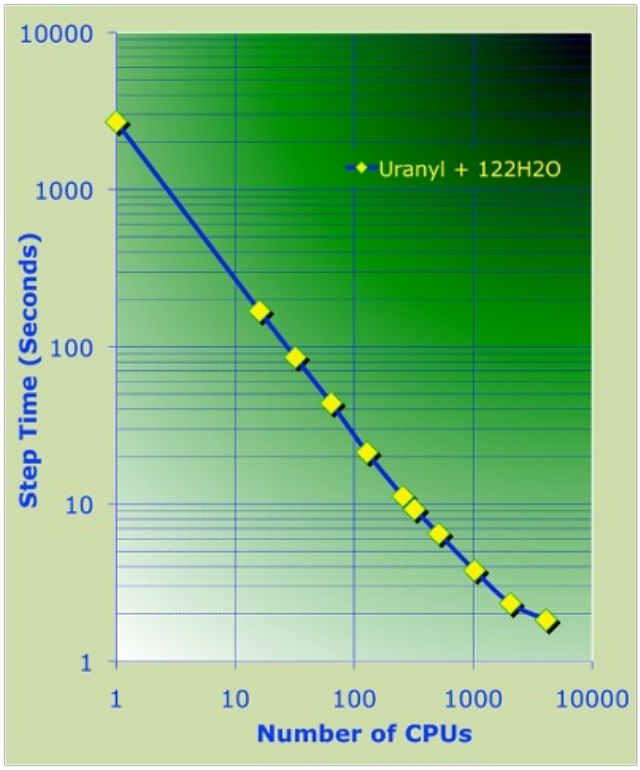
Performance of the Gaussian basis set DFT module in NWChem. This calculation involved performing a PBE0 calculation (in direct mode) on the on C_{240} system with the 6-31G* basis set (3600 basis functions) without symmetry. These

calculations were performed on the Cascade supercomputer located at PNNL. Input and output files are available.

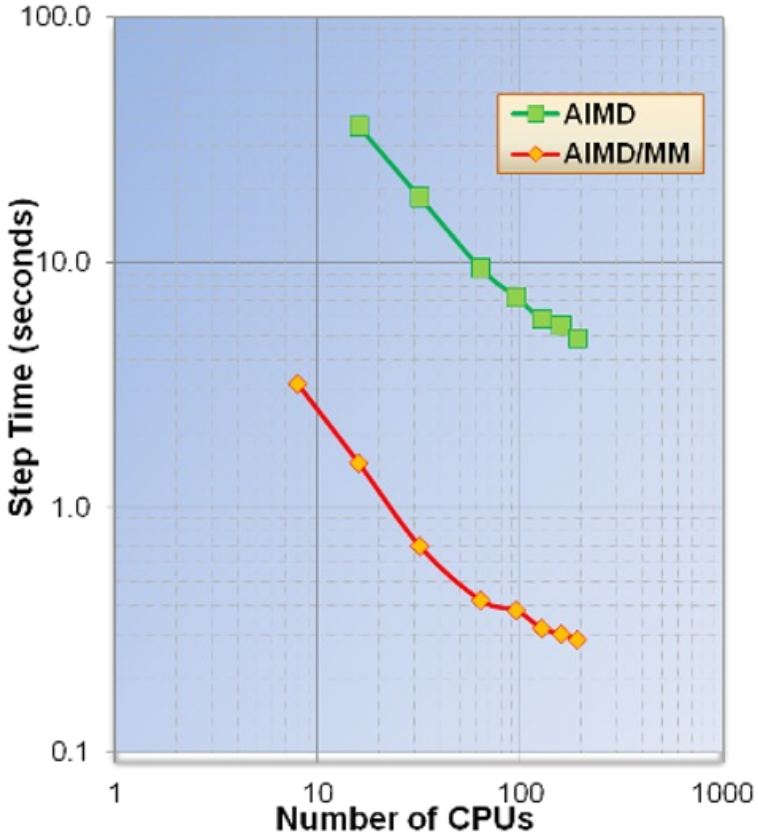


20 Parallel performance of *Ab initio* Molecular Dynamics using plane waves

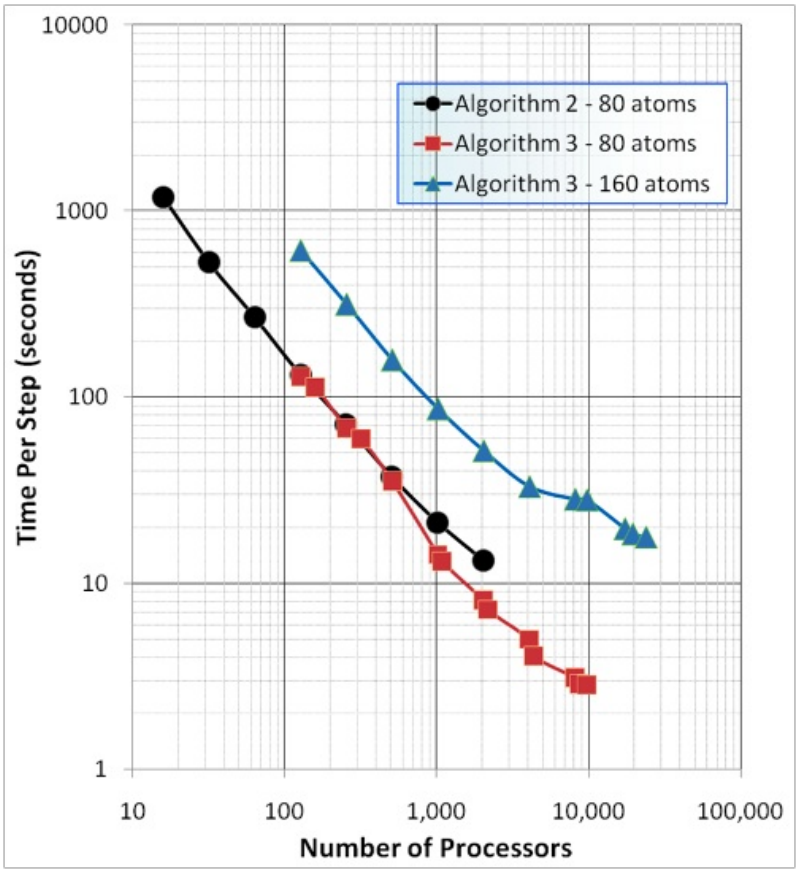
AIMD Parallel timings for \(UO_2^{2+}\)+122\(H_2\)O. These calculations were performed on the Franklin Cray-XT4 computer system at NERSC.



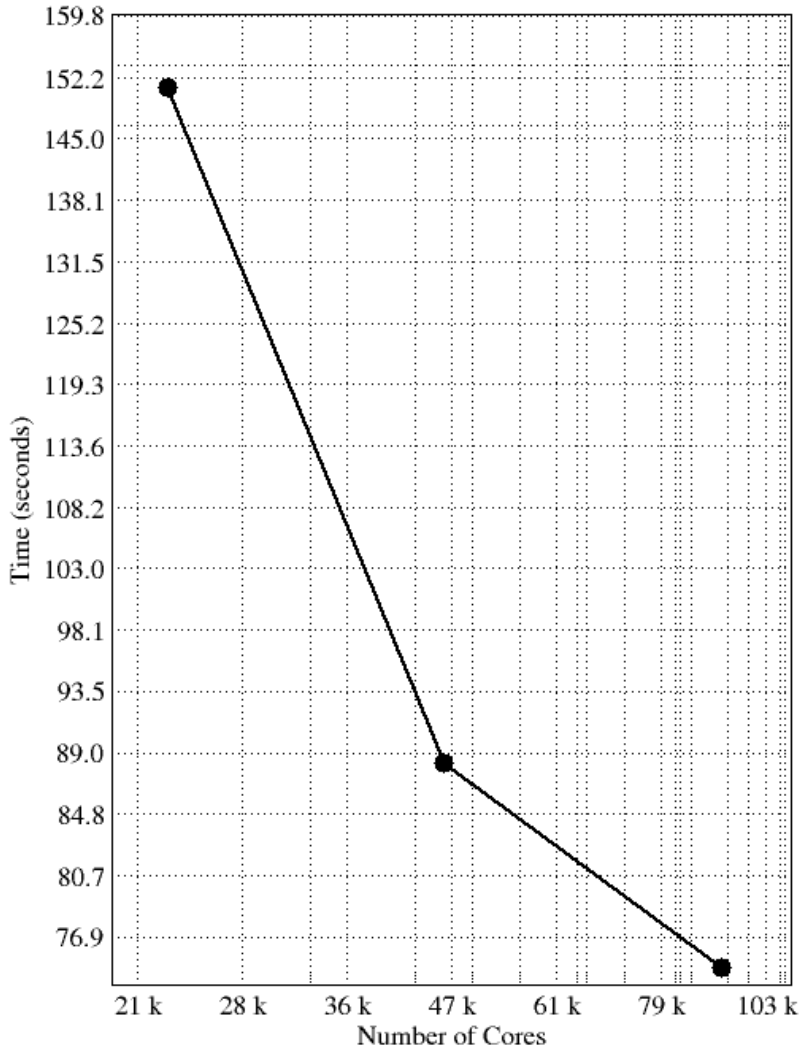
AIMD and AIMD/MM Parallel Timings for \(Zn^{2+}\)+64\(H_2\)O (unit cell parameters SC=12.4 Angs. and cutoff energy =100Ry). These calculations were performed on the Chinook HP computer system at MSCF EMSL, PNNL.



Exact exchange timings – 80 atom cell of hematite (cutoff energy=100Ry). These calculations were performed on the Franklin Cray-XT4 computer system at NERSC.

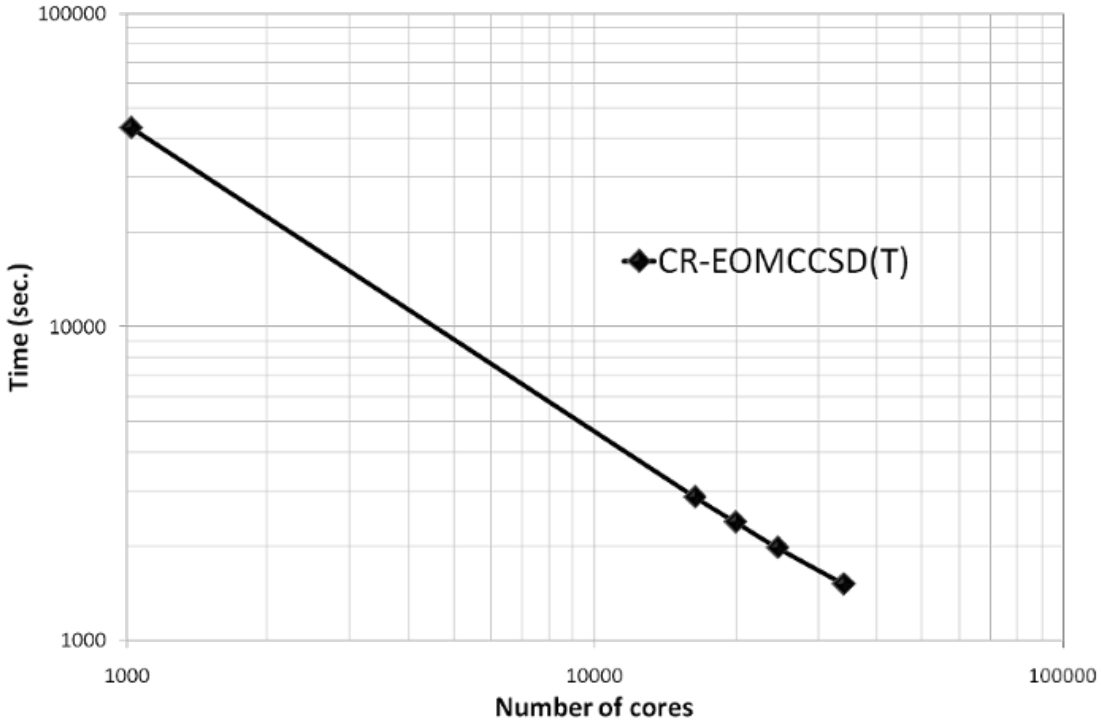


Exact exchange timings – 576 atom cell of water (cutoff energy=100Ry). These calculations were performed on the Hopper Cray-XE6 computer system at NERSC.

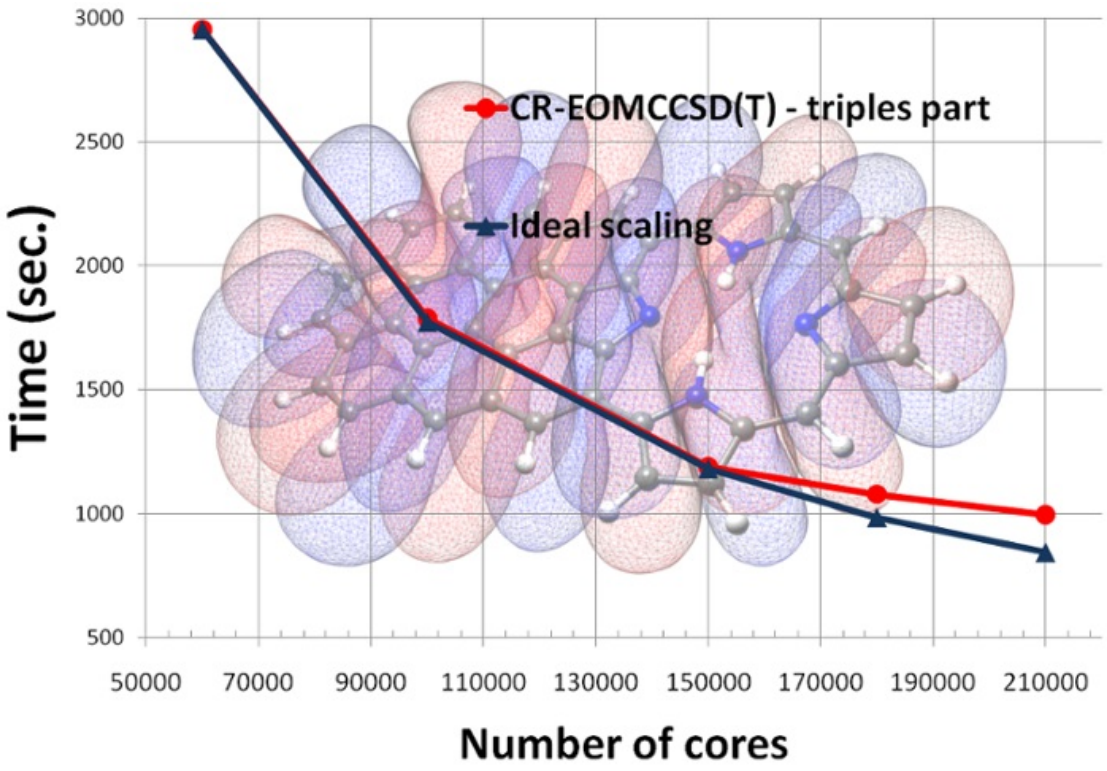


20 Parallel performance of the CR-EOMCCSD(T) method (triples part)

An example of the scalability of the triples part of theCR-EOMCCSD(T) approach for Green Fluorescent Protein Chromophore (GFPC) described by cc-pVTZ basis set (648 basis functions) as obtained from NWChem. Timings were determined from calculations on the Franklin Cray-XT4 computer system at NERSC. See the input file for details.



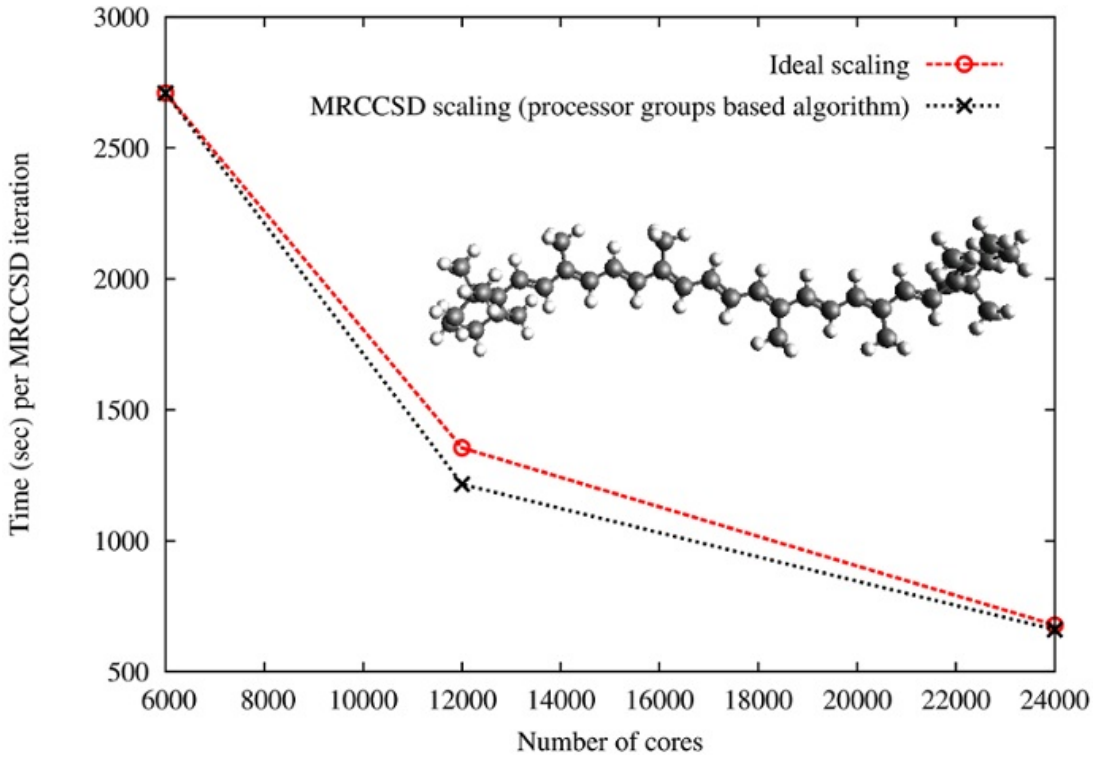
And more recent scalability test of the CR-EOMCCSD(T) formalism (Jaguar Cray XT5 at ORNL, see K. Kowalski, S. Krishnamoorthy, R.M. Olson, V. Tipparaju, E. Aprà , SC2011, for details).



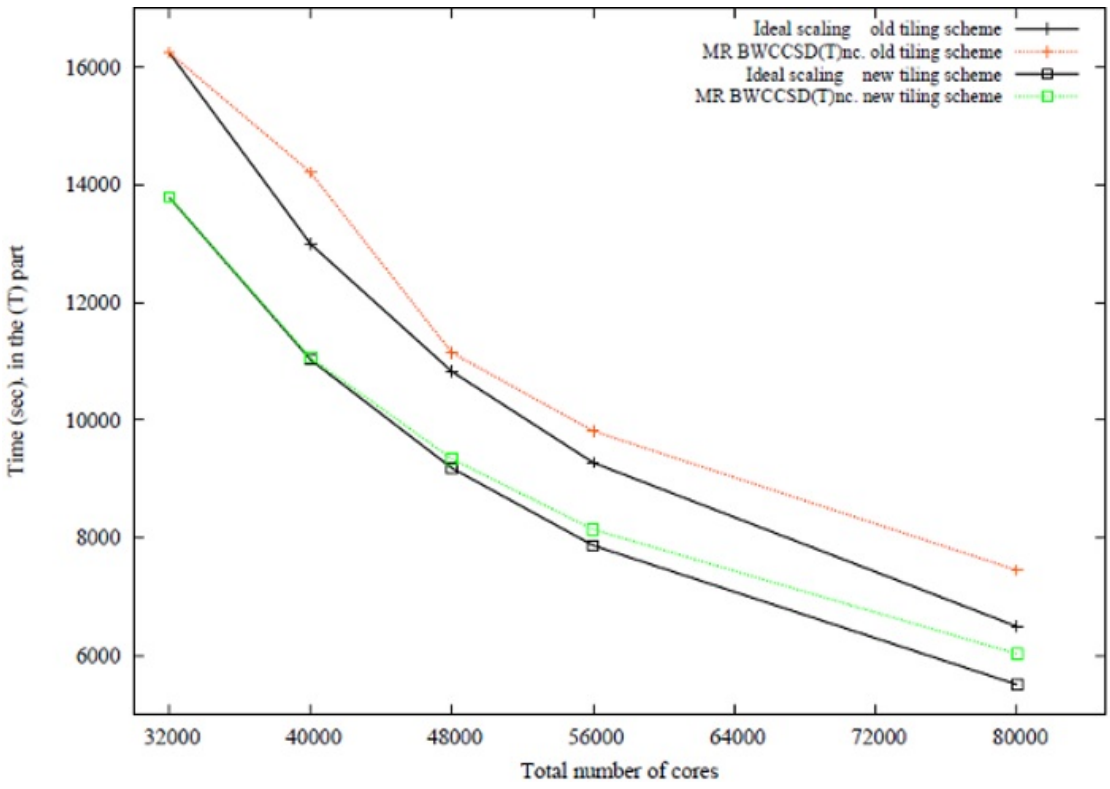
20 Parallel performance of the multireference coupled cluster (MRCC) methods

In collaboration with Dr. Jiri Pittner's group from Heyrovsky Institute of Physical Chemistry implementations of two variants of state-specific MRCC approaches have been developed. During his internship at PNNL Jirka Brabec, using novel processor-group-based algorithms, implemented Brillouin-Wigner and Mukherjee MRCC models with singles and doubles. The scalabililty tests for the Brillouin-Wigner MRCCSD approach have been performed on Jaguar XT5 system at ORNL for \(\beta\)-carotene in 6-31 basis set (472 orbitals, 216 correlated electrons, 20 reference functions; see J.Brabec, J. Pittner,

H.J.J. van Dam, E. Aprà, K. Kowalski, JCTC 2012, 8(2), pp 487–497). The input file and output files for runs at6000 cores, at 12000 cores and at 24000 cores are available.



Former PNNL postdoctoral fellow Dr. Kiran Bhaskaran Nair developed perturbative MRCCSD(T) approaches, which accounts for the effect of triple excitations. Scaling of the triples part of the BW-MRCCSD(T) method for \(\beta\)" -carotene in 6-31 basis set (JCP 137, 094112 (2012)). The scalability tests of the BW-MRCCSD(T) implementation of NWChem have been performed on the Jaguar Cray-XK6 computer system of the National Center for Computational Sciences at Oak Ridge National Laboratory.



20 Timings of CCSD/EOMCCSD for the oligoporphyrin dimer

CCSD/EOMCCSD timings for oligoporphyrin dimer (942 basis functions, 270 correlated electrons, D2h symmetry, excited-state calculations were performed for state of b1g symmetry, in all test calculation convergence threshold was relaxed, 1024 cores were used). See the input file for details.

```
------------------------------------------------------------
 Iter          Residuum       Correlation     Cpu    Wall
------------------------------------------------------------
   1   0.7187071521175  -7.9406033677717   640.9   807.7
   ......
 MICROCYCLE DIIS UPDATE: 10 5
  11   0.0009737920958  -7.9953441809574   691.1   822.2
 ------------------------------------------------------------
 Iterations converged
 CCSD correlation energy / hartree =        -7.995344180957357
 CCSD total energy / hartree       =     -2418.570838364838890

 EOM-CCSD right-hand side iterations
 ------------------------------------------------------------------
     Residuum       Omega / hartree  Omega / eV    Cpu    Wall
 ------------------------------------------------------------------
......
Iteration   2 using    6 trial vectors
  0.1584284659595   0.0882389635508    2.40111   865.3  1041.2
Iteration   3 using    7 trial vectors
  0.0575982107592   0.0810948687618    2.20670   918.0  1042.2
```

20 Performance tests of the GPU implementation of non-iterative part of the CCSD(T) approach

Recent tests of the GPU CCSD(T) implementation performed on Titan Cray XK71 system at ORNL ($C_{22}H_{14}$, 378 basis set functions, C1 symmetry; 98 nodes: 8 cores per node + 1GPU)

Using 8 CPU cores

```
Using CUDA CCSD(T) code
Using 0 device(s) per node
CCSD[T] correction energy / hartree = -0.150973754992986
CCSD[T] correlation energy / hartree = -3.067917061062492
CCSD[T] total energy / hartree = -844.403376796441080
CCSD(T) correction energy / hartree = -0.147996460406684
CCSD(T) correlation energy / hartree = -3.064939766476190
CCSD(T) total energy / hartree = -844.400399501854849
Cpu & wall time / sec 9229.9 9240.3
```

Using 7 CPU cores and one GPU

```
Using CUDA CCSD(T) code
Using 1 device(s) per node
CCSD[T] correction energy / hartree = -0.150973754993019
CCSD[T] correlation energy / hartree = -3.067917061062597
CCSD[T] total energy / hartree = -844.403376796441307
CCSD(T) correction energy / hartree = -0.147996460406693
CCSD(T) correlation energy / hartree = -3.064939766476270
CCSD(T) total energy / hartree = -844.400399501854963
Cpu & wall time / sec 1468.0 1630.7
```

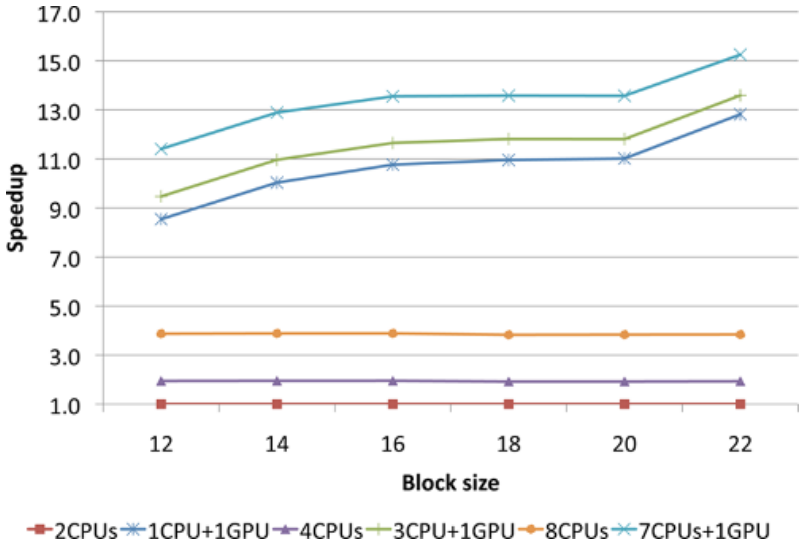
Using 1 CPU core and one GPU

```
Using CUDA CCSD(T) code
Using 1 device(s) per node
CCSD[T] correction energy / hartree = -0.150973754993069
CCSD[T] correlation energy / hartree = -3.067917061063028
CCSD[T] total energy / hartree =  -844.**************
CCSD(T) correction energy / hartree = -0.147996460406749
CCSD(T) correlation energy / hartree = -3.064939766476708
CCSD(T) total energy / hartree = -844.400399501861216
Cpu & wall time / sec 1410.9 1756.5
```

Without GPU 9240.3 sec. With GPU 1630.7 sec.

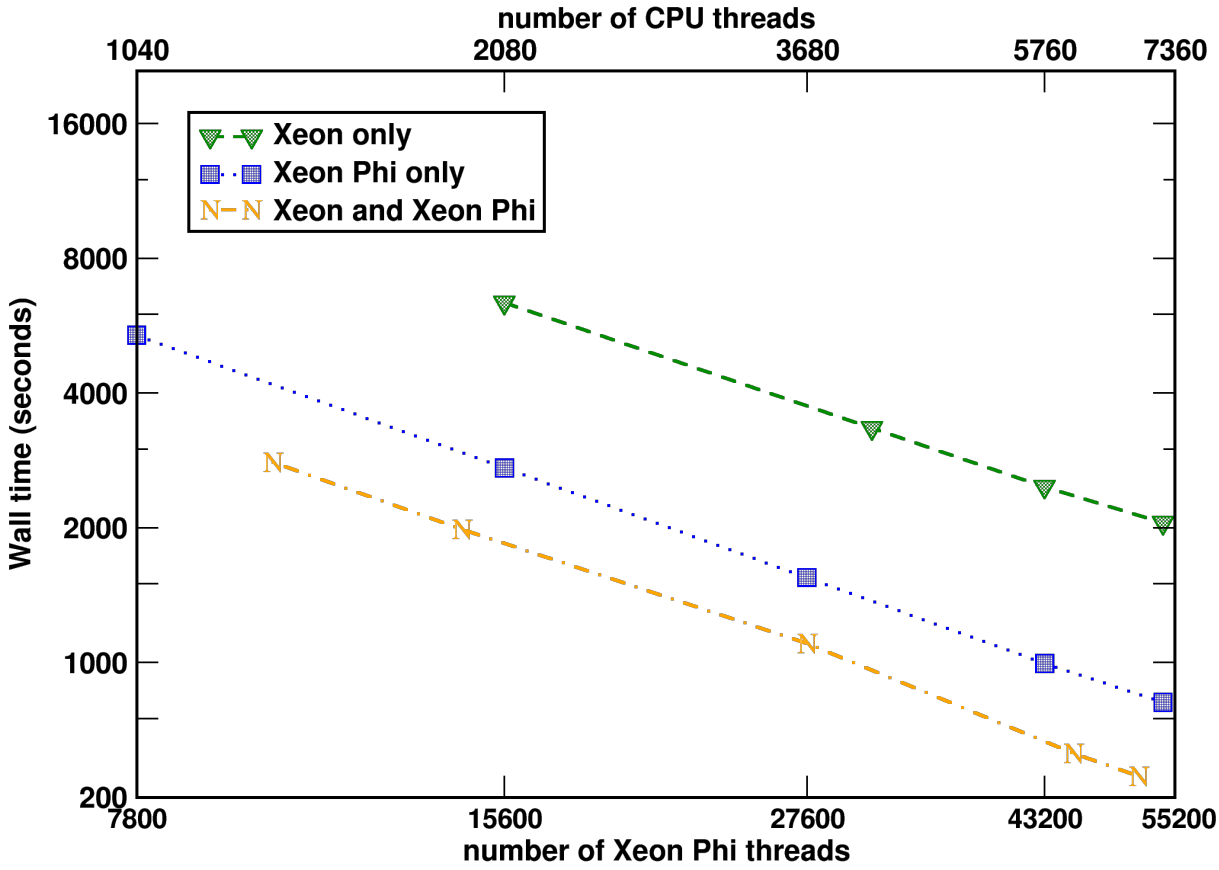
Next release: GPU implementation of non-iterative part of the MRCCSD(T) approach (K. Bhaskarsan-Nair, W. Ma, S.

Krishnamoorthy, O. Villa, H. van Dam, E. Aprà, K. Kowalski, J. Chem. Theory Comput. 9, 1949 (2013))



20 Performance tests of the Xeon Phi implementation of non-iterative part of the CCSD(T) approach

Tests of the Xeon Phi CCSD(T) implementation performed on the EMSL cascade system at PNNL



Aprà, E.; Klemm, M.; Kowalski, K., “Efficient Implementation of Many-Body Quantum Chemical Methods on the Intel® Xeon Phi Coprocessor,” High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for , vol., no., pp.674-684, 16-21 Nov. 2014 http://dx.doi.org/10.1109/SC.2014.60

(Triplet state of $Si_4C_3N_2H_{12}$, 706 basis set functions, C1 symmetry)

20 Non-iterative part of the CCSD(T) approach: Comparing Xeon Phi and NVidia K20X performance

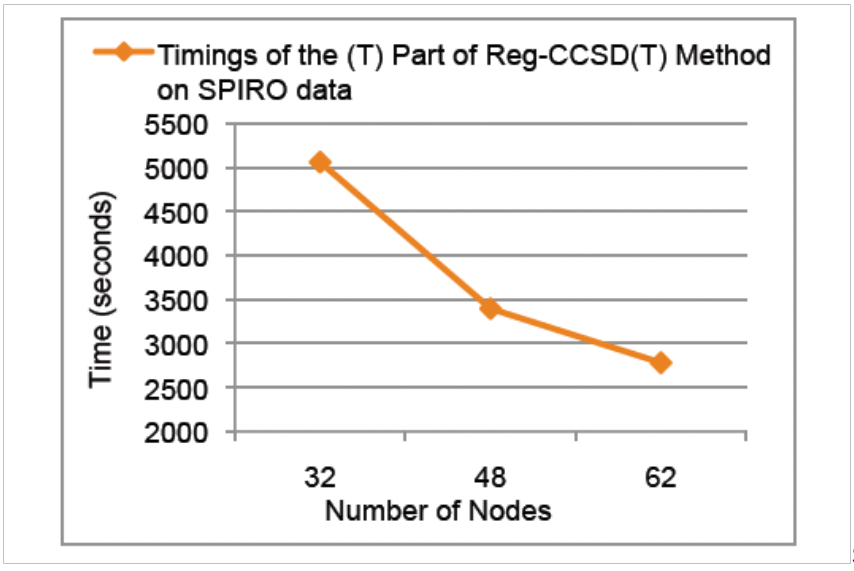
Wall time to solution (in seconds) of non-iterative triples part of the single-reference CCSD(T) approach for the pentacene molecule using Intel MIC and Nvidia GPU implementations. Tests were performed using 96 compute nodes on the Cascade system at EMSL (Intel® Xeon™ Phi 5110P) and Titan system at ORNL (NVIDIA Tesla® K20X).

(input file)

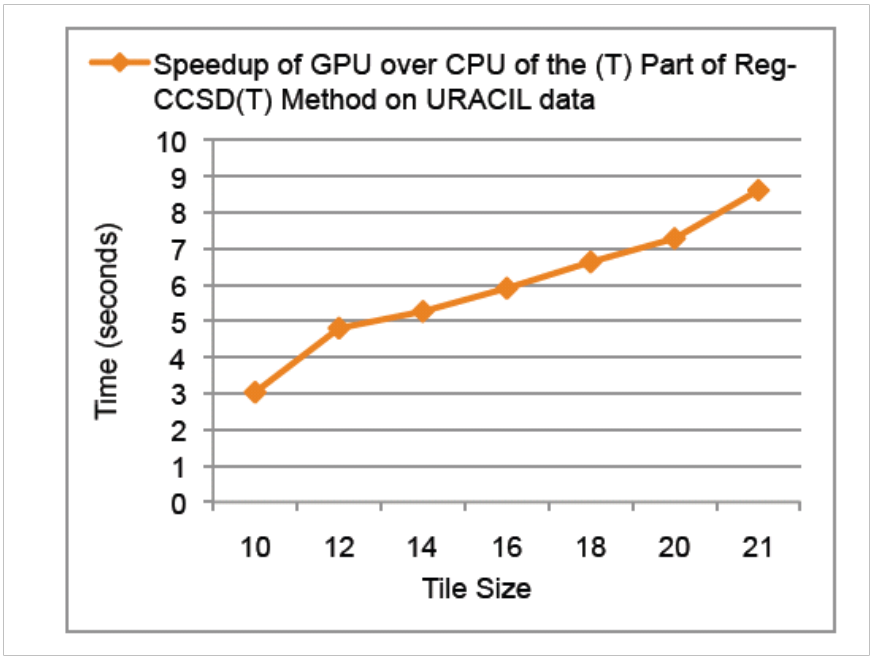
Tilesize	Intel Xeon Phi 5110P	Nvidia K20X
18	1806.4	1824.9
21	1652.2	1699.3
24	1453.3	1554.4

20 Current developments for high accuracy: alternative task schedulers (ATS)

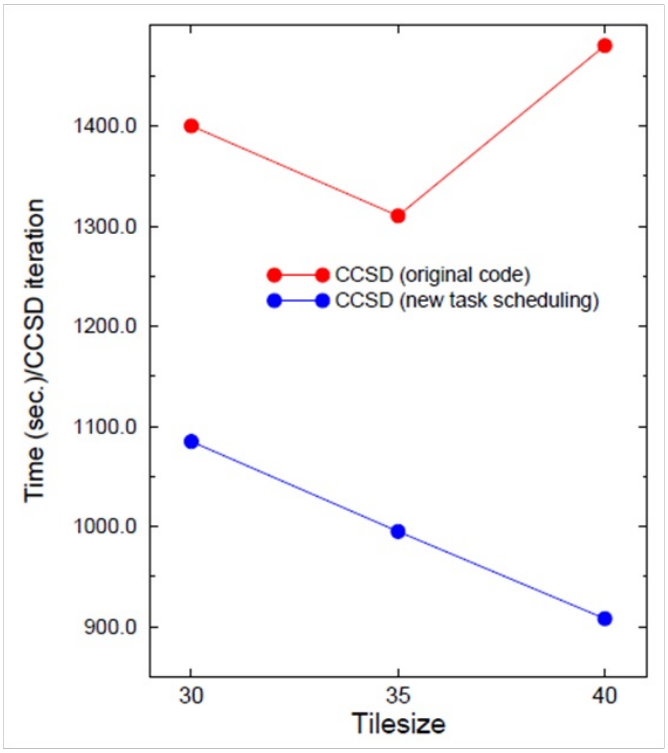
Currently various development efforts are underway for high accuracy methods that will be available in future releases of NWChem. The examples below shows the first results of the performance of the triples part of Reg-CCSD(T) on GPGPUs (left two examples) and of using alternative task schedules for the iterative CCSD and EOMCCSD.

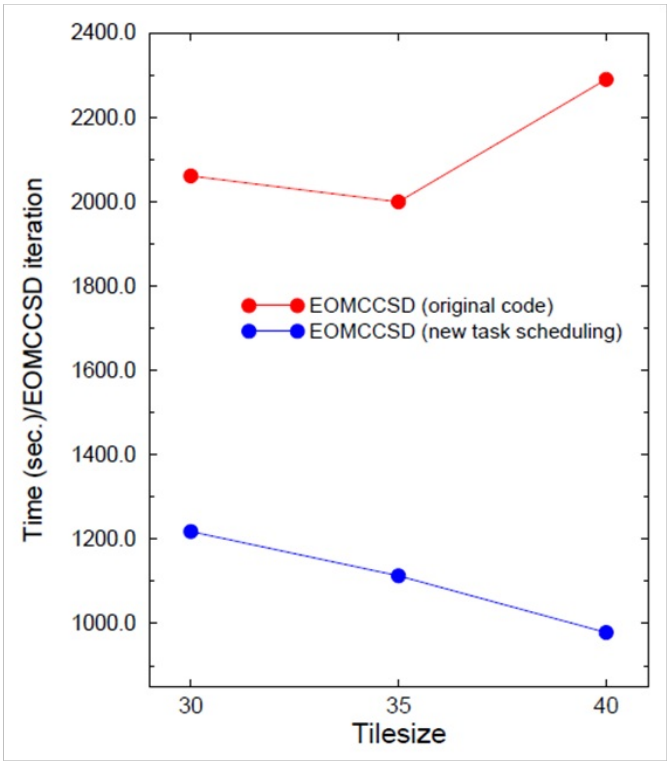


Scalability of the triples part of the Reg-CCSD(T) approach for Spiro cation described by the Sadlej's TZ basis set (POL1). The calculations were performed using

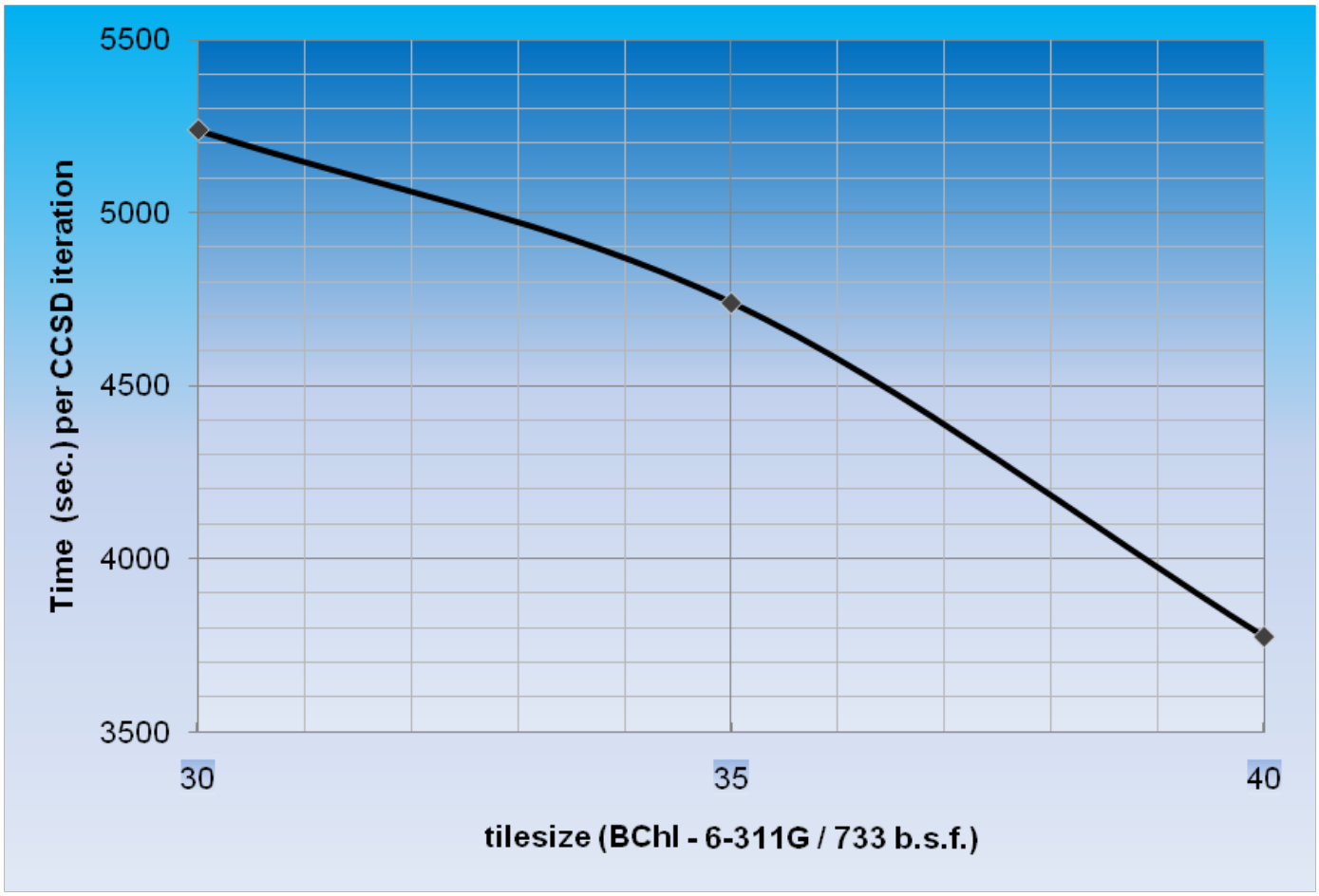


Barracuda cluster at EMSL. Speedup of GPU over CPU of the (T) part of the (T) part of the Reg-CCSD(T) approach as a function of the tile size for the uracil molecule. The calculations were performed using Barracuda cluster at EMSL.

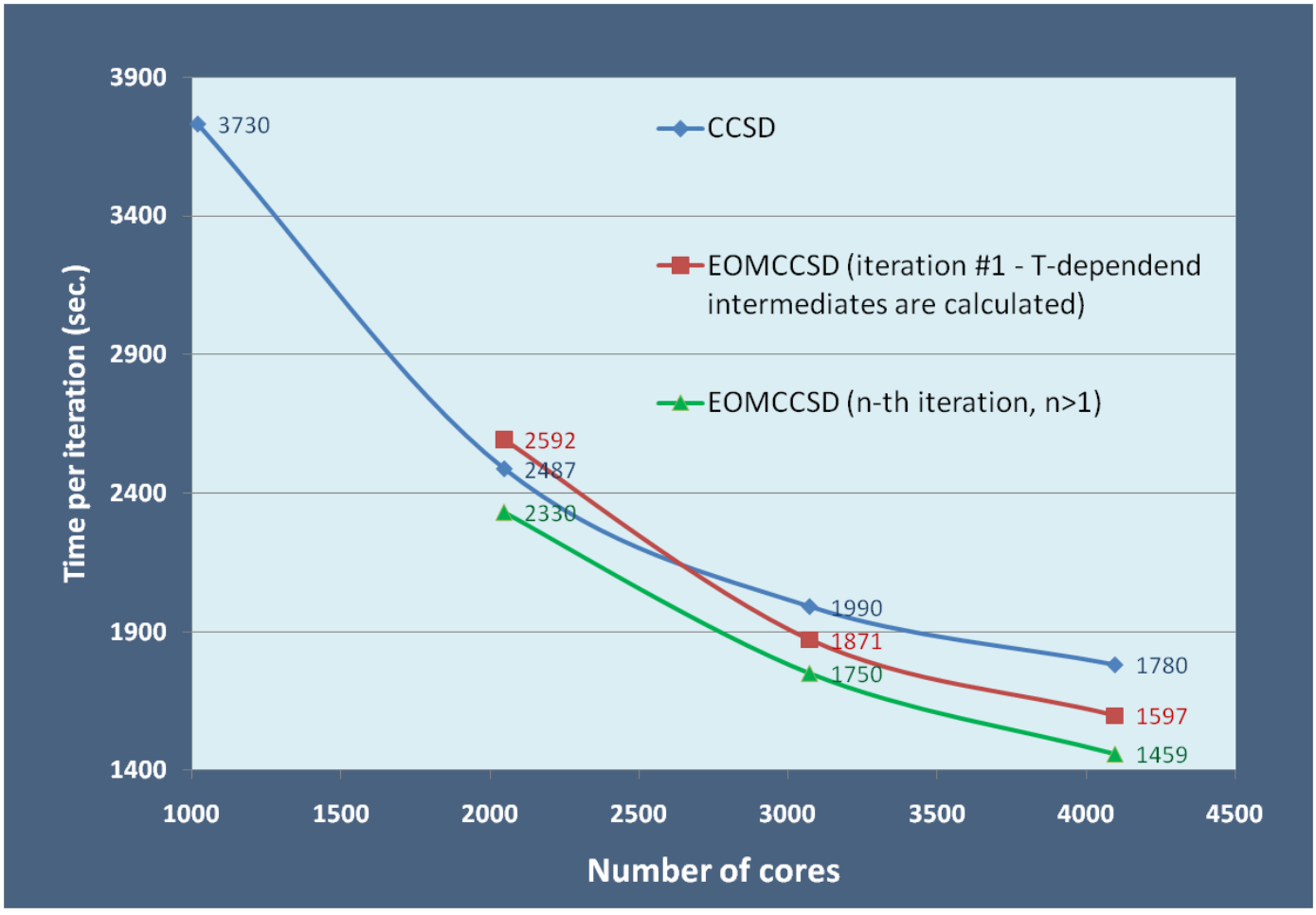




*Comparison of the CCSD/EOMCCSD iteration times for BacterioChlorophyll (BChl, Mg O6 N4 C 36 H38) for various tile sizes. Calculations were performed for 3-21G basis set (503 basis functions, C1 symmetry, 240 correlated electrons, 1020 cores).



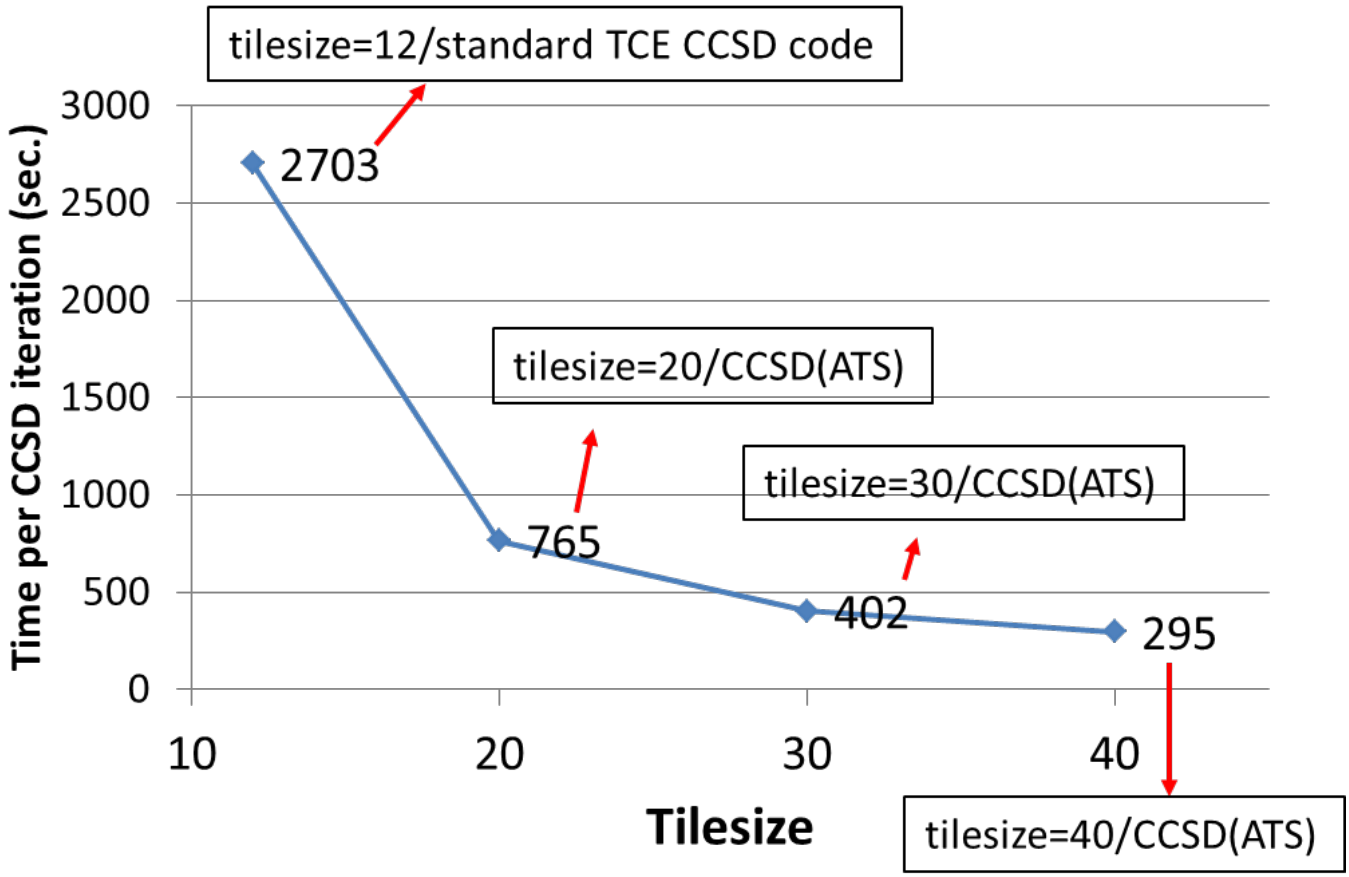
*Time per CCSD iteration for BChl in 6-311G basis set (733 basis functions, C1 symmetry, 240 correlated electrons, 1020 cores) as a function of tile size.



Scalability of the CCSD/EOMCCSD codes for BChl in 6-311G basis set (733 basis functions; tilesize=40, C1 symmetry, 240 correlated electrons).

Other tests:

The impact of the tilesize on the CCSD(ATS) timings: All tests have been performed for uracil trimer (6-31G* basis set; all core electrons frozen) on Hopper using 25 nodes (600 cores). One can observe almost 10-fold speedup of the CCSD(ATS) code for tilesize=40 compared to standard TCE CCSD implementation using tilesize=12.



Performance tests for water clusters

CCSD iteration time (in sec.). All tests have been performed using 64 nodes of Hopper.

System	CTF [a]	NWChem [a] (tilesize=12 ?)	NWChem: new task scheduling [b]
$(H_2O)_7$	90	178	70 (tilesize=30)
$(H_2O)_9$	127	---	84 (tilesize=40)

(a) Taken from: E. Solomonik, D. Matthews, J.R. Hammond, J. Demmel, 2013 IEEE 27th Symposium on Parallel & Distributed Processing. DOI 10.1109/IPDPS.2013.112

(b) Timings obtained with new iterative TCE CCSD algorithm (available in 6.1.1 and 6.3 NWChem releases): K. Kowalski, S. Krishnamoorthy, R. Olson, V. Tipparaju, E. Apra, SC2011 Seattle doi:10.1145/2063384.2063481. Calculations have been performed using 2eorb/2emet sequence to store 2-electron integrals using orbital representation (available for RHF and ROHF references). It means that each block of anti-symmetric 2-electron integrals in the spinorbital representation is created in an on-the-fly manner from the orbital 2-electron integrals.

Luciferin (aug-cc-pVDZ basis set; RHF reference; frozen core) - time per CCSD iteration (input file)

```
tilesize = 30
  256 cores     644 sec.
  512           378 sec.
  664           314 sec.
 1020           278 sec.
 1300           237 sec.


tilesize = 40
  128           998 sec.
  256           575 sec.
```

Sucrose (6-311G** basis set; RHF reference; frozen core) - time per CCSD iteration (input file)

```
tilesize = 40
  256 cores   1486 sec.
  512         910 sec.
 1024         608 sec.
```

Cytosine-OH (POL1; UHF reference; frozen core) - time per EOMCCSD iteration (input file)

```
tilesize = 30
256 cores    44.5 sec.


tilesize = 40
128 cores    55.6 sec.
```

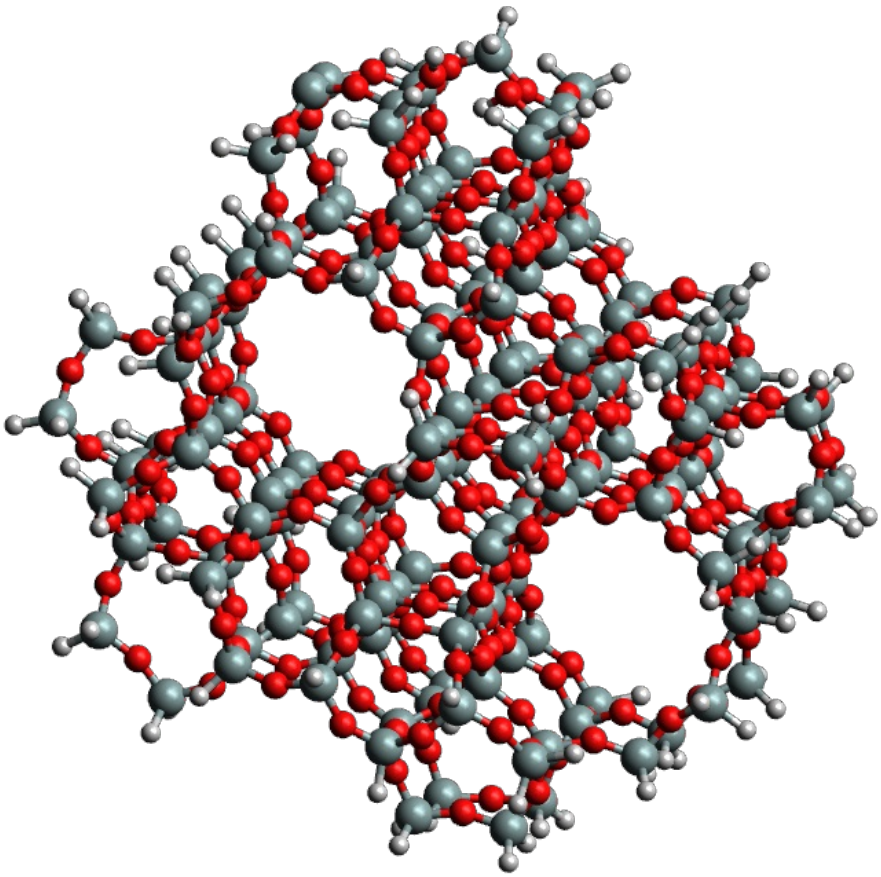
20 Density functional calculation of a zeolite fragment

Benchmark results with NWChem 7.0.0 for LDA calculations (energy plus gradient) on a 533 atomssiosi8 zeolite fragment. The input uses an atomic orbital basis set with 7108 functions and a charge density fitting basis with 16501 functions. The input file is available at this link.

computer	# nodes	cores/node	total # cores	Wall time (seconds)
cascade	9	16	144	1247
cascade	20	16	320	703
tahoma	4	36	144	927
tahoma	9	36	324	524

Hardware used:

- EMSL cascade specifications
- EMSL tahoma specifications



21 Known Bugs

21.0.1 How to report bugs

Use our Github issue tracker to report new bugs.

Also check there for other issues that may not have made it into these lists.

See the FAQ page for solutions to common issues.

21.0.2 Bugs present in NWChem binary packages

- The packages shipped with Ubuntu Bionic 18.04 are buggy. Calculations stop with the error

```
spcart_bra2etran: nbf_xj.ne.nbf_sj  (xj-sj) =                5
```

https://groups.google.com/g/nwchem-forum/c/RA0tYxdfvaw
https://nwchemgit.github.io/Special_AWCforum/st/id3386/Ubuntu_18.html
https://bugs.launchpad.net/ubuntu/+source/nwchem/+bug/1675817

Please use commands to install an updated version (as described at theNWChem 7.0.0 release page)

```
sudo apt -y install curl  python3-dev gfortran  mpi-default-bin mpi-default-dev libopenblas-dev ssh

curl -LJO https://github.com/nwchemgit/nwchem/releases/download/v7.0.0-release/nwchem-data_7.0.0-3_all.ubuntu_bionic.deb
curl -LJO https://github.com/nwchemgit/nwchem/releases/download/v7.0.0-release/nwchem_7.0.0-3_amd64.ubuntu_bionic.deb

sudo dpkg -i nwchem-data_7.0.0-3*_bionic.deb  nwchem_7.0.0-3*_bionic.deb
```

21.0.3 Known bugs for NWChem 6.8

- AR compilation failure on Mac OSX https://github.com/nwchemgit/nwchem/issues/5 Temporary fix: set the env. variable USE_ARUR=n, e.g.

```
make USE_ARUR=n
```

Fix available in the branches master and hotfix/release-6-8

- Moldenfile property bug when symmetry and linear dependencies are present https://github.com/nwchemgit/nwchem/issues/7 Workaround described in the issue entry

22 General information about NWChem

22.1 Where is the User's Manual?

The NWChem User's Manual is now athttps://nwchemgit.github.io/Home.html

22.2 Where do I go for help with a Global Arrays problem?

If you have problems with compiling the tools directory, please visit the Global Arrays website at https://globalarrays.github.io/

22.3 Where do I go for help with NWChem problems?

Please post your NWChem issue to the NWChem forum hosted on Google Groups at https://groups.google.com/g/nwchem-forum

22.4 Where do I find the installation instructions?

For updated instructions for compiling NWChem please visit the following URLhttps://nwchemgit.github.io/Compiling-NWChem.html

22.5 Installation Problem for the tools directory

When compiling the tools directory, you might see the compilation stopping with the message

```
configure: error: could not compile simple C MPI program
```

This is most likely due to incorrect settings for the MPI_LIB , MPI_INCLUDE and LIBMPI environment variables. The suggested course of action is to unset all of the three variables above and point your PATH env. variable to the location of mpif90 . If bash is your shell choice, this can be accomplished by typing

```
unset MPI_LIB
unset MPI_INCLUDE
unset LIBMPI
export PATH="directory where mpif90 is located":$PATH
```

22.6 What are ARMCI and ARMCI_NETWORK?

ARMCI is a library used by Global Arrays (both ARMCI and GA source code is located in NWChem's tools directory). More information can be found at the following URL http://hpc.pnl.gov/armci
If your installation uses a fast network and you are aiming to get optimal communication performance, you might want to assign a non-default value to ARMCI_NETWORK .
The following links contained useful information about ARMCI_NETWORK :

- Choosing the ARMCI library
- Choosing the proper environment variables when compiling NWChem

22.7 Input Problem: no output

You might encounter the following error message:

```
! warning: processed input with no task
```

Have you used emacs to create your input file? Emacs usually does not put and an end-of-line as a last character of the file, therefore the NWChem input parser ignores the last line of your input (the one containing the task directive). To fix the problem, add one more blank line after the task line and your task directive will be executed.

22.8 Input problem: AUTOZ fails to generate valid internal coordinates

If AUTOZ fails, NWChem will default to using Cartesian coordinates (and ignore any zcoord data) so you don't have to do anything unless you really need to use internal coordinates. An exception are certain cases where we have a molecule that contains a linear chain of 4 or more atoms, in which case the code will fail (see item 2. for work arounds). For small systems you can easily construct a Z-matrix, but for larger systems this can be quite hard.

First check your input. Are you using the correct units? The default is Angstroms. If you input atomic units but did not tell NWChem, then it's no wonder things are breaking. Also, is the geometry physically sensible? If atoms are too close to each other you'll get many unphysical bonds, whereas if they are too far apart AUTOZ will not be able to figure out how to connect things.

Once the obvious has been checked, there are several possible modes of failure, some of which may be worked around in the input.

1. Strictly linear molecules with 3 or more atoms. AUTOZ does not generate linear bend coordinates, but, just as in a real Z-matrix, you can specify a dummy center that is not co-linear. There are two relevant tips:
2. constrain the dummy center to be not co-linear otherwise the center could become co-linear. Also, the inevitable small

forces on the dummy center can confuse the optimizer.

3. put the dummy center far enough away so that only one connection is generated.

E.g., this input for acetylene will not use internals

```
geometry
  h  0  0  0
  c  0  0  1
  c  0  0  2.2
  h  0  0  3.2
end
```

but this one will

```
geometry
  zcoord
    bond    2 3  3.0  cx constant
    angle 1 2 3 90.0 hcx constant
  end
  h  0  0  0
  c  0  0  1
  x  3  0  1
  c  0  0  2.2
  h  0  0  3.2
end
```

1. Larger molecules that contain a strictly linear chain of four or more atoms (that ends in a free atom). For these molecules the autoz will fail and the code can currently not recover by using cartesians. One has to explicitly define noautoz in the geometry input to make it work. If internal coordinates are required one can fix it in the same manner as described above. However, you can also force a connection to a real nearby atom.
2. Very highly connected systems generate too many internal coordinates which can make optimization in redundant internals less efficient than in Cartesians. For systems such as clusters of atoms or small molecules, try using a smaller value of the scaling factor for covalent radii

```
zcoord; cvr_scaling 0.9; end
```

In addition to this you can also try specifying a minimal set of bonds to connect the fragments.

If these together don't work, then you're out of luck. Use either Cartesians coordinates (by using the geometry NOAUTOZ keyword) or supply your own Z-matrix (using the ZMATRIX input option).

22.9 How do I restart a geometry optimization?

If you have saved the restart information that is kept in the permanent directory, then you can restart a calculation, as long as it did not crash while writing to the data base.

Following are two input files. The first starts a geometry optimization for ammonia. If this stops for nearly any reason such as it was interrupted, ran out of time or disk space, or exceeded the maximum number of iterations, then it may be restarted with the second job.

The key points are

- The first job contains a START directive with a name for the calculation.
- All subsequent jobs should contain a RESTART directive with the same name for the calculation.
- All jobs must specify the same permanent directory. The default permanent directory is the current directory.
- If you want to change anything in the restart job, just put the data before the task directive. Otherwise, all options will be the same as in the original job.

Job 1.

```
start ammonia
permanent_dir /u/myfiles

geometry
  zmatrix
    n
    h 1 nh
    h 1 nh 2 hnh
    h 1 nh 2 hnh 3 hnh -1
  variables
    nh 1.
    hnh 115.
  end
end

basis
  n library 3-21g; h library 3-21g
end

task scf optimize
```

Job 2.

```
restart ammonia
permanent_dir /u/myfiles

task scf optimize
```

22.10 Execution Problem: How do I set the value of ARMCI_DEFAULT_SHMMAX?

Some ARMCI_NETWORK values (e.g. OPENIB) depend on the ARMCI_DEFAULT_SHMMAX value for large allocations of Global memory. We recommend a value of – at least – 2048, e.g. in bash shell parlance

```
export ARMCI_DEFAULT_SHMMAX=2048
```

A value of 2048 for ARMCI_DEFAULT_SHMMAX corresponds to 2048 GBytes, equal to 2048*1024*1024=2147483648 bytes. For ARMCI_DEFAULT_SHMMAX=2048 to work, it is necessary that kernel parameter kernel.shmmax to be greater than 2147483648. You can check the current value of kernel.shmmax on your system by typing

```
sysctl kernel.shmmax
```

More detail about kernel.shmmax can be found at thislink

22.11 WSL execution problems

NWChem runs on Windows Subsystem for Linux (WSL) can crash with the error message

```
--------------------------------------------------------------------------
WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

  Local host: hostabc
--------------------------------------------------------------------------
[hostabc:16805] 1 more process has sent help message help-btl-vader.txt / cma-permission-denied
[hostabc:16805] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
```

The error can be fixed with the following command

```
echo 0 | sudo tee /proc/sys/kernel/yama/ptrace_scope
```

More details at

- https://github.com/Microsoft/WSL/issues/3397#issuecomment-417876710
- https://nwchemgit.github.io/Special_AWCforum/st/id2939/mpirun_nwchem_on_Windows_Subsyst....html

22.12 How do I increase the number of digits of the S matrix printout

The only way to increase the number of digits of the AO overlap matrix printout is by modify the source code of the `ga_print()` function.

For example, in the cagse NWChem 7.0.2, you can do this by editing the C source code in $NWCHEM_TOP/src/tools/ga-5.7.2/global/src/global.util.c by increaseing the number of digits from 5 to 7

```
--- global.util.c.org 1969-07-20 15:50:45.000000000 -0700
+++ global.util.c 1969-07-20 15:51:19.000000000 -0700
@@ -122,22 +122,22 @@
           case C_DBL:
             pnga_get(g_a, lo, hi, dbuf, &ld);
             for(jj=0; jj<(jmax-j+1); jj++)
-              fprintf(file," %11.5f",dbuf[jj]);
+              fprintf(file," %11.7f",dbuf[jj]);
             break;
           case C_DCPL:
             pnga_get(g_a, lo, hi, dbuf, &ld);
             for(jj=0; jj<(jmax-j+1); jj+=2)
-              fprintf(file," %11.5f,%11.5f",dbuf[jj], dbuf[jj+1]);
+              fprintf(file," %11.7f,%11.7f",dbuf[jj], dbuf[jj+1]);
             break;
           case C_SCPL:
             pnga_get(g_a, lo, hi, dbuf, &ld);
             for(jj=0; jj<(jmax-j+1); jj+=2)
-              fprintf(file," %11.5f,%11.5f",dbuf[jj], dbuf[jj+1]);
+              fprintf(file," %11.7f,%11.7f",dbuf[jj], dbuf[jj+1]);
             break;
           case C_FLOAT:
             pnga_get(g_a, lo, hi, fbuf, &ld);
             for(jj=0; jj<(jmax-j+1); jj++)
-              fprintf(file," %11.5f",fbuf[jj]);
+              fprintf(file," %11.7f",fbuf[jj]);
             break;
           case C_LONG:
             pnga_get(g_a, lo, hi, lbuf, &ld);
@@ -229,22 +229,22 @@
           case C_DBL:
             pnga_get(g_a, lo, hi, dbuf, &ld);
             for(jj=0; jj<(jmax-j+1); jj++)
-              fprintf(file," %11.5f",dbuf[jj]);
+              fprintf(file," %11.7f",dbuf[jj]);
             break;
           case C_FLOAT:
             pnga_get(g_a, lo, hi, dbuf, &ld);
             for(jj=0; jj<(jmax-j+1); jj++)
-              fprintf(file," %11.5f",fbuf[jj]);
+              fprintf(file," %11.7f",fbuf[jj]);
             break;
           case C_DCPL:
             pnga_get(g_a, lo, hi, dbuf, &ld);
             for(jj=0; jj<(jmax-j+1); jj+=2)
-              fprintf(file," %11.5f,%11.5f",dbuf[jj], dbuf[jj+1]);
+              fprintf(file," %11.7f,%11.7f",dbuf[jj], dbuf[jj+1]);
             break;
           case C_SCPL:
             pnga_get(g_a, lo, hi, dbuf, &ld);
             for(jj=0; jj<(jmax-j+1); jj+=2)
-              fprintf(file," %11.5f,%11.5f",dbuf[jj], dbuf[jj+1]);
+              fprintf(file," %11.7f,%11.7f",dbuf[jj], dbuf[jj+1]);
             break;
           default: pnga_error("ga_print: wrong type",0);
         }
@@ -761,28 +761,28 @@
                     if(ndim > 1)
                       for(j=0; j<(hip[1]-lop[1]+1); j++)
                         if((double)dbuf_2d[j*bufsize+i]<100000.0)
-                          fprintf(file," %11.5f",
+                          fprintf(file," %11.7f",
                                   dbuf_2d[j*bufsize+i]);
                         else
                           fprintf(file," %.5e",
                                   dbuf_2d[j*bufsize+i]);
                     else
                       if((double)dbuf_2d[i]<100000.0)
-                        fprintf(file," %11.5f",dbuf_2d[i]);
```

```
+                          fprintf(file," %11.7f",dbuf_2d[i]);
                         else
                           fprintf(file," %.5e",dbuf_2d[i]);
                       break;
                     case C_FLOAT:
                       if(ndim > 1)
                         for(j=0; j<(hip[1]-lop[1]+1); j++)
-                          fprintf(file," %11.5f", fbuf_2d[j*bufsize+i]);
-                       else fprintf(file," %11.5f", fbuf_2d[i]);
+                          fprintf(file," %11.7f", fbuf_2d[j*bufsize+i]);
+                       else fprintf(file," %11.7f", fbuf_2d[i]);
                       break;
                     case C_DCPL:
                       if(ndim > 1)
                         for(j=0; j<(hip[1]-lop[1]+1); j++)
                           if(((double)dcbuf_2d[(j*bufsize+i)*2]<100000.0)&&((double)dcbuf_2d[(j*bufsize+i)*2+1]<100000.0))
-                            fprintf(file," %11.5f,%11.5f",
+                            fprintf(file," %11.7f,%11.7f",
                                     dcbuf_2d[(j*bufsize+i)*2],
                                     dcbuf_2d[(j*bufsize+i)*2+1]);
                           else
@@ -792,7 +792,7 @@
                       else
                         if(((double)dcbuf_2d[i*2]<100000.0) &&
                           ((double)dcbuf_2d[i*2+1]<100000.0))
-                          fprintf(file," %11.5f,%11.5f",
+                          fprintf(file," %11.7f,%11.7f",
                                   dcbuf_2d[i*2], dcbuf_2d[i*2+1]);
                         else
                           fprintf(file," %.5e,%.5e",
@@ -802,7 +802,7 @@
                       if(ndim > 1)
                         for(j=0; j<(hip[1]-lop[1]+1); j++)
                           if(((float)fcbuf_2d[(j*bufsize+i)*2]<100000.0)&&((float)fcbuf_2d[(j*bufsize+i)*2+1]<100000.0))
-                            fprintf(file," %11.5f,%11.5f",
+                            fprintf(file," %11.7f,%11.7f",
                                     fcbuf_2d[(j*bufsize+i)*2],
                                     fcbuf_2d[(j*bufsize+i)*2+1]);
                           else
@@ -812,7 +812,7 @@
                       else
                         if(((float)fcbuf_2d[i*2]<100000.0) &&
                           ((float)fcbuf_2d[i*2+1]<100000.0))
-                          fprintf(file," %11.5f,%11.5f",
+                          fprintf(file," %11.7f,%11.7f",
                                   fcbuf_2d[i*2], fcbuf_2d[i*2+1]);
                         else
                           fprintf(file," %.5e,%.5e",
```

https://nwchemgit.github.io/Special_AWCforum/sp/id3358.html

22.13 Linear Dependencies

Two or more basis functions can be consider linearly dependent when they span the same region of space. This can result in SCF converge problems. Analysis of the eigenvectors of the $S^{-1/2}$ matrix (where S is the overlap matrix) is used to detect linear dependencies: if there are eigenvalues close to zero, the basis set goes through the process of canonical orthogonalization (as described in Section 3.4.5 of Szabo & Ostlund “Modern Quantum Chemistry” book). This has net effect of a reduction of number of basis function used, compared to the original number set by input. By setting

```
set lindep:n_dep 0
```

this orthogonalization process is skipped.

22.14 Discrepancy on the number of basis functions: spherical vs cartesian functions

If you are comparing NWChem results with the ones obtained from other codes and you believe there is a discrepancy in the number of basis functions, keep in mind that NWChem uses cartesian functions by default, while other codes could be using spherical functions, instead.
If you need to use spherical functions, the beginning of the basis input field needs to be

```
basis spherical
```

More details in the documentation at the linkhttps://nwchemgit.github.io/Basis.html#spherical-or-cartesian.

See also the followingforum entries.

22.15 Starting NWChem with `mpirun -np 1` crashes

This is most likely due to the fact that NWChem was compiled with the setting`ARMCI_NETWORK=MPI-PR`.
This is the expected behavior, since `ARMCI_NETWORK=MPI-PR` requires asking for for n+1 processes. In other words, a serial run (with a single computing process) is triggered by executing `mpirun -np 2`.
If you would prefer `mpirun -np 1` to work, other choice of `ARMCI_NETWORK` are possible as described in theARMCI documentation.

22.16 nb_wait_for_handle Error

If you get the following error

```
{1} nb_wait_for_handle Error: all user-level nonblocking handles have been exhausted
application called MPI_Abort(comm=0x84000002, -1)
```

you can fix it by executing the following command

```
export COMEX_MAX_NB_OUTSTANDING=16
```

22.17 Memory errors

If you get the following error

```
[0] Received an Error in Communication: (-1) 0: ptsalloc: increase memory in input line:
```

you can fix it by either
* increasing the memory line in the input file using the syntax described in the Memory section (e.g. `memory total 1000 mb`), or by
* recompiling the NWChem binary with the `getmem.nwchem` script as described in the section avaible at this link

23 Tutorials

23.1 Links to material of past NWChem Tutorials

- EMSL Integration 2018 meeting Tutorial EMSL 2018
- VI-SEEM NAT-GR LS+ : 2018 NWChem Workshop (10-11 September 2018, Athens, Greece) Tutorial Athens 2018
- Tutorial 2019 EMSL/ARM Aerosol Summer School
- Tutorial 2012 Singapore A*STAR

23.2 Websites containing NWChem material

- Simple DFT calculations and processing with NWChem (Youtube Videos' playlist) from the Department of Materials and Optoelectronic Science at the National Sun Yat-sen University, Taiwan Link for Youtube playlist
- PRACE Spring School in Computational Chemistry 2019 https://web.archive.org/web/20221103195703/https://events.prace-ri.eu/event/786/attachments/840/1256/QC-

workshop-advanced.pdf

- Introduction To NWChem by B.J. Lynch (UMN) 2006
  https://web.archive.org/web/20230418140258/https://www.msi.umn.edu/sites/default/files/IntroNWChem.pdf
- Quantum Chemistry Course at Radboud University - Nijmegen https://www.theochem.ru.nl/quantumchemistry
- NWChem tips from C.J. O'Brien (NCSU)
  https://sites.google.com/a/ncsu.edu/cjobrien-nwchem-tips/

24 Forum

We use google groups for the users' forum.

The group is open to registered members.

Please do mention your real name as display name during the registration process.

The group can be accessed from the following link
https://groups.google.com/g/nwchem-forum

If you have trouble registering to this group, please send an email tonwchemgit@gmail.com

You can use the form below to search the entries of the forum

Search forum

27 Supplementary Information

27.2 Choosing the ARMCI Library

27.2.1 Overview

The Global Arrays parallel environment relies upon a one-sided communication runtime system. There are at least three options currently available:

- ARMCI - This is the original one-sided library developed with Global Arrays by PNNL. It supports the widest range of platforms. See http://hpc.pnl.gov/globalarrays/support.shtml for details.
- ComEx - This library is being developed by PNNL to replace ARMCI. It supports the fewer platforms than ARMCI but may be more robust on modern ones like InfiniBand. See http://hpc.pnl.gov/globalarrays/support.shtml for details.
- ARMCI-MPI - This library is a completely separate implementation of the ARMCI interface maintained by Jeff Hammond. It uses the one-sided communication features of MPI, rather than a platform-specific conduit. It currents provides both MPI-2 and MPI-3 implementations; the MPI-3 one is supported by nearly all platforms. See https://www.mpi-forum.org/implementation-status/ for details.

By default, Global Arrays will choose ARMCI or ComEx, based upon the environment variables selected by the user (see GA documentation for details). When a native implementation is available and works reliably, this is the best option for the NWChem user. However, in cases where an native implementation of ARMCI is not available or is not reliable, the user should consider using one of the MPI-based implementations.

There are many different ways to use MPI as the communication runtime of Global Arrays:

ARMCI_NETWORK	Result	Notes
MPI-PR	ARMCI with progress rank	High-performance option.
MPI-PT	ARMCI with progress thread	Not recommended.
MPI-MT	ARMCI over multi-threaded MPI	Do not use Open-MPI 1.x.
MPI-TS	ARMCI over MPI without data server	Universally portable but not always high-performance.
MPI-SPAWN	ARMCI using MPI dynamic processes	Not recommended. Requires MPI_Comm_spawn support.
ARMCI	Uses ARMCI-MPI. Requires EXTERNAL_ARMCI_PATH.	See https://github.com/pmodels/armci-mpi for details.

It is difficult to provide complete guidance to the user as to which option to choose. However, we observe the following:

- ARMCI_NETWORK=MPI-PR is stable and performs well on many platforms (including Cray XC platforms, e.g. NERSC Cori). This port will use one processes on each node for communication, therefore subtracting one process (again on each node) for NWChem. Therefore, when executing on a single node (i.e. the case of desktop execution) you would need to ask for n+1 processes; in other words, a serial execution would require the following mpirun invocation mpirun -np 2 ...
- On Intel Omni Path systems, MPI-PR is more reliable than OPENIB or MPI-SPAWN. ARMCI-MPI with Casper and Intel MPI is also recommended. See this page for details.
- The performance of ARMCI-MPI is greatly enhanced by Casper. See this link for design details.

When using ARMCI-MPI, please make sure to use the most recent version of MPI (MPICH 5.+, Cray MPI 7.2+, MVAPICH2 2.0+, Intel MPI 5.1+, Open-MPI 5.0+). Older versions of MPI are known to have bugs in the MPI-3 RMA features that affect the correctness of NWChem.

27.2.2 Support

Global Arrays, ARMCI and ComEx are developed and supported by PNNL. Usehttps://github.com/GlobalArrays/ga/issues for support.

ARMCI-MPI is maintained by Jeff Hammond. Usehttps://github.com/pmodels/armci-mpi/issues for support.

27.2.3 Automated installation of ARMCI-MPI

If you wish to use ARMCI-MPI, a script is available to automatically install it:

```
cd $NWCHEM_TOP/tools && ./install-armci-mpi
```

28 Interfaces with External Software

28.1 Overview

NWChem can be interfaced with external software packages by following the instructions below.

The interface can be set either using a pre-exisiting software package or by downloading and compiling the software from the NWChem makefile infrastructure.

28.2 Simint integrals library

To generate the Simint) library and enable the NWChem interface (only for energy and first derivative code), you need to define the following environment variables at compile time:

- USE_SIMINT=y (mandatory)
- SIMINT_MAXAM= “Maximum angular momentum” (optional, default is 3, therefore up to f orbitals)

The following set directives are required in the input file to trigger use of Simint

```
set int:cando_txs f
set int:cando_nw f
```

28.3 OpenBLAS

To build NWChem with the optimized BLAS and LapackOpenBLAS library, you need to define the following environment variables at compile time:

```
BUILD_OPENBLAS=1
BLAS_SIZE=8
```

This procedure requires an internet connection to dowload the OpenBLAS source.
Instead, to use a pre-compiled OpenBLAS library, the BLASOPT , LAPACK_LIB and BLAS_SIZE environment variable need to be set.

28.4 ScaLAPACK

To build NWChem with the ScaLAPACK library, you need to define the following environment variables at compile time:

```
BUILD_SCALAPACK=1
SCALAPACK_SIZE=8
```

This procedure requires an internet connection to dowload the OpenBLAS source.
Instead, to use a pre-compiled ScaLAPACK library, the SCALAPACK_LIB and SCALAPACK_SIZE environment variable need to be set, as described in the related Compiling NWChem section

28.5 ELPA

To build NWChem with the ELPA eigensolver library, you need first to set the ScaLAPACK settings as described in the previous section and then you need to define the following environment variables at compile time:

```
BUILD_ELPA=1
```

This procedure requires an internet connection to dowload the OpenBLAS source.
Instead, to use a pre-compiled ELPA library, the ELPA and ELPA_SIZE environment variable need to be set.

28.6 Plumed

The BUILD_PLUMED environment variable installs Plumed and interfaces it with the qmd module. This procedure requires an internet connection to dowload the Plumed source.

Instead, if you wish to use an existing Plumed installation, the following environment variables must be set (after having unset BUILD_PLUMED):

```
USE_PLUMED=1
```

Requirements: * The environment variable PATH should to point to the location of the plumed command and LD_LIBRARY_PATH should point to the location of the plumed libraries. * BLAS_SIZE and SCALAPACK_SIZE must be equal to 4 (this is not a requirement when using BUILD_PLUMED)

28.7 Libxc

Building NWChem with the libxc DFT library requires setting the environment variable USE_LIBXC=1.

This procedure requires an internet connection to dowload the Libxc source.

Instead, if you wish to use an existing libxc library, the following environment variables must be set, after having unset USE_LIBXC:

- LIBXC_INCLUDE location of the libxc C header files
- LIBXC_MODDIR location of the libxc fortran90 module files
- LIBXC_LIB location of the libxc libraries files

For example, for Debian/Ubuntu systems, the following is needed after having installed the libxc-dev package

```
unset USE_LIBXC
export LIBXC_LIB=/usr/lib/x86_64-linux-gnu
export LIBXC_INCLUDE=/usr/include
```

For example, for Fedora systems, the following is needed after having installed the libxc-devel package

```
unset USE_LIBXC
export LIBXC_LIB=/usr/lib64
export LIBXC_INCLUDE=/usr/include
export LIBXC_MODDIR=/usr/lib64/gfortran/modules
```

28.8 XTB

Building NWChem with the Light-weight tight-binding framework tblite requires

- setting the environment variable USE_TBLITE=1
- adding xtb to list of NWCHEM_MODULES

Example:

```
make nwchem_config NWCHEM_MODULES='tinyqmpw xtb'
export USE_TBLITE=1
make
```

29 Software supporting NWChem

29.1 Overview

While we have done our best to compile an exhaustive list of software using NWChem, we might have missed packages and/or incorrectly described some software features. Please use the Github Issue feature to provide feedback on this page content.

29.2 User interface software

- **ECCE** Extensible Computational Chemistry Environment https://github.com/FriendsofECCE/ECCE [1]

- **EMSL Arrows** Evolution of Chemical and Materials Computation https://nwchemgit.github.io/EMSL_Arrows.html
- **Avogadro** reads cube files, generates NWChem input files, analyzes output files (including frequencies) https://avogadro.cc
- **WebMO** World Wide Web-based interface to computational chemistry packages https://www.webmo.net/ [2]
- **Jmol** analyzes output and cube files https://wiki.jmol.org/index.php/NWChem
- **Scienomics MAPS** platform has a NWChem Plugin that will allow users to easily create NWChem input files. Since MAPS platform also has complex builders available, users can create complex models and then submit NWChem simulations to HPCs. MAPS also allows easy analysis of NWChem output files https://www.scienomics.com/maps-platform/simulate/quantum/nwchem-plugin/
- **CULGI** computational platform https://www.plm.automation.siemens.com/global/en/products/simcenter/culgi.html
- **Chemcraft** https://www.chemcraftprog.com
- **ASE** Atomic Simulation Environment https://wiki.fysik.dtu.dk/ase
- **Ascalaph** http://www.biomolecular-modeling.com/Ascalaph/index.html
- **MoCalc2012** https://mocalc2012.sourceforge.net/
- **Chemissian** https://www.chemissian.com/
- **Gausssum** a GUI application that can analyze the output since version 3.0 using the cclib library https://gausssum.sf.net
- **OpenDFT** brings cutting edge solid state research to the people https://github.com/JannickWeisshaupt/OpenDFT
- **STREAMM** generates structures and input files for quantum chemical and molecular dynamics codes https://github.com/NREL/streamm-tools
- **SparcleQC** is a Python package that creates QM/MM-like input files for quantum chemistry codeshttps://sparcle-qc.readthedocs.io
- **Winmostar** is a front-end GUI for pre- and post-processing quantum chemistry codeshttps://winmostar.com
- **ASH** is a Python-based computational chemistry and multiscale modelling programhttps://ash.readthedocs.io

29.3 Codes using NWChem wavefunctions and/or post-processing NWChem output files

- **KiSThelP** is a program to predict thermodynamic properties and rate constants from quantum chemistry results http://kisthelp.univ-reims.fr/
- **Fiesta** is a Gaussian-basis GW and Bethe-Salpeter code http://perso.neel.cnrs.fr/xavier.blase/fiesta/
- **JANPA** performs Natural Population Analysis https://janpa.sf.net
- **CamCASP** Cambridge package for Calculation of Anisotropic Site Properties https://gitlab.com/anthonyjs/camcasp/-/wikis/home
- **ChemShell** is a computational chemistry environment for standard quantum chemical or force field calculations https://www.chemshell.org
- **PUPIL** allows developers to perform multi-scale simulations https://pupil.sf.net
- **LICHEM** interfaces between QM and MM software https://github.com/kratman/LICHEM_QMMM
- **VENUS** interfaces NWChem with chemical dynamics https://www.depts.ttu.edu/chemistry/Venus/index.php https://www.sciencedirect.com/science/article/pii/S0010465513004049
- **VOTCA-XTP** is a GW-BSE code to calculate excited state properties http://www.votca.org
- **DP4-AI** integrates NMR-AI, software for automatic processing, assignment and visualisation of raw NMR data https://github.com/KristapsE/DP4-AI

- **Fafoom** Flexible algorithm for optimization of molecules https://github.com/adrianasupady/fafoom
- **Pymatgen** Python Materials Genomics open-source Python library for materials analysis http://pymatgen.org
- **PVSCF** A parallel vibrational self-consistent field program http://pvscf.org
- **ResLibCal** A tool to compute triple-axis neutron spectrometer resolution http://ifit.mccode.org/Applications/ResLibCal/doc/ResLibCal.html
- **SMFA** General program package for performing quantum chemistry calculations on large molecules using an energy-based fragmentation approach https://github.com/mickcollins/SMFAPAC
- **OCLIMAX** Free program for simulation of inelastic neutron scattering https://sites.google.com/site/ornliceman/download
- **Artaios** A code for calculating spin-dependent electron transport properties for molecular junctions in the coherent tunneling regime https://www.chemie.uni-hamburg.de/institute/ac/arbeitsgruppen/herrmann/software/artaios.html
- **Cuby** A computational chemistry framework that provides access to various computational methods available in different software package http://cuby.molecular.cz/?page=Interfaces
- **autoDIAS** A python tool for an automated Distortion/Interaction Activation Strain Analysis https://github.com/dsvatunek/autoDIAS
- **PolyParGen** provides OPLS-AA and Amber force field parameters for polymers or large molecules http://polypargen.com
- **BiKi Life Sciences** is a suite for Molecular Dynamics and related methods in Drug Discovery http://www.bikitech.com/products/
- **Shermo** is a general code for calculating molecular thermodynamic properties http://sobereva.com/soft/shermo/
- **QCengine** is a program executor and IO standardizer for quantum chemistry https://github.com/MolSSI/QCEngine
- **AiiDA plugin for NWChem** AiiDA is a Python infrastructure to deal with complex scientific workflows https://aiida-nwchem.readthedocs.io/
- **cclib** is a Python library for parsing and interpreting the results of computational chemistry packages https://cclib.github.io/
- **CHARMM** can perform combined Quantum Mechanical and Molecular Mechanics simulations using NWChem https://www.charmm.org/archive/charmm/documentation/by-version/c45b1/nwchem.html
- **kMap** is a program for simulation and data analysis in photoemission tomography https://github.com/brands-d/kMap
- **QMCube** is a Python suite focused on multiscale QM/MM simulations of biological systems https://github.com/sergio-marti/qm3
- **autodE** is a Python module designed for the automated generation of reaction profiles https://duartegroup.github.io/autodE
- **GoodVibes** is a Python program to compute thermochemical data from electronic structure calculations https://github.com/patonlab/GoodVibes
- **ChemDyME** is a Kinetically Steered, Automated Mechanism Generation Through Combined Molecular Dynamics and Master Equation Calculations https://github.com/RobinShannon/ChemDyME
- **xtbdft** is a wrapper script for multi-level molecular modelling powered by CREST/GFN2-XTB and NWChem (DFT) https://github.com/sibo/xtbdft
- **OctaDist** is an inorganic chemistry and crystallography program for computing the distortion parameters in coordination complexes https://octadist.github.io
- **PyADF** is a scripting framework for multiscale quantum chemistry https://github.com/chjacob-tubs/pyadf-releases
- **UniMoVib** is a unified interface for molecular harmonic vibrational frequency calculations https://github.com/zorkzou/UniMoVib
- **MoBioTools** is a toolkit to automatically setup QM/MM calculations https://github.com/mobiochem/MoBioTools

- **Fragme∩t** is a framework that makes it easy to prototype, implement, and benchmark fragmentation methods https://gitlab.com/john-herbert-group/fragment
- **pyVPT2** is a program to perform second-order vibrational perturbation theory (VPT2) computations to obtain anharmonic vibrational frequencies https://github.com/philipmnel/pyvpt2
- **Quick-QM-Spectra** creates a IR, UV-VIS or Raman spectrum from a NWChem output file https://hanlev.github.io/QQMS
- **Quantum Chemistry Operate** is a package for operating Quantum Chemistry programs using qcio standardized data structures https://github.com/coltonbh/qcop
- **SHARC** is a software package to study the excited-state dynamics of moleculeshttps://sharc-md.org/

29.4 Programs that can display or manipulate cube and/or Molden files

The following programs can display cube files from charge density and ESP and/or use Molden files

- **gOpenMol** https://web.archive.org/web/20090518024059/http://www.csc.fi/english/pages/g0penMol
- **Molden** https://www.theochem.ru.nl/molden/
- **Molekel** http://ugovaretto.github.io/molekel/
- **GaussView** http://www.gaussian.com/g_prod/gv5.htm
- **VMD** http://www.ks.uiuc.edu/Research/vmd
- **VESTA** http://jp-minerals.org/vesta/en/
- **Jamberoo** is a molecular viewer/editor http://www.jamberoo.org/
- **Molden2AIM** is a utility program which can be used to create AIM-WFN, AIM-WFX, and NBO-47 files from a Molden file https://github.com/zorkzou/Molden2AIM
- **Multiwfn** is a wavefunction analysis program http://sobereva.com/multiwfn
- **CrystalExplorer** is a program for Hirshfeld surface analysis, visualization and quantitative analysis of molecular crystals that can read NWChem Molden Files https://crystalexplorer.net/
- **VibeMol** is a molecular visualization and editing tool https://evangelistalab.org/vibemol/

29.5 Programs post-processing AIM files

NWChem can generate AIM³ wavefunction files (.wfn/.wfx) can be post-processed with a variety of codes, e.g.

- **XAIM** is a graphical user interface to several programs based on some aspects of the Theory of Atoms in Molecules http://www.quimica.urv.es/XAIM
- **NCIPLOT** computates and visualizes inter- and intra-molecular non-covalent interactions https://github.com/aoterodelaroza/nciplot
- **Multiwfn** performs a variety of electronic wavefunction analysishttp://sobereva.com/multiwfn/
- **Postg** calculates the dispersion energy and related quantities in gas-phase systems using the exchange-hole dipole moment (XDM) model https://github.com/aoterodelaroza/postg
- **GPUAM** computes Molecular Electrostatic Potential over Graphics Processing Units http://www.fqt.izt.uam.mx/Profes/JGO/GPUAM/GPUAM.html
- **CHARGEMOL** computes Density Derived Electrostatic and Chemical (DDEC) net atomic charges and atomic multipoles https://sourceforge.net/projects/ddec/
- **PAMoC** is a program for the analysis of experimental and theoretical electron charge density distributions https://www.pamoc.it/

- **DensToolKit2** is a suite of programs for analyzing electronic densities https://github.com/jmsolano/denstoolkit

1. No longer been actively developed at PNNL. New development effort at https://github.com/FriendsofECCE/ECCE/releases
2. The WebMo interface might not be compatible with NWChem 6.0 and later versions
3. **WARNING:** Since we have discovered issues in generating .WFN files with this module (e.g. systems with ECPs), the recommended method for generating .WFN file is to first generate a Molden file with the Moldenfile option, then convert the Molden file into a WFN file by using the Molden2AIM program.

30 Containers

30.1 Docker

30.1.1 Running NWChem using docker compose

Instruction for using the NWChem Docker container with thedocker compose command

- Install docker as described in https://docs.docker.com/engine/install
- Download the compose.yaml file

```
wget https://github.com/nwchemgit/nwchem-dockerfiles/raw/refs/heads/master/nwchem-cross/compose.yaml
```

- Create the nwchem service

```
docker compose up -d
```

In some installation, `docker compose` is available as the command `docker-compose`, therefore in those cases, this command becomes

```
docker-compose up -d
```

- Run NWChem

```
docker compose run --rm nwchem h2o.nw
```

In some installation, `docker compose` is available as the command `docker-compose`, therefore in those cases, this command becomes

```
docker-compose run --rm nwchem h2o.nw
```

In the example above the input file name is `h2o.nw`.
The default setting will run NWChem using 2 processes. If you would like to use more processes, then you would have to set the environment variable `MYNPROC` to the number of processes plus one. For example, if you wish to use 4 processes, then you will execute the following (bash syntax)

```
export MYNPROC=5
```

30.1.2 Running NWChem on MacOS using Apple container

After installing Apple Container as described athttps://github.com/apple/container on computers equipped with Apple silicon, it is possible to run NWChem docker images as in the example below.

```
container run -c 7 --shm-size 1G --memory 3G --volume $(pwd):/data --entrypoint=mpirun  --rm ghcr.io/nwchemgit/nwchemcross-dev-mpi-pr:arm64 -np 7 nwchem /data/siosi3.nw
```

30.2 Singularity/Apptainer

Singularity/Apptainer recipes for NWChem are available at.
https://github.com/edoapra/nwchem-singularity

Singularity images are available at ghcr.io/edoapra/nwchem-singularity/nwchem-dev.ompi41x
and ghcr.io/edoapra/nwchem-singularity/nwchem-dev.mpich3.4.2

30.2.1 Instruction for running on EMSL Tahoma

Slurm script for running NWChem Singularity/Apptainer images on EMSL tahoma

```
#!/bin/bash
#SBATCH -N 2
#SBATCH -t 00:29:00
#SBATCH -A allocation_name
#SBATCH --ntasks-per-node 36
#SBATCH -o singularity_library.output.%j
#SBATCH -e ./singularity_library.err.%j
#SBATCH -J singularity_library
#SBATCH --export ALL
source /etc/profile.d/modules.sh
export https_proxy=http://proxy.emsl.pnl.gov:3128
module purge
module load gcc/9.3.0
module load openmpi/4.1.4
SCRATCH=/big_scratch
# pull new image to the current directory
singularity pull -F --name ./nwchems_`id -u`.img oras://ghcr.io/edoapra/nwchem-singularity/nwchem-dev.ompi41x:latest
# copy image from current directory to local /big_scratch/ on compute nodes
srun -N $SLURM_NNODES -n $SLURM_NNODES cp ./nwchems_`id -u`.img $SCRATCH/nwchems.img
# basis library files
export SINGULARITYENV_NWCHEM_BASIS_LIBRARY=/cluster/apps/nwchem/nwchem/src/basis/libraries/
# use /big_scratch as scratch_dir
export SINGULARITYENV_SCRATCH_DIR=$SCRATCH
# bind local file system
MYFS=$(findmnt -r -T . | tail -1 |cut -d ' ' -f 1)
# run
srun --mpi=pmi2 -N $SLURM_NNODES -n $SLURM_NPROCS singularity exec --bind $SCRATCH,$MYFS $SCRATCH/nwchems.img nwchem  "file name"
```

30.2.2 Instruction for running on OLCF Frontier

Slurm script for running NWChem Singularity/Apptainer images on OLCF Frontier

```
#!/bin/bash
#SBATCH -t 00:29:00
#SBATCH -A ABCXYZ123
#SBATCH -N 4
#SBATCH  --tasks-per-node 56
#SBATCH  --cpus-per-task 1
#SBATCH -J siosi6
#SBATCH -o siosi6.err.%j
module  load cray-mpich-abi
export APPTAINERENV_LD_LIBRARY_PATH="$CRAY_MPICH_DIR/lib-abi-mpich:$CRAY_MPICH_ROOTDIR/gtl/lib:$CRAY_LD_LIBRARY_PATH:$LD_LIBRARY_PATH:/opt/c
export APPTAINER_CONTAINLIBS="/usr/lib64/libcxi.so.1,/usr/lib64/libjson-c.so.3,/lib64/libtinfo.so.6,/usr/lib64/libnl-3.so.200,/usr/lib64/libgfortran.so.5,/usr/lib64/libjansson.so.4"
MYFS=$(findmnt -r -T . | tail -1 |cut -d ' ' -f 1)
export https_proxy=http://proxy.ccs.ornl.gov:3128/
export BINDS=/usr/share/libdrm,/var/spool/slurmd,/opt/cray,${MYFS}
export FI_CXI_RX_MATCH_MODE=hybrid
export COMEX_EAGER_THRESHOLD=32768
export MPICH_SMP_SINGLE_COPY_MODE=NONE
export OMP_NUM_THREADS=1
export APPTAINERENV_OMP_NUM_THREADS=1
MYIMG=oras://ghcr.io/edoapra/nwchem-singularity/nwchem-dev.mpich3.4.2:
srun apptainer exec --bind $BINDS --workdir `pwd` $MYIMG nwchem siosi6.nw >& siosi6.out.$SLURM_JOBID
```

30.2.3 Instruction for running on NERSC Perlmutter CPU partition

Slurm script for running NWChem Singularity/Apptainer images on the CPU partition ofNERSC Perlmutter

```
#SBATCH -C cpu
#SBATCH -N 8
#SBATCH -A ABC123
#SBATCH --ntasks-per-node=64
#SBATCH --cpus-per-task=2
module load PrgEnv-gnu
module load cray-pmi
module load  cray-mpich-abi
module list
export APPTAINERENV_LD_LIBRARY_PATH="$CRAY_MPICH_DIR/lib-abi-mpich:$CRAY_MPICH_ROOTDIR/gtl/lib:$CRAY_LD_LIBRARY_PATH:$LD_LIBRARY_PATH:/opt/c
export APPTAINER_CONTAINLIBS="/usr/lib64/libcxi.so.1,/usr/lib64/libjson-c.so.3,/lib64/libtinfo.so.6,/usr/lib64/libnl-3.so.200,/usr/lib64/libgfortran.so.5,/usr/lib64/libjansson.so.4"
export OMP_NUM_THREADS=1
export OMP_PROC_BIND=true
export COMEX_MAX_NB_OUTSTANDING=6
export COMEX_EAGER_THRESHOLD=65536
export SCRATCH_DIR=/tmp
export APPTAINER_CONFIGDIR=${SCRATCH}/apptainer
export APPTAINER_CACHEDIR=${SCRATCH}/apptainer/cache
export APP_VER=current
export PATH=/cvmfs/oasis.opensciencegrid.org/mis/apptainer/$APP_VER/x86_64/bin:$PATH
export PATH=/cvmfs/oasis.opensciencegrid.org/mis/apptainer/$APP_VER/x86_64/utils/bin/:$PATH
export LD_LIBRARY_PATH=/cvmfs/oasis.opensciencegrid.org/mis/apptainer/current/x86_64/utils/lib:$LD_LIBRARY_PATH
MYFS=$(findmnt -r -T . | tail -1 |cut -d ' ' -f 1)
export BINDS=/usr/share/libdrm,/var/spool/slurmd,/opt/cray,${MYFS}
export MPICH_GPU_SUPPORT_ENABLED=0
export MPICH_SMP_SINGLE_COPY_MODE=NONE
export FI_CXI_RX_MATCH_MODE=software
export FI_MR_CACHE_MONITOR=disabled
export FI_CXI_RDZV_THRESHOLD=16384
MYIMG=oras://ghcr.io/edoapra/nwchem-singularity/nwchem-dev.mpich3.4.2.libfabric1.21.1:
apptainer pull  /tmp/nwchem.sif $MYIMG
srun  -v  -N  $SLURM_NNODES  -n $SLURM_NPROCS   apptainer exec --bind $BINDS --workdir `pwd`  $MYIMG nwchem siosi6.nw >& siosi6.out.$SLURM_JOBID
```

30.3 Shifter

Instructions for running the NWChem Shifter images are available at https://docs.nersc.gov/applications/nwchem

30.4 Podman

Docker images could be run using podman commands

```
podman run --rm --shm-size 256m --volume /tmp:/data -i -t ghcr.io/nwchemgit/nwchem-dev/amd64 xvdw.nw
```

Made with Material for MkDocs