```
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
```

23.1.1 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.. See section 6.3). 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 $1.084/1.528 \approx 0.709$.

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 Section 23.6.3 for a more complex example.

23.1.2 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.

23.2 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.

23.2.1 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 23.6.2 for an example.

23.2.2 Effective core potentials

If an effective core potential is specified in the usual fashion (see Section 8) 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 Section 23.6.2 for sample input.

23.2.3 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 Section 23.6.4 for another example.

23.3 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).

23.4 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 Section 10.5).

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 (see Section 10.5.1) 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
```

23.5 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.

23.6 Examples

23.6.1 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 Table 23.1

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

Table 23.1: Energies for ONIOM example 1, hydrocarbon bond energy using MP2:HF two-layer model.

The following input first performs a calculation on $CH_3 - CH_2$, 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
  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
  C    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
```

23.6.2 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
```

23.6.3 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 (NH2)2-C=C-(NH2)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
```

23.6.4 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
```

Chapter 24

Hessians

This section relates to the computation of analytic hessians which are available for open and closed shell SCF, except ROHF and for closed shell DFT. 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. This will be changed in the next release.

There is no required input for the Hessian module. This module only impacts the hessian calculation. For options for calculating the frequencies, please see Section 25, the Vibrational module.

24.1 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
```

24.1.1 Defining the wavefunction threshold

You may modify the default threshold for the wavefunction. This keyword is identical to THRESH in the SCF, Section 10.7, and the CONVERGENCE gradient in the DFT, Section 11.6. 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.

24.1.2 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.

24.1.3 Print Control

Known controllable print options are:

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

Table 24.1: Hessian Print Control Specifications

Chapter 25

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 Section 24 for details). The appropriate nuclear hessian generation algorithm is chosen based on the user input when `TASK <theory> 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[1] 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 the geometry input, (c.f., 6) or redefined using the vibrational module input. The default mass is the mass of the most abundant isotope of each element.[2] If the abundance was roughly equal, the mass of the isotope with the longest half life was used.

25.1 Vibrational Module Input

All input for the Vibrational Module is optional since the default definitions will compute the frequencies and IR intensities[3]. 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
  animate [<real> step_size_for_animation]
end
```

[1] Intensities are only computed if the dipole derivatives are available; these are computed by default for most methods that use the finite difference driver routines

[2] c.f., "The Elements" by John Emsley, Oxford University Press, (C) 1989, ISBN 0-19-855237-8.

[3] 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.

25.1.1 Hessian File Reuse

By default the `task <theory> 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.

25.1.2 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
```

25.1.3 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.

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 input directive

```
  animate real <step_size>
```

where `<step_size>` is the real number that is the magnitude of each step along the eigenvector of each nuclear hessian mode in atomic units.

25.1.4 An Example Input Deck

This example input deck will optimize the geometry for the given basis set, compute the frequencies for H_2O, 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; mass H 2.014101779
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
```

Chapter 26

DPLOT

```
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.

26.1 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.

26.2 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.

26.3 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`.

26.4 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.

26.5 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.

26.6 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.

26.7 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).

26.8 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`. 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. α orbitals are always plotted unless `Spin` is set to `beta`.

26.9 Examples

Charge Density

Example of 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 lumo.cube
end
task scf
task dplot
```

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
```

Chapter 27

Electron Transfer Calculations with ET

The NWChem ET module uses the method of *Corresponding Orbital Transformation* to calculate the electron transfer matrix element between ET reactant and product states, V_{RP} (H_{AB} or V_{AB} in the literature). The only input required for the ET module is the name of the files containing the open-shell (UHF or ODFT) MO vectors for the localized 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 magnitude of V_{RP} is a measurement of the degree of coupling between R and P and, therefore, the facility of the electron transfer. It's natural then that V_{RP} is an important component in the expression for the electron transfer rate:

$$k_{ET} = \frac{2\pi}{\hbar} V_{RP}^2 \frac{1}{\sqrt{4\pi\lambda k_B T}} \exp\left(\frac{-\Delta G^*}{k_B T}\right) \tag{27.1}$$

where λ is the nuclear relaxation energy associated with the attachment of the electron and ΔG^* is the free energy activation barrier between the ET reactant and product states (in the simplest case, $\Delta G^* \approx \lambda/4$).

Suggested references are listed below. The first reference gives a good basic description of Marcus' two-state ET model, and the appendix of the second reference details the method used in the ET module.

1. J.R. Bolton, N. Mataga, and G. McLendon in "Electron Transfer in Inorganic, Organic and Biological Systems" (American Chemical Society, Washington, D.C., 1991)

2. A. Farazdel, M. Dupuis, E. Clementi, and A. Aviram, J. Am. Chem. Soc., 112, 4206 (1990).

27.1 VECTORS — input of MO vectors for ET reactant and product states

```
VECTORS [reactants] <string reactants_filename>
```

```
VECTORS [products ] <string products_filename>
```

The `VECTORS` directive allows the user to specify 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.

27.2 FOCK/NOFOCK — method for calculating the two-electron contribution to the ET Hamiltonian

```
<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.

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.

27.3 SWITCH2E — overlap threshold which determines the method for calculating the two-electron contribution to the ET Hamiltonian

```
<SWITCH2E <real switch2e default 1.0d-12>
```

In cases when the ET reactants and products are *very* weakly interacting (small S_{RP}), the two-electron contribution to the electronic coupling should not be calculated with the NWChem Fock matrix routine (see 10.7). When S_{RP} is less than `switch2e` , the `NOFOCK` routine will be used for the two electron contribution to the coupling (see 27.2).

27.4 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}), within the range 1.0d-12 $\rightarrow$ 1.0d-7.

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

```
tol2e 1e-9
```

27.5 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 ET reactant and product wavefunctions reflect *localized states*. This can be accomplished using either a fragment guess (shown in the example, see 10.5.1), or a charged atomic density guess (see 10.5.2). Once the localized orbitals for the ET reactants have been calculated, you can use the reactants' vectors with the `REORDER` keyword to move the electron from the first helium to the second see 27.1).

Example input :

```
#ET reactants:
charge 1
scf
  doublet
  uhf
  vectors input fragment hep.mo he.mo output hea.mo
# hep.mo are the vectors for He(+),
# he.mo  are the vectors for neutral He.
end
task scf

#ET products:
charge 1
scf
  doublet
  uhf
  vectors input hea.mo reorder 2 1 output heb.mo
end
task scf

et
 vectors reactants hea.mo
 vectors products heb.mo
end
task scf et
```

It is important to verify the localization of the electron in the calculation of the vectors `hea.mo` and `heb.mo`. To do this, carefully examine the Mulliken population analysis. For the ET product state in the helium example, the Mulliken population analysis looks like this:

```
   Mulliken analysis of the total density
   -------------------------------------

    Atom       Charge   Shell Charges
 -----------   ------   -------------------------------------------------------
    1 He   2     1.00   0.56  0.44
    2 He   2     2.00   0.78  1.22
```

```
   Atom       Charge   Shell Charges
-----------   ------   -------------------------------------------------------
   1 He   2     1.00   0.56  0.44
   2 He   2     0.00   0.00  0.00
```

The Mulliken population analyses of the total and spin densities show that there is a single electron on the first helium and a pair of electrons on the second helium.

Here is what the output looks like for this example:

```
                          Electron Transfer Calculation
                          -----------------------------

 MO vectors for reactants: hea.mo
 MO vectors for products : heb.mo

 Electronic energy of reactants     H(RR)      -5.3402392824
 Electronic energy of products      H(PP)      -5.3402392824

 Reactants/Products overlap         S(RP)      -0.0006033839

 Reactants/Products interaction energy:
 -------------------------------------
 One-electron contribution         H1(RP)       0.0040314092

 Beginning calculation of 2e contribution
 Two-electron integral screening (tol2e) : 6.03E-11

 Two-electron contribution         H2(RP)      -0.0007837138
 Total interaction energy           H(RP)       0.0032476955

 Electron Transfer Coupling Energy |V(RP)|      0.0000254810
                                                       5.592 cm-1
                                                    0.000693 eV
                                                       0.016 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 spin density 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.

Chapter 28

Properties

```
PROPERTY
  [property name]
  [CENTER ((com || coc || origin || arb <real x y z>) default coc)]
  [VECTORS ...]
END
```

Calculation of properties is accomplished with `TASK PROPERTY` after the completion of an energy (or MP2 gradient) calculation. The following properties can be computed for all wavefunctions that produce orbitals, including Hartree-Fock (closed-shell RHF, open-shell ROHF, and open-shell UHF), DFT (closed-shell and open-shell spin unrestricted), MCSCF (complete active space), and MP2 (closed-shell RHF and open-shell UHF).

- natural bond analysis
- dipole moment
- quadrupole moment
- octupole moment
- Mulliken population analysis and bond order analysis
- electrostatic potential (diamagnetic shielding) at nuclei
- electric field at nuclei
- electric field gradient at nuclei
- electron and spin density at nuclei
- NMR chemical shifts (GIAO method) only for closed-shell RHF

The default molecular orbital file `$file_prefix$.movecs` is used unless a vectors directive (Section 10.5) is provided. It is therefore only necessary to include a vectors directive if the MO vectors to be analyzed are not coming from the default file, e.g., if they have been previously redirected, or if MP2 natural orbitals (file extension `".mp2nos"`) are being anaylzed. The MP2 natural orbitals MUST be used if the user wants MP2 properties.

28.1 Subdirectives

Note that presenting any property input causes all previous property input to be "forgotten", unlike other NWChem modules.

Each property can be requested by means of a subdirective among the subdirectives provided :

- `NBOFILE`
- `DIPOLE`
- `QUADRUPOLE`
- `OCTUPOLE`
- `MULLIKEN`
- `ESP`
- `EFIELD`
- `EFIELDGRAD`
- `ELECTRONDENSITY`
- `GIAO`
- `ALL`

The "`ALL`" keyword generates all currently available properties.

The request `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. To execute the NBO analysis directly, see Section 37.1. All other properties are calculated upon request.

An additional subdirective is provided to specify the origin of the molecular orbitals used in the calculation of the molecular properties. This is the '`VECTORS`' subdirective, also used in the SCF and DFT tasks. For a full description of this subdirective the user is refered to the description found in the SCF description. By default, the input file used for the calculation of the properties has the `.movecs` name extension.

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` (See Section 6.1).

28.1.1 Nbofile

Following the successful completion of an electronic structure calculation, a Natural Bond Orbital (NBO) analysis may be carried out in the following way. On restart specify the `TASK` as `PROPERTY` and supply the sub-directive `NBOFILE` to 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 RESTART directive. The input deck may be edited to provide additional options to the NBO calculation, (see the NBO user's manual for details.) The other option in to directly run the NBO analysis (See Section 37.1 for more information).

Chapter 29

Electrostatic potentials

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
```

29.1 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

  ```
  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

```
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>]
```

29.2 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 **katom+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.

29.3 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>])
```

where **hfree** can be specified to exclude hydrogen atoms from the restaining procecure. Variable **scale** is the strength of the restraint potential, with a default of $0.005au$ for the harmonic restraint and a default value of $0.001au$ for the hyperbolic restraint. For the hyperbolic restraints the tightness **tight** can be specified to change the default value of $0.1e$. 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 $0.001e$.

Chapter 30

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 sequence 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

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.

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.

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.

30.1 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>_(s | x | q | u | t | c) <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.

```
<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_s /msrc/proj/nwchem/share/amber/amber_s/amber99.par
amber_x /msrc/proj/nwchem/share/amber/amber_x/
amber_u /usr/people/username/data/amber/amber_u/
spce /msrc/proj/nwchem/share/solvents/spce.rst
charmm_s /msrc/proj/nwchem/share/charmm/charmm_s/
charmm_x /msrc/proj/nwchem/share/charmm/charmm_x/
```

30.2 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.

30.3 Sequence file generation

If no existing sequence 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 in Table 34.8. 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.

30.4 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 `<integer isgm>:<string name>`, 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> [set <integer mset> | initial | final] \
( dummy | self | uncharged | quantum | quantum_high )
```

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
```

30.5 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 contraint. 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.

30.6 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)
```

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.

```
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).

Chapter 31

Molecular dynamics

31.1 Introduction

31.1.1 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.

31.1.2 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.

31.1.3 Files

File names used have the form `$system$_$calc$.$ext$`, with exception of the topology file (Section 31.1.2), 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

Table 31.1: List of file extensions for nwchem chemical system files.

31.1.4 Databases

Database file used by the **prepare** module are found in directories with name `$ffield$_$level$`, where `$ffield$` is any of the supported force fields (Section 31.1.5). The source of the data is identified by `$level$`, and can be

level	Description	Availability
s	original published data	public
x	additional published data	public
q	contributed data	public
u	user preferred data	private
t	user defined temporary data	private
c	current working data	private

Typically, only the level **s**, **x** and **q** databases are publicly available. The user is responsible for the private level **u**, **t** and **c** database files. When the **prepare** module scans the databases, the priority is **c**>**t**>**u**>**q**>**x**>**s**.

The extension `<ext>` defines the type of database file within each database directory.

frg	fragments
par	parameters
seq	sequences
sgm	segments

Table 31.2: List of database file extensions.

The paths of the different database directories should be defined in a file `.nwchemrc` in a user's home directory, and provides the user the option to select which database files are scanned.

31.1.5 Force fields

Force fields recognized are

Keyword	Force field	Status
`amber`	AMBER99	AMBER95,GLYCAM also available
`charmm`	CHARMM	incomplete

31.2 Format of fragment files

Fragment files contain the basic information needed to specify all interactions that need to be considered in a molecular simulation. The format of the fragment files is described in Table 34.1. 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.

31.3 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 restriction is that bonded interactions may only involve atoms on at most two segments does no longer exist as of **NWChem** release 3.2.1. 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.

The format of a segment is described in Tables 34.2–34.7.

31.4 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. The file format is given in Table 34.8

31.5 Creating topology files

The topology (Section 31.1.2) 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`.

31.6 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.

31.7 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.

31.8 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`.

31.9 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.

31.10 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.

31.11 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.

31.12 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, where <ddacq> is the maximum drift allowed in the ensemble average derivative of the Hamiltonian with respect to $\lambda$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.

31.13 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 forces f^f evaluated using the force field functions to obtain the forces f used to advance the coordinates.

$$f_i = f_i^f + fguide * g_{i-1} \tag{31.1}$$

Variable $tguide$ defines the length of the averaging relative to the timestep Δt.

$$g_i = \frac{\Delta t}{tguide} f_i + \left(1 - \frac{\Delta t}{tguide}\right) g_{i-1} \tag{31.2}$$

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.

31.14 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^2N^{-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.

31.15 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.

31.16 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.

31.17 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.

31.18 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.

31.19 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.

31.20 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.

31.21 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 `idlast`may 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 `idlast`may 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.

31.22 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> [ 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.

31.23 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.

31.24 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.

31.25 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 center of mass 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

31.26 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.

31.27 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

$$\left(1 - factld * \frac{\frac{T_{sync}}{n_p} - t_{sync}^{min}}{t_{wall}}\right)^{\frac{1}{3}} \tag{31.3}$$

where T_{sync} is the accumulated synchronization time of all nodes, n_p is the total number of nodes, t_{sync}^{min} is the synchronization time of the busiest node, and t_{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.

Chapter 32

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
```

32.1 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`.

32.2 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.

32.3 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 ? wild card 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
```

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. Upto 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>]]
```

32.4 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.

32.5 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 `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.

32.6 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
```

32.7 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

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 wild card 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>
```

32.8 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 $\mathrm{mol}^{-1}\mathrm{e}^{-1}$. If no files are specified, only the parameters are set. This analysis applies to solute(s) only.

Chapter 33

Combined quantum and molecular mechanics

Combined or hybrid Quantum Mechanics and Molecular Mechanics (QM/MM) is a simulation methodology that is about 15 years old but in all the literature there are cautions that calibration computations must be done to validate the model for each particular chemical system studied. This is not a black box style computation and the NWChem users are advised that without calibration QM/MM may not give the appropriate results[1].

The QM/MM module in NWChem is driven by the molecular dynamics module. This module currently works for any QM method that has analytic gradients[2]. The input for this requires the definition of chemical system via the same interface that is used by the MD module (c.f. Section 31). The extensions to this interface include the definition of "Quantum" atoms and "Link" where appropriate. The QM information must be present in the traditional NWChem input deck except for the geometry[3]. The geometrical information will be constructed automatically by nwmd. For dynamics and free energy simulations the input is again identical to that for nwmd with limitations on the kinds of simulations that can be done. Link atoms are defined by the molecular dynamics module, based on the specification of the quantum atoms in the prepare module. For the link atoms, basis sets `X_L` need to be defined in the basis set input block.

The QM/MM module is invoked with the task directive where the "theory" is QMMM. The recognized operations on the QM/MM theory directive are energy, optimize, and dynamics.

```
task qmmm (energy | optimize | dynamics)
```

Tasks `gradient`, `saddle`, `frequencies` and `thermodynamics` are currently not available in the QM/MM mode.

The QM/MM input consists of the standard NWChem input block:

```
qmmm
```

[1] c.f., Singh and Kollman, J. Comp. Chem. **7**, 718 (1986); M. J. Field, P. A. Bash and M. Karplus, J. Comp. Chem. **11**, 700, (1990); J. Gao, "Methods and Applications of Combined Quantum Mechanical and Molecular Mechanical Potentials." In *Reviews in Computational Chemistry*; K. B. Lipkowitz, D. B. Boyd, Eds.; VCH Publishers: New York; Vol. 7, pp 119-185 (1995); and M. A. Thompson and G. K. Schenter, J. Phys. Chem **99** 6374 (1995)

[2] The QM/MM method will work with numerical gradients available in NWChem, but it is expected that the performance will not allow any substantive simulations

[3] Any geometry information in the traditional form will be ignored

```
  ...
end
```

The `qmmm` has the following the additional sub-directive that the user may specify for the particular simulation. These options currently are:

- `eatoms < real eatoms>`

  There is one compound input directive that must exist for the QM/MM simulation to proceed. This sets the relative zero of energy for the QM component of the system. It is not incorrect to leave this value as zero but the energetics of the QM system will likely over shadow the MM component of the system. Properties based on energy fluctuations of the system will be overly sensitive to the energy of the QM component of the system. The zero of energy for the MM system is by definition of most parameterized force fields the separated atom energy. The zero of energy for QM systems by definition of most QM methods is the vacuum. The *a priori* determination of the separated atom energy for a particular QM method is not well defined and thus leads to a number of assumptions or guess work depending upon the particular QM method being utilized. Therefor, the determination of the QM separated atom energy ("eatoms") is left to the user. There is no default for this and the input **must** be present for a QM/MM simulation.

- `link hydrogen`

  This directive specified to use hydrogens as link atoms. The default is to use the next group VII atom.

- `nobq ( hydrogen | all | none )`

  This directive specifies that hydrogen atoms, all atoms, or no atoms, respectively, bonded to link atoms will carry a zero charge in the QM part of the calculation.

All other parameters that control the QM/MM simulation are set via the input to nwmd (see chapter 31).

Chapter 34

File formats

Card	Format	Description
I-1-1	a1	$ to identify the start of a fragment
I-1-2	a10	name of the fragment, 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	i5	number of atoms in the fragment
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
		: 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
		4: atom in aromatic ring
II-1-7	i5	charge group
II-1-8	i5	polarization group
II-1-9	f12.6	atomic partial charge
II-1-10	f12.6	atomic polarizability
Any number of cards in deck III to specify complete connectivity		
III-1-1	16i5	connectivity, duplication allowed

Table 34.1: The format of fragment files.

Deck	Format	Description
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	i5	number of atoms in the segment
I-2-2	i5	number of bonds in the segment
I-2-3	i5	number of angles in the segment
I-2-4	i5	number of proper dihedrals in the segment
I-2-5	i5	number of improper dihedrals in the segment

Table 34.2: Segment file format, table 1 of 6.

Deck	Format	Description
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, generic set 1
II-1-4	a1	dynamics type
		: normal
		`D` : dummy atom
		`S` : solute interactions only
		`Q` : quantum atom
		other : intramolecular solute interactions only
II-1-4	a5	atom type, generic set 2
II-1-5	a1	dynamics type
		: normal
		`D` : dummy atom
		`S` : solute interactions only
		`Q` : quantum atom
		other : intramolecular solute interactions only
II-1-6	a5	atom type, generic set 3
II-1-7	a1	dynamics type
		: normal
		`D` : dummy atom
		`S` : solute interactions only
		`Q` : quantum atom
		other : intramolecular solute interactions only
II-1-8	i5	charge group
II-1-9	i5	polarization group
II-1-10	i5	link number
II-1-11	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-2-1	f12.6	atomic partial charge in e, set 1
II-2-2	f12.6	atomic polarizability/$4\pi\epsilon_o$ in nm^3, set 1
II-2-3	f12.6	atomic partial charge in e, set 2
II-2-4	f12.6	atomic polarizability/$4\pi\epsilon_o$ in nm^3, set 2
II-2-5	f12.6	atomic partial charge in e, set 3
II-2-6	f12.6	atomic polarizability/$4\pi\epsilon_o$ in nm^3, set 3

Table 34.3: Segment file format, table 2 of 6.

Deck	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
III-2-1	f12.6	bond length in nm, set 1
III-2-2	e12.5	bond force constant in kJ nm^2 mol^{-1}, set 1
III-2-3	f12.6	bond length in nm, set 2
III-2-4	e12.5	bond force constant in kJ nm^2 mol^{-1}, set 2
III-2-5	f12.6	bond length in nm, set 3
III-2-6	e12.5	bond force constant in kJ nm^2 mol^{-1}, set 3

Table 34.4: Segment file format, table 3 of 6.

Deck	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
IV-2-1	f10.6	angle in radians, set 1
IV-2-2	e12.5	angle force constant in kJ mol^{-1}, set 1
IV-2-3	f10.6	angle in radians, set 2
IV-2-4	e12.5	angle force constant in kJ mol^{-1}, set 2
IV-2-5	f10.6	angle in radians, set 3
IV-2-6	e12.5	angle force constant in kJ mol^{-1}, set 3

Table 34.5: Segment file format, table 4 of 6.

Deck	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: $C\cos(m\phi - \delta)$
V-1-7	i5	proper dihedral parameter origin
		0: from database, next card ignored
		1: from next card
V-2-1	i3	multiplicity, set 1
V-2-2	f10.6	proper dihedral in radians, set 1
V-2-3	e12.5	proper dihedral force constant in kJ mol^{-1}, set 1
V-2-4	i3	multiplicity, set 2
V-2-5	f10.6	proper dihedral in radians, set 2
V-2-6	e12.5	proper dihedral force constant in kJ mol^{-1}, set 2
V-2-7	i3	multiplicity, set 3
V-2-8	f10.6	proper dihedral in radians, set 3
V-2-9	e12.5	proper dihedral force constant in kJ mol^{-1}, set 3

Table 34.6: Segment file format, table 5 of 6.

Deck	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
VI-2-1	3x,f10.6	improper dihedral in radians, set 1
VI-2-2	e12.5	improper dihedral force constant in kJ mol^{-1}, set 1
VI-2-3	3x,f10.6	improper dihedral in radians, set 2
VI-2-4	e12.5	improper dihedral force constant in kJ mol^{-1}, set 2
VI-2-5	3x,f10.6	improper dihedral in radians, set 3
VI-2-6	e12.5	improper dihedral force constant in kJ mol^{-1}, set 3

Table 34.7: Segment file format, table 6 of 6.

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		

Table 34.8: Sequence file format.

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
For each atoms per solvent molecule one card I-3		
I-3-1	a5	solvent name
I-3-2	a5	atom name
I-3-3	6x,i10	solvent atom counter
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
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	f12.6	temperature of frame in K
II-2-3	e12.5	pressure of frame in Pa
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 lxs for solute coordinates
II-4-4	l1	logical lvs for solute velocities
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		
II-5-1	3f8.3	solvent atom coordinates, if lxw or lvw
II-5-4	3f8.3	solvent atom velocities, if lvw
For each solute atom one card II-6 for each atom		
II-6-1	3f8.3	solute atom coordinates, if lxs or lvs
II-6-4	3f8.3	solute atom velocities, if lvs

Table 34.9: Trajectory file format.

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

Table 34.10: Free energy file format.

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

Table 34.11: Root mean square deviation file format.

Card	Format	Description
I-1-1	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

Table 34.12: Property file format.

Chapter 35

Pseudopotential plane-wave density functional theory (NWPW)

The NWChem plane-wave (NWPW) module uses pseudopotentials and plane-wave basis sets to perform Density Functional Theory calculations. This module complements the capabilities of the more traditional Gaussian function based approaches (i.e. DFT module and GAPPS Module) by having an accuracy at least as good as the traditional Gaussian function based approaches 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 prototype (gamma point) projector augmented plane-wave code for calculating molecules, crystals, and surfaces

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 35.1 describes the tasks contained within the PSPW module, section 35.2 describes the tasks contained within the Band module, section 35.3 describes the tasks contained within the PAW module, and section 35.4 describes the pseudopotential library included with NWChem. The datafiles used by the PSPW module are described in section 35.5. Car-Parrinello output data files are described in section 35.5.8, and the minimization and Car-Parrinello algorithms are described in sections ??-35.6. 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 in sections 35.7, 35.8, and 35.11, and 35.12. Finally in section 35.13 the capabilities and limitations of the NWPW module are discussed.

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 in sections 35.7-35.12.

35.1 PSPW Tasks

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 5.10).

```
  TASK PSPW
```

In addition to the directives listed in Section 5.10, 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             ||
           pspw_dplot                 ||
           wannier                    ||
           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:

```
PSPW
  CELL_NAME <string cell_name default 'cell_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>
  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  default Vosko)
  DFT||ODFT||RESTRICTED||UNRESTRICTED
  MULT <integer mult default 1>
  MULLIKEN
  ALLOW_TRANSLATION

  SIMULATION_CELL            ... (see input description) END
  DPLOT                      ... (see input description) END
  WANNIER                    ... (see input description) END
  CAR-PARRINELLO             ... (see input description) END
  PSP_GENERATOR              ... (see input description) END
  WAVEFUNCTION_INITIALIZER   ... (see input description) END
  V_WAVEFUNCTION_INITIATIZER ... (see input description) END
  WAVEFUNCTION_EXPANDER      ... (see input description) END
  STEEPEST_DESCENT           ... (see input description) END
END
```

The following list describes the keywords contained in the PSPW input block.

- <cell_name> - name that points to the the simulation_cell named <cell_name>. See section 35.1.1.
- <input_wavefunctions> - name that points to a 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 (μ). This parameter is not presently used in a conjugate gradient simulation
- <time_step> - value for the time step (Δt). 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.
- <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 $\frac{MIN(|\vec{a_i}|)}{\pi}, i = 1, 2, 3$.
- (Vosko || PBE96) - Choose between Vosko et al's LDA parameterization or the Perdew, Burke, and Ernzerhof GGA functional.
- MULLIKEN - optional keyword which if specified causes a Mulliken anaysis to be performed at the end of the simulation. The angular momentum weights for each kind of atom can be modified using the PSPW ANALYSIS sub-block (see section 35.5.3).
- SIMULATION_CELL (see section 35.1.1)
- DPLOT (see section 35.1.3)
- WANNIER (see section 35.1.4)
- CAR-PARRINELLO(see section 35.1.7)
- PSP_GENERATOR (see section 35.1.8)
- WAVEFUNCTION_INITIALIZER (see section 35.1.9)
- V_WAVEFUNCTION_INITIALIZER (see section 35.1.10)
- WAVEFUNCTION_EXPANDER (see section 35.1.11).
- STEEPEST_DESCENT (see section 35.1.12)

A prototype limited memory BFGS (LMBFGS) minimizer can be used to minimize the energy. To use this new optimizer 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 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. # Default - initial wavefunctions generated using an LCAO guess.
set nwpw:lcao_skip .true.  # Initial wavefunctions generated using a random plane-wave guess.

set nwpw:lcao_print .false. # Default - Ouput 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
```

35.1.1 Simulation Cell

Simulation cells are stored in the RTDB. To enter a simulation cell into the RTDB the user defines a simulation_cell sub-block within the PSPW block. Listed below is the format of a simulation_cell sub-block.

```
PSPW
...
   SIMULATION_CELL
      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 boundry 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.

35.1.2 (

`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.
```

35.1.3 DPLOT

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.

```
PSPW
  ...
  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.

```
PSPW
...
   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 <integer orbital_number no default> <string orbital_name no default>
   END
...
END
```

The following list describes the input for the DPLOT sub-block.

- <input_wavefunctions> - name of pspw wavefunction file.
- <density_name> - filename containing Gaussian Cube file of density plot.
- <elf_name> - filename containing Gaussian Cube file of and ELF plot
- <orbital_name> - filename containing Gaussian Cube file of orbital plot.

35.1.4 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. The current version of this code works only for cubic cells.

Input to the Wannier task is contained within the Wannier sub-block.

```
PSPW
  ...
  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.

```
PSPW
...
   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.

35.1.5 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>
```

35.1.6 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 pspw_APC:Gc <real Gc_cutoff>  # specifies the maximum frequency component of the density to be used in the fitting in units of au.

set pspw_APC:nga <integer number_gauss> # specifies the the number of Gaussian functions per
atom.

set pspw_APC:gamma <real gamma_list> # specifies the decay lengths of each atom centered Gaussian.
```

We suggest using the following parameters.

```
set pspw_APC:Gc 2.5
set pspw_APC:nga 3
set pspw_APC:gamma 0.6 0.9 1.35
```

35.1.7 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 35.6.

Input to the Car-Parrinello simulation is contained within the Car-Parrinello sub-block.

```
PSPW
  ...
  Car-Parrinello
     ...
  END
  ...
END
```

To run a Car-Parrinello calculation the following directive is used:

```
TASK PSPW 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.

```
PSPW
...
   Car-Parrinello
```

```
      CELL_NAME <string cell_name default 'cell_default'>
      INPUT_WAVEFUNCTION_FILENAME    <string input_wavefunctions    default input_movecs>
      OUTPUT_WAVEFUNCTION_FILENAME   <string output_wavefunctions   default input_movecs>
      INPUT_V_WAVEFUNCTION_FILENAME  <string input_v_wavefunctions  default input_vmovecs>
      OUTPUT_V_WAVEFUNCTION_FILENAME <string output_v_wavefunctions default input_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 || PBE96  default Vosko)
      [Nose-Hoover <real Period_electron Temperature_electrion Period_ion Temperature_ion
                    default 100.0 298.15 100.0 298.15>]
      [SA_decay <real sa_scale_c sa_scale_r default 1.0 1.0>]
      XYZ_FILENAME <string xyz_filename default XYZ>
      EMOTION_FILENAME <string emotion_filename default EMOTION>
      HMOTION_FILENAME <string hmotion_filename default HMOTION>
      OMOTION_FILENAME <string omotion_filename default OMOTION>
      EIGMOTION_FILENAME <string eigmotion_filename default EIGMOTION>
      ION_MOTION_FILENAME <string ion_motion_filename default MOTION>

   END
...

END
```

The following list describes the input for the Car-Parrinello sub-block.

- <cell_name> - name that points to the the simulation_cell named <cell_name>. See section 35.1.1.
- <input_wavefunctions> - name that points to a file containing one-electron orbitals
- <output_wavefunctions> - name that will point to file containing the one-electron orbitals at the end of the run.
- <input_v_wavefunctions> - name that points to a file containing one-electron orbital velocities.
- <output_v_wavefunctions> - name that will point to file containing the one-electron orbital velocities at the end of the run.
- <fake_mass> - value for the electronic fake mass (μ).
- <time_step> - value for the Verlet integration time step (Δt).
- <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 $\frac{MIN(|\vec{a_i}|)}{\pi}, i = 1, 2, 3$.
- (Vosko || PBE96) - Choose between Vosko et al's LDA parameterization or the Perdew, Burke, and Erzherhoff GGA functional.
- Nose-Hoover - optional subblock which if specified causes the simulation to perform Nose-Hoover dynamics. If this subblock is not specified the simulation performs constant energy dyanmics. See section 35.6.2 for a description of the parameters.
  - <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
- 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 35.6.2 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 that points to the XYZ motion file generated
- <emotion_filename> - name that points to the emotion motion file generated. See section 35.5.8 for a description of the datafile.
- <hmotion_filename> - name that points to the hmotion motion file generated. See section 35.5.8 for a description of the datafile.
- <eigmotion_filename> - name that points to the eigmotion motion file generated. See section 35.5.8 for a description of the datafile.
- <ion_motion_filename> - name that points to the ion_motion motion file generated. See section 35.5.8 for a description of the datafile.
- MULLIKEN - optional keyword which if specified causes an omotion motion file to be created. For this option to work angular momentum weights for each kind of atom have to be entered into the RTDB using the PSPW ANALYSIS sub-block (see section 35.5.3.
- <omotion_filename> - name that points to the omotion motion file generated. See section 35.5.8 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 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
```

35.1.8 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 35.4) 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.

```
PSPW
  ...
  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.

```
PSPW
...
   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 || SCRHODINGER 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)

ATOMIC_FILLING Block

This required block is used to define the reference atom which is used to define the pseudopotential. After the ATOMIC_FILLING: <ncore> <nvalence> 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.

`CUTOFF` Block

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 cufoff raddi 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
```

`SEMICORE_RADIUS` Option

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 implmentation in the PSPW module defines the semi-core denisty, $\rho_{semicore}$ in terms of the core density, ρ_{core}, by using the sixth-order polynomial

$$\rho_{semicore}(r) = \begin{cases} \rho_{core} & \text{if } r \geq r_{semicore} \\ c_0 + c_3 r^3 + c_4 r^4 + c_5 r^5 + c_6 r^6 & \text{if } r < r_{semicore} \end{cases} \tag{35.1}$$

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.

35.1.9 `WAVEFUNCTION_INTITIALIZER`

The functionality of this task is now performed automatically. For backwards compatibility, we provide a description of the input to this task.

The wavefunction_initializer task is used to generate an initial wavefunction datafile. Input to the WAVEFUNCTION_INITIALIZER task is contained within the WAVEFUNCTION_INITIALIZER sub-block.

```
PSPW
  ...
  WAVEFUNCTION_INITIALIZER
     ...
  END
  ...
END
```

To run a WAVEFUNCTION_INITIALIZER calculation the following directive is used:

```
TASK PSPW WAVEFUNCTION_INITIALIZER
```

Listed below is the format of a WAVEFUNCTION_INITIALIZER sub-block.

```
PSPW
...
   WAVEFUNCTION_INITIALIZER
     CELL_NAME: <string cell_name>
     WAVEFUNCTION_FILENAME: <string wavefunction_name default input_movecs>
     (RESTRICTED||UNRESTRICTED)
     if (RESTRICTED)
        RESTRICTED_ELECTRONS: <integer restricted electrons>
     if (UNRESTRICTED)
        UP_ELECTRONS: <integer up_electrons>
        DOWN_ELECTRONS: <integer down_electrons>
   END
...
END
```

The following list describes the input for the WAVEFUNCTION_INITIALIZER sub-block.

- <cell_name> - name that points to the simulation_cell named <cell_name>. See section 35.1.1.
- <wavefunction_name> - name that will point to a wavefunction file.
- RESTRICTED - keyword specifying that the calculation is restricted.
- UNRESTRICTED - keyword specifying that the calculation is unrestricted.
- <restricted_electrons> - number of restricted electrons. Not used if an UNRESTRICTED calculation.
- <up_electrons> - number of spin-up electrons. Not used if a RESTRICTED calculation.
- <down_electrons> - number of spin-down electrons. Not used if a RESTRICTED calculation.

Old Style Input (version 3.3) to WAVEFUNCTION_INTITIALIZER

For backwards compatibility, the input to the WAVEFUNCTION_INITIALIZER sub-block can also be of the form

```
PSPW
...
   WAVEFUNCTION_INITIALIZER
     CELL_NAME: <string cell_name>
     WAVEFUNCTION_FILENAME: <string wavefunction_name default input_movecs>
     (RESTRICTED||UNRESTRICTED)

     [UP_FILLING: <integer up_filling>
        [0 0 0   0]
        {<integer kx ky kz> (-2||-1||1||2)}]
     [DOWN_FILLING: <integer down_filling>
        [0 0 0   0]
        {<integer kx ky kz> (-2||-1||1||2)}]
   END
...
END
```

where

- <cell_name> - name that points to the simulation_cell named <cell_name>. See section 35.1.1.
- <wavefunction_name> - name that will point to a wavefunction file.
- RESTRICTED - keyword specifying that the calculation is restricted.
- UNRESTRICTED - keyword specifying that the calculation is unrestricted.
- <up_filling> - number of restricted molecular orbitals if RESTRICTED and number of spin-up molecular orbitals if UNRESTRICTED.
- <down_filling> - number of spin-down moleclar orbitals if UNRESTRICTED. Not used if a RESTRICTED calculation.
- <kx ky kz> - specifies which planewave is to be filled.

The values for the planewave $(-2||-1||1||2)$ are used to represent whether the specified planewave is a cosine or a sine function, in addition random noise can be added to these base functions. That is $+1$ represents a cosine function, and -1 represents a sine function. The $+2$ and -2 values are used to represent a cosine function with random components added and a sine function with random components added respectively.

35.1.10 V_WAVEFUNCTION_INITIALIZER

The functionality of this task is now performed automatically. For backwards compatibility, we provide a description of the input to this task.

The v_wavefunction_initializer task is used to generate an initial velocity wavefunction datafile. Input to the V_WAVEFUNCTION_INITIALIZER task is contained within the V_WAVEFUNCTION_INITIALIZER sub-block.

```
PSPW
  ...
  V_WAVEFUNCTION_INITIALIZER
```

```
    ...
  END
  ...
END
```

To run a V_WAVEFUNCTION_INITIALIZER calculation the following directive is used:

```
TASK PSPW WAVEFUNCTION_INITIALIZER
```

Listed below is the format of a V_WAVEFUNCTION_INITIALIZER sub-block.

```
PSPW
...
   V_WAVEFUNCTION_INITIALIZER
     V_WAVEFUNCTION_FILENAME: <string v_wavefunction_name default input_vmovecs>
     CELL_NAME: <string cell_name>
     (RESTRICTED||UNRESTRICTED)
     UP_FILLING: <integer up_filling>
     DOWN_FILLING: <integer down_filling>
   END
...
END
```

The following list describes the input for the V_WAVEFUNCTION_INITIALIZER sub-block.

- <cell_name> - name that points to the simulation_cell named <cell_name>. See section 35.1.1.
- <wavefunction_name> - name that will point to a velocity wavefunction file.
- RESTRICTED - keyword specifying that the calculation is restricted.
- UNRESTRICTED - keyword specifying that the calculation is unrestricted.
- <up_filling> - number of restricted velocity molecular orbitals if RESTRICTED and number of spin-up velocity molecular orbitals if UNRESTRICTED.
- <down_filling> - number of spin-down velocity moleclar orbitals if UNRESTRICTED. Not used if a RESTRICTED calculation.

35.1.11 WAVEFUNCTION_EXPANDER

The wavefunction_expander task is used to convert a new wavefunction file that spans a larger grid space from an old wavefunction file. Input to the WAVEFUNCTION_EXPANDER task is contained within the WAVEFUNCTION_EXPANDER sub-block.

```
PSPW
  ...
  WAVEFUNCTION_EXPANDER
    ...
  END
  ...
END
```

To run a WAVEFUNCTION_EXPANDER calculation the following directive is used:

```
TASK PSPW WAVEFUNCTION_EXPANDER
```

Listed below is the format of a WAVEFUNCTION_EXPANDER sub-block.

```
PSPW
...
   WAVEFUNCTION_EXPANDER
     OLD_WAVEFUNCTION_FILENAME: <string old_wavefunction_name default input_movecs>
     NEW_WAVEFUNCTION_FILENAME: <string new_wavefunction_name default input_movecs>
     NEW_NGRID: <integer na1 na2 na3>

   END
...
END
```

The following list describes the input for the WAVEFUNCTION_EXPANDER sub-block.

- <old_wavefunction_name> - name that points to a wavefunction file.
- <new_wavefunction_name> - name that will point to a wavefunction file.
- <na1 na2 na3> - number of grid points in each dimension for the new wavefunction file.

35.1.12 STEEPEST_DESCENT

The functionality of this task is now performed automatically by the PSPW minimizer. For backwards 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. A detailed description of the this method is described in section ??

Input to the steepest_descent simulation is contained within the steepest_descent sub-block.

```
PSPW
  ...
  STEEPEST_DESCENT
     ...
  END
  ...
END
```

To run a steepest_descent calculation the following directive is used:

```
TASK PSPW 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.

```
PSPW
...
   STEEPEST_DESCENT
      CELL_NAME <string cell_name>
      [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 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 || PBE96  default Vosko)
      [MULLIKEN]

   END
...

END
```

The following list describes the input for the STEEPEST_DESCENT sub-block.

- <cell_name> - name that points to the the simulation_cell named <cell_name>. See section 35.1.1.
- GEOMETRY_OPTIMIZE - optional keyword which if specified turns on geometry optimization.
- <input_wavefunctions> - name that points to a 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 (μ).
- <time_step> - value for the time step (Δt).
- <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 $\frac{MIN(|\vec{a_i}|)}{\pi}, i = 1, 2, 3$.
- (Vosko || PBE96) - Choose between Vosko et al's LDA parameterization or the Perdew, Burke, and Erzherhoff GGA functional.
- MULLIKEN - optional keyword which if specified causes a Mulliken anaysis to be performed at the end of the simulation. For this option to work angular momentum weights for each kind of atom have to be entered into the RTDB using the PSPW ANALYSIS sub-block (see section 35.5.3.

35.2 Band Tasks

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 5.10).

```
  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>
  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)>
  EXCHANGE_CORRELATION: (Vosko || PBE96  default Vosko)
  DFT||ODFT||RESTRICTED||UNRESTRICTED
  MULT <integer mult default 1>

  SIMULATION_CELL ... (see input description) END
  BRILLOUIN_ZONE  ... (see input description) END

END
```

The following list describes these keywords.

- <cell_name> - name that points to the the simulation_cell named <cell_name>. See section 35.1.1.
- <input_wavefunctions> - name that points to a 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 (μ). This parameter is not presently used in a conjugate gradient simulation
- <time_step> - value for the time step (Δt). 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.
- <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 $\frac{MIN(|\vec{a_i}|)}{\pi}, i = 1, 2, 3$.
- (Vosko || PBE96) - Choose between Vosko et al's LDA parameterization or the Perdew, Burke,
- SIMULATION_CELL (see section 35.1.1)
- BRILLOUIN_ZONE (see section 35.2.1)

35.2.1 Brillouin Zone

The special points of the Brillouin zone for a Band structure calculation are stored in the RTDB. To enter special points of the Brillouin zone into the RTDB 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.

35.3 PAW Tasks

All input to the PAW Tasks is contained within the compound NWPW block,

```
NWPW
   ...
END
```

To perform an actual calculation the following is used (Section 5.10).

```
  TASK PAW 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>
  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)>
  EXCHANGE_CORRELATION: (Vosko || PBE96  default Vosko)
  DFT||ODFT||RESTRICTED||UNRESTRICTED
  MULT <integer mult default 1>

  SIMULATION_CELL ... (see input description) END

END
```

The following list describes these keywords.

- <cell_name> - name that points to the the simulation_cell named <cell_name>. The current version of PAW only accepts periodic unit cells. See section 35.1.1.
- GEOMETRY_OPTIMIZE - optional keyword which if specified turns on geometry optimization.
- <input_wavefunctions> - name that points to a 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 (μ). This parameter is not presently used in a conjugate gradient simulation
- <time_step> - value for the time step (Δt). 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.
- <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 smooth compensation summation.
- <rcut> - value for the cutoff radius used in the smooth compensation summation.
  Default set to be $\frac{MIN(|\vec{a_i}|)}{\pi}, i = 1, 2, 3$.
- (Vosko || PBE96) - Choose between Vosko et al's LDA parameterization or the Perdew, Burke,
- SIMULATION_CELL (see section 35.1.1)

35.4 Pseudopotential and PAW basis Libraries

A library of pseudopotentials used by PSPW and BAND is currently available in the directory
`$NWCHEM_TOP/src/nwpw/libraryps/library1`

The elements listed in the following table are present:

```
 H
-------
 Li Be                                 B  C  N  O
-------                               ------------------
                                       Al Si P  S
------------------------------------------------------
                                    Zn Ga
------------------------------------------------------
```

Similary, a library of PAW basis used by PAW is currently available in the directory
`$NWCHEM_TOP/src/nwpw/libraryps/library2`

Currently there are not very many elements available for PAW. However, the user can request additional basis sets from Eric J. Bylaska at (nwchem-support@emsl.pnl.gov or eric.bylaska@pnl.gov) The elements available are: H, O Al, Sc, V, and Fe.

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.

35.5 NWPW RTDB Entries and 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.

35.5.1 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.

35.5.2 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.

35.5.3 `ANALYSIS`: Mulliken RTDB data

To perform Mulliken analysis information is needed from one-dimensional pseudopotential files. In-order to facilitate the transfer of this information to the simulation the ANALYSIS sub-block is used to exract the necessary information and put it into the RTDB.

```
PSPW
...
   ANALYSIS
      (psp_filename: <string psp_name> \
       ...)
   END
...
END
```

Basically, the user needs to enter each pseudopotential used in the simulation.

- <psp_name> - name that ponts to a one-dimensional pseudopotential file.

35.5.4 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.

35.5.5 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 used by the Car-Parrinello task and can be generated using the v_wavefunction_initializer task.

35.5.6 Formatted Pseudopotential Datafile

The pseudopotentials in Kleinman-Bylander form expanded on a simulation cell (3d grid) are stored in a formatted pseudopotential 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 pseudpotential_formatter task.

35.5.7 One-Dimensional Pseudopotential Datafile

The one-dimensional pseudopotentials are stored in a one-dimensional pseudopotential file. This is an ascii file and can be directly edited with a text editor. However, the user will usually use the psp_generator task to generate this datafile.

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
```

35.5.8 PSPW Car-Parrinello Output Datafiles

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:  ]  ....
```

ION_MOTION motion file

Datafile that stores ion positions and velocities as a function of time

```
[line 1:         ]  it_out, n_ion, omega
[line 2:         ]  time
do ii=1,n_ion
[line 2+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: ] x(ii),y(ii),z(ii), vx(ii),vy(ii),vz(ii)
end do
[line 2*n_ion+4:  ]  ....
```

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)
[line 2:         ] ...
```

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):  ] ....
```

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:         ] ...
```

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.

35.6 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 ficticious motion for the Kohn-Sham orbitals of density-functional theory (R. Car and M. Parrinello, Phys. Rev. Lett. **55**, 2471, (1985)). At the heart of this method they introduced a ficticious kinetic energy functional for the Kohn-Sham orbitals.

$$KE(\{\psi_{i,\sigma}(\vec{r})\}) = \sum_{i,\sigma}^{occ} \int d\vec{r}\; \mu \left| \dot{\psi}_{i,\sigma}(\vec{r}) \right|^2 \tag{35.2}$$

Given this kinetic energy the constrained equations of motion are found by taking the first variation of the auxiliary Lagrangian.

$$\begin{aligned} L &= \sum_{i,\sigma}^{occ} \int d\vec{r}\; \mu \left| \dot{\psi}_{i,\sigma}(\vec{r}) \right|^2 + \frac{1}{2} \sum_I M_I \left| \dot{\vec{R}}_I \right|^2 - E\left[\{\psi_{i,\sigma}(\vec{r})\}, \left\{ \vec{R}_I \right\} \right] \\ &\quad + \sum_{ij,\sigma} \Lambda_{ij,\sigma} \left(\int d\vec{r}\; \psi^*_{i,\sigma}(\vec{r}) \psi_{j,\sigma}(\vec{r}) - \delta_{ij,\sigma} \right) \end{aligned} \tag{35.3}$$

Which generates a dynamics for the wavefunctions $\psi_{i,\sigma}(\vec{r})$ and atoms positions $\vec{R}_I$ through the constrained equations of motion:

$$\mu\ddot{\psi}_{i,\sigma}(\vec{r},t) = -\frac{\delta E}{\delta\psi^*_{i,\sigma}(\vec{r},t)} + \sum_j \Lambda_{ij,\sigma}\psi_{j,\sigma}(\vec{r},t) \tag{35.4}$$

$$M_I\ddot{\vec{R}}_I = -\frac{\partial E}{\partial\vec{R}_I} \tag{35.5}$$

where μ is the fictitious mass for the electronic degrees of freedom and M_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 Eqs. 35.4-35.5.

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.

35.6.1 Verlet Algorithm for Integration

Eqs. 35.4-35.5 integrated using the Verlet algorithm results in

$$\psi^{t+\Delta t}_{i,\sigma} \leftarrow 2\psi^{t}_{i,\sigma} - \psi^{t-\Delta t}_{i,\sigma} + \frac{(\Delta t)^2}{\mu}\left[\frac{\delta E}{\delta\psi^*_{i,\sigma}} + \sum_j \psi_{j,\sigma}\Lambda_{ji,\sigma}\right]_t \tag{35.6}$$

$$\vec{R}^{t+\Delta t}_I \leftarrow 2\vec{R}^t_I - \vec{R}^{t-\Delta t}_I + \frac{(\Delta t)^2}{M_I}\frac{\partial E}{\partial\vec{R}_I} \tag{35.7}$$

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

$$\begin{aligned}\frac{\delta E}{\delta\psi^*_{i,\sigma}} &= -\frac{1}{2}\nabla^2\psi_{i,\sigma}(\vec{r}) \\ &+ \int d\vec{r'}W_{ext}(\vec{r},\vec{r'})\psi_{i,\sigma}(\vec{r'})\end{aligned}$$

$$
\begin{aligned}
&+ \int d\vec{r'} \frac{n(\vec{r'})}{|\vec{r}-\vec{r'}|}\psi_{i,\sigma}(\vec{r}) \\
&+ \mu_{xc}^{\sigma}(\vec{r})\psi_{i,\sigma}(\vec{r}) \\
&\equiv \hat{H}\psi_{i,\sigma}
\end{aligned} \tag{35.8}
$$

To find the matrix $\Lambda_{ij,\sigma}$ we impose the orthonormality constraint on $\psi_{i,\sigma}^{t+\Delta t}$ to obtain a matrix Riccatti equation, and then Riccatti equation is solved by an iterative solution (see section ??).

35.6.2 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 auxillary energy functionals to the total energy.

$$
ION\_THERMOSTAT(x_R) = \frac{1}{2}Q_R\dot{x_R} + E_{R0}x_R \tag{35.9}
$$

$$
ELECTRON\_THERMOSTAT(x_e) = \frac{1}{2}Q_e\dot{x_e} + E_{e0}x_e \tag{35.10}
$$

In these equations, the average kinetic energy for the ions is

$$
E_{R0} = \frac{1}{2}fk_BT \tag{35.11}
$$

where f is the number of atomic degrees of freedom, k_B is Boltzmans constant, and T is the desired temperature. Defining the average ficticious kinetic energy of the electrons is not as straighforward. Blöchl and Parrinello (P.E. Blöchl and M. Parrinello, Phys. Rev. B, **45**, 9413, (1992)) have suggested the following formula for determining the average ficticious kinetic energy

$$
E_{e0} = 4k_BT\frac{\mu}{M}\sum_i < \psi_i| - \frac{1}{2}\nabla^2|\psi_i > \tag{35.12}
$$

where μ is the ficticious 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.

$$
P_{ion} = 2\pi\sqrt{\frac{Q_R}{4E_{R0}}} \tag{35.13}
$$

$$
P_{electron} = 2\pi\sqrt{\frac{Q_e}{4E_{e0}}} \tag{35.14}
$$

where P_{ion} and $P_{electron}$ are the periods of oscillation for the ionic and ficatious electronic thermostats.

In simulated annealing simulations the electronic and ionic Temperatures are scaled according to an exponential cooling schedule,

$$
T_e(t) = T_e^0 \exp^{-\frac{t}{\tau_e}} \tag{35.15}
$$

$$
T_{ionic}(t) = T_{ionic}^0 \exp^{-\frac{t}{\tau_{ionic}}} \tag{35.16}
$$

where T_e^0 and T_{ionic}^0 are the intial temperatures, and τ_e and τ_{ionic} are the cooling rates in atomic units.

35.7 PSPW Tutorial 1: Minimizing the geometry for a C_2 molecule

In this section we show how use the PSPW module to optimize the geometry for a C_2 molecule at the PBE96 levels.

In the following example we show the input needed to optimize the geometry 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 PBE96, 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).

```
start c2_pspw_pbe96
title "C2 restricted singlet dimer optimization - PBE96/44Ry"

geometry
C    -0.62 0.0 0.0
C     0.62 0.0 0.0
end

pspw
   simulation_cell units angstroms
      boundary_conditions aperiodic
      SC 10.0
      ngrid 40 40 40
   end
   xc pbe96
end
set nwpw:minimizer 2
task pspw optimize
```

35.8 PSPW Tutorial 2: Running a Car-Parrinello Simulation

In this section we show how use the PSPW module to perform a Car-Parrinello molecular dynamic simulation for a C_2 molecule at the LDA level. 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). 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_pspw_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 pspw energy
task pspw Car-Parrinello
```

35.9 PSPW Tutorial 3: Running a Simulated Annealing Car-Parrinello Simulation

35.10 PSPW Tutorial 4: optimizing a unit cell and geometry for Silicon-Carbide

The following example demonstrates how to uses the PSPW 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

task pspw optimize
```

35.11 Band Tutorial 1: Minimizing the energy of a silicon-carbide crystal by running a PSPW and Band simulation in tandem

The following input deck performs a PSPW energy calculation followed by a Band energy calculation at the Γ-point for a cubic (8-atom) silicon-carbide crystal. Since the geometry is entered using fractional coordinates the unit cell parameters do not have to be re-specified in the simulation_cell nwpw sub-block. In this example, default pseudopotential from the pseudopotential library are used for C and Si. The advantage of running these calculations in tandem is that the Band code uses the wavefunctions generated from the faster PSPW calculation for its initial guess. The PSPW energy is -38.353570, and the Band energy is -38.353570.

```
start SiC_band
title "SiC 8 atom cubic cell"

#**** geometry entered 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
   brillouin_zone
     kvector  0.0 0.0 0.0
   end
   ewald_ncut 8
end
set nwpw:minimizer 2
set nwpw:psi_brillioun_check .false.
task pspw energy
task band energy
```

35.12 PAW Tutorial

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 paramters 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 steepest_descent

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
```

35.13 NWPW Capabilities and Limitations

- You cannot use more processors than the size of the third dimension (e.g. a 64x64x64 FFT grid can use at most 64 processors).
- The second and third dimensions of the FFT grid must be the same (i.e. the parameters na2 and na3 must be the same for each simulation cell).
- Wannier orbital generation only works with cubic unit cells ($\alpha = \beta = \gamma = 90^o$)
- A default Brillouin zone has not been defined for Band simulations.
- PAW only works with units cells that have periodic boundary conditions.
- PAW is not interfaced to driver, stepper, and vib modules.

35.14 Questions and Difficulties

Questions and encountered problems should be reported to nwchem-support@emsl.pnl.gov or to Eric J. Bylaska, Eric.Bylaska@pnl.gov

Chapter 36

Controlling NWChem with Python

Python (version 1.5.1) 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 http://www.python.org for a full manual and lots of useful code and resources.

36.1 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 36.3.11).

36.2 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 36.3.9).

- `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 (Section 36.3.10) explains, in lieu of a Python wrapper for the geometry object, how to obtain the Cartesian molecular coordinates directly from the database.

36.3 Examples

Several examples will provide the best explanation of how the extensions are used, and how Python might prove useful.

36.3.1 Hello world

```
python
  print 'Hello world from process ', ga_nodeid()
end

task python
```

This input prints the traditional greeting from each parallel process.

36.3.2 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 = ', exponent, ' energy = ', 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.

36.3.3 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

print none

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 = ', exponent, ' energy = ', energy
        plotdata.write('%f %f\n' % (exponent , energy))
     exponent = exponent + 0.1

  if (ga_nodeid() == 0): plotdata.close()
end
```

```
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.

36.3.4 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.

36.3.5 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 noprint
  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.

36.3.6 Scan the geometry and compute the energy and gradient

```
basis noprint; H library sto-3g; O library sto-3g; end

python noprint
  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.

36.3.7 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.

36.3.8 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.

36.3.9 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 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.

36.3.10 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 (see Section 6) 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.

36.3.11 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 = 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 example from Section 36.3.3 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 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.

36.4 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.

Chapter 37

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 to the `nwchem-support@emsl.pnl.gov` e-mail list.

37.1 NBO — Natural Bond Orbital Analysis

```
NBO
  ...
END
```

This directive is used to run the NBO package within NWChem. (To only print out an input file for NBO, see Section 28.1.1.) The current version of NBO in NWChem is version 5.0.

Inside the NBO block are the typical commands that would be needed for NBO (Please see an NBO user's manual for more information.). The following directive is needed to execute NBO.

```
task nbo
```

As an example:

```
title "Methylamine...rhf/3-21g//Pople-Gordon standard geometry"

start methylamine

echo

memory 8 mw

basis
 C library 3-21g
 N library 3-21g
 H library 3-21g
end
```

```
geometry
 C           .052902     .711852     .000000
 N           .052902    -.758148     .000000
 H          -.974760    1.075185     .000000
 H           .566733    1.075185     .889981
 H           .566733    1.075185    -.889981
 H          -.423217   -1.094815     .824662
 H          -.423217   -1.094815    -.824662
 symmetry c1
end

task SCF energy

nbo
 $nbo cmo $end
end

task nbo
```

37.2 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 from Section 1:

> 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.

37.2.1 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.

37.2.2 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).

37.2.3 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
```

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 (see Section 6). `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 preverse 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.

Basis specification

The basis name on the theory or sptheory directive 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 sptheory defaults to the theory basis which defaults to `"ao basis"`.

Effective core potentials

If an effective core potential is specified in the usual fashion (see Section 8) 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.

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 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.

POLYRATE related options

These keyword options are simlar to the POLYRATE input format, except there are no ENERGETICS, OPTIMIZATION, SECOND, TUNNELING, and RATE sections.

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.

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

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.

Restart

`DIRDYVTST` calculations should be restarted through the normal NWChem mechanism (See Section 5.1). 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.

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
```

Chapter 38

Acknowledgments

This work was supported by funds from the Environmental and Molecular Sciences Laboratory Construction Project at Pacific Northwest National Laboratory. Development of some of the parallel programming tools and algorithms employed by NWChem was performed under the auspices of the High Performance Computing and Communications Program of the Mathematical, Information, and Computational Sciences Division, U.S. Department of Energy. Pacific Northwest National Laboratory is operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract DE-AC06-76RLO 1830.

Appendix A

Standard Basis Sets

Basis sets and effective core potentials were obtained (1/1/2002) from the Extensible Computational Chemistry Environment (ECCE) Basis Set Database, as developed and distributed by the Molecular Science Computing Facility, Environmental and Molecular Sciences Laboratory which is part of the Pacific Northwest National Laboratory, P.O. Box 999, Richland, Washington 99352, USA, and is funded by the U.S. Department of Energy. The Pacific Northwest National Laboratory is a multi-program laboratory operated by Battelle Memorial Institute for the U.S. Department of Energy under contract DE-AC06-76RLO. Contact David Feller (`df_feller@pnl.gov`) or Deborah Gracio (`gracio@pnl.gov`) for further information.

The names in the NWChem library are consistent with those in the ECCE database and thus may include spaces. The standard NWChem input routines require that strings including spaces are enclosed in quotation marks (`"..."`) or that blanks are escaped with a backslash. As a convenience, basis set names may also have the blanks replaced with underscores. Thus, the following all yield the same basis set for oxygen

```
oxygen library "DZP + Diffuse (Dunning)"
oxygen library DZP\ +\ Diffuse\ (Dunning)
oxygen library DZP_+_Diffuse_(Dunning)
```

Case may be ignored when specifying basis set names, but otherwise names should be specified exactly as provided below. A good method is just to cut/paste from the WWW pages since they were generated electronically from the library source.

Errors found in the basis set library of NWChem version 4.0 have been corrected in the current library of NWChem version 4.1. The changes are listed below:

1. Basis Set `"aug-cc-pVTZ"`
   Element: Br
   Correction of the diffuse s-exponent from 0.041601 to 0.045593.

2. Basis Set `"cc-pV5Z"`
   Element: Ar
   Questionable contraction coefficients were replaced with HF AO's.

3. Basis Set `"Stuttgart RSC 1997 ECP"`
   Element: Ba
   This basis was called Stuttgart RSC ECP in version 4.0. The 4th s- and p-primitive are now uncontracted.

4. ECP `"LANL2DZ ECP"`
   Element: K and Ca
   The ECP was defined with a core of 18 electrons, inconsistent with the other elements in this series (3d has 10 electron core). The ECP has been replaced with the Hay-Wadt ECP used in Hay-Wadt VDZ.

IMPORTANT NOTE: The Stuttgart basis set and ECP developers have recently developed a new set of Stuttgart RSC ECP's for the Lanthanide series of elements. This new set is now called "Stuttgart RSC ECP", while the previous, and most widely used, set has been renamed to "Stuttgart RSC 1997 ECP" in both the EMSL Basis Set library and in the NWChem basis set library.

Relativistic contractions of standard basis sets for use in the Douglas-Kroll and Dyall-modified-Dirac method have also been included in the library. These are identified by tags following the standard basis set name.

For the Douglas-Kroll method the tag `dk` should be specified:

```
oxygen library cc-pVDZ_DK
```

For the Dyall-modified-Dirac (DMD) method three tags should be specified. The first is a tag for the nuclear model, which can be `pt` or `fi` for a point or a finite Gaussian model (see Section6). The second is a tag for the relativistic Hamiltonian: `sf` is for the spin-free modified Dirac Hamiltonian. The third tags the component type, `fw` for the atomic FW transformed large component, `lc` for the large component and `sc` for the small component:

```
oxygen library cc-pvdz_pt_sf_fw
oxygen library cc-pvdz_pt_sf_lc
```

Basis sets which are available with either the DmD or DK contractions are indicated in the list below. Additional DmD basis sets are available in the directory `nwchem/contrib/basissets/Dyall_DMD`.

Here is a list of known all-electron non-relativistic, DK and DmD basis sets, effective core potentials with their respective basis sets, fitting basis sets, and Polarization, Diffuse and Core-valence sets of functions, along with the elements included for each. Additional information about each basis set in the NWChem library can be obtained from the online EMSL Gaussian Basis Set library.

Standard all-electron basis sets:

1. Basis Set `"STO-2G"` (number of atoms 21)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca Sr

2. Basis Set `"STO-3G"` (number of atoms 53)
   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

3. Basis Set `"STO-6G"` (number of atoms 36)
   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

4. Basis Set `"STO-3G*"` (number of atoms 18)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar

5. Basis Set `"3-21G"` (number of atoms 55)
   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
6. Basis Set `"3-21++G"` (number of atoms 18)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar
7. Basis Set `"3-21G*"` (number of atoms 18)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar
8. Basis Set `"3-21++G*"` (number of atoms 18)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar
9. Basis Set `"3-21GSP"` (number of atoms 18)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar
10. Basis Set `"4-22GSP"` (number of atoms 18)
    H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar
11. Basis Set `"4-31G"` (number of atoms 13)
    H He Li Be B C N O F Ne P S Cl
12. Basis Set `"6-31G"` (number of atoms 30)
    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
13. Basis Set `"6-31G*"` (number of atoms 30)
    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
14. Basis Set `"6-31G**"` (number of atoms 30)
    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
15. Basis Set `"6-31++G"` (number of atoms 20)
    H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca
16. Basis Set `"6-31++G*"` (number of atoms 18)
    H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar
17. Basis Set `"6-31++G**"` (number of atoms 18)
    H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar
18. Basis Set `"6-31+G*"` (number of atoms 18)
    H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar
19. Basis Set `"6-31G(3df,3pd)"` (number of atoms 18)
    H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar
20. Basis Set `"6-31G-Blaudeau"` (number of atoms 2)
    K Ca
21. Basis Set `"6-31G*-Blaudeau"` (number of atoms 2)
    K Ca
22. Basis Set `"6-311G"` (number of atoms 25)
    H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr I
23. Basis Set `"6-311G*"` (number of atoms 25)
    H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr I

24. Basis Set `"6-311G**"` (number of atoms 25)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr I

25. Basis Set `"6-311++G**"` (number of atoms 10)
H He Li Be B C N O F Ne

26. Basis Set `"6-311++G(2d,2p)"` (number of atoms 20)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca

27. Basis Set `"6-311G(2df,2pd)"` (number of atoms 12)
H He Li Be B C N O F Ne K Ca

28. Basis Set `"6-311+G*"` (number of atoms 10)
H He Li Be B C N O F Ne

29. Basis Set `"6-311++G(3df,3pd)"` (number of atoms 18)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar

30. Basis Set `"MINI (Huzinaga)"` (number of atoms 20)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca

31. Basis Set `"MINI (Scaled)"` (number of atoms 20)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca

32. Basis Set `"MIDI (Huzinaga)"` (number of atoms 19)
H He Li Be B C N O F Ne Na Al Si P S Cl Ar K Cs

33. Basis Set `"MIDI!"` (number of atoms 11)
H C N O F Si P S Cl Br I

34. Basis Set `"SV (Dunning-Hay)"` (number of atoms 9)
H Li Be B C N O F Ne

35. Basis Set `"SVP (Dunning-Hay)"` (number of atoms 9)
H Li Be B C N O F Ne

36. Basis Set `"SVP + Diffuse (Dunning-Hay)"` (number of atoms 9)
H Li Be B C N O F Ne

37. Basis Set `"TZ (Dunning)"` (number of atoms 8)
H Li B C N O F Ne

38. Basis Set `"Chipman DZP + Diffuse"` (number of atoms 6)
H B C N O F

39. Basis Set `"DZ (Dunning)"` (number of atoms 13)
H Li B C N O F Ne Al Si P S Cl

40. Basis Set `"DZP (Dunning)"` (number of atoms 13)
H Li B C N O F Ne Al Si P S Cl

41. Basis Set `"DZP + Diffuse (Dunning)"` (number of atoms 7)
H B C N O F Ne

42. Basis Set `"cc-pVDZ"` (number of atoms 24)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr

43. Basis Set "`cc-pVTZ`" (number of atoms 24)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr

44. Basis Set "`cc-pVQZ`" (number of atoms 22)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se

45. Basis Set "`cc-pV5Z`" (number of atoms 24)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr

46. Basis Set "`cc-pV6Z`" (number of atoms 14)
H He B C N O F Ne Al Si P S Cl Ar

47. Basis Set "`pV6Z`" (number of atoms 4)
H C N O

48. Basis Set "`cc-pVDZ(seg-opt)`" (number of atoms 24)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr

49. Basis Set "`cc-pVTZ(seg-opt)`" (number of atoms 24)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr

50. Basis Set "`cc-pVQZ(seg-opt)`" (number of atoms 4)
H He C O

51. Basis Set "`cc-pCVDZ`" (number of atoms 12)
Li Be B C N O F Ne Na Mg Si Cl

52. Basis Set "`cc-pCVTZ`" (number of atoms 13)
Li Be B C N O F Ne Na Mg Si P Cl

53. Basis Set "`cc-pCVQZ`" (number of atoms 13)
Li Be B C N O F Ne Na Mg Si P Cl

54. Basis Set "`cc-pCV5Z`" (number of atoms 8)
B C N O F Ne Na Mg

55. Basis Set "`cc-pCV6Z`" (number of atoms 1)
O

56. Basis Set "`aug-cc-pVDZ`" (number of atoms 24)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr

57. Basis Set "`aug-cc-pVTZ`" (number of atoms 23)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br

58. Basis Set "`aug-cc-pVQZ`" (number of atoms 24)
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr

59. Basis Set "`aug-cc-pV5Z`" (number of atoms 20)
H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr

60. Basis Set "`aug-cc-pV6Z`" (number of atoms 14)
H He B C N O F Ne Al Si P S Cl Ar

61. Basis Set "`aug-cc-pCVDZ`" (number of atoms 7)
B C N O F Na Mg

62. Basis Set `"aug-cc-pCVTZ"` (number of atoms 8)
    B C N O F Ne Na Mg

63. Basis Set `"aug-cc-pCVQZ"` (number of atoms 6)
    B C N O F Ne

64. Basis Set `"aug-cc-pCV5Z"` (number of atoms 5)
    B C N O F

65. Basis Set `"d-aug-cc-pVDZ"` (number of atoms 8)
    H He B C N O F Ne

66. Basis Set `"d-aug-cc-pVTZ"` (number of atoms 8)
    H He B C N O F Ne

67. Basis Set `"d-aug-cc-pVQZ"` (number of atoms 8)
    H He B C N O F Ne

68. Basis Set `"d-aug-cc-pV5Z"` (number of atoms 8)
    H He B C N O F Ne

69. Basis Set `"d-aug-cc-pV6Z"` (number of atoms 5)
    H B C N O

70. Basis Set `"cc-pV(D+d)Z"` (number of atoms 6)
    Al Si P S Cl Ar

71. Basis Set `"cc-pV(T+d)Z"` (number of atoms 6)
    Al Si P S Cl Ar

72. Basis Set `"cc-pV(Q+d)Z"` (number of atoms 6)
    Al Si P S Cl Ar

73. Basis Set `"cc-pV(5+d)Z"` (number of atoms 6)
    Al Si P S Cl Ar

74. Basis Set `"cc-pV(6+d)Z"` (number of atoms 6)
    Al Si P S Cl Ar

75. Basis Set `"aug-cc-pV(D+d)Z"` (number of atoms 6)
    Al Si P S Cl Ar

76. Basis Set `"aug-cc-pV(T+d)Z"` (number of atoms 6)
    Al Si P S Cl Ar

77. Basis Set `"aug-cc-pV(Q+d)Z"` (number of atoms 6)
    Al Si P S Cl Ar

78. Basis Set `"aug-cc-pV(5+d)Z"` (number of atoms 6)
    Al Si P S Cl Ar

79. Basis Set `"aug-cc-pV(6+d)Z"` (number of atoms 6)
    Al Si P S Cl Ar

80. Basis Set `"Feller Misc. CVDZ"` (number of atoms 1)
    K

81. Basis Set `"Feller Misc. CVTZ"` (number of atoms 1)
    K

82. Basis Set `"Feller Misc. CVQZ"` (number of atoms 1)
    K

83. Basis Set `"GAMESS VTZ"` (number of atoms 16)
    H Be B C N O F Ne Na Mg Al Si P S Cl Ar

84. Basis Set `"GAMESS PVTZ"` (number of atoms 8)
    H Be B C N O F Ne

85. Basis Set `"Partridge Uncontr. 1"` (number of atoms 36)
    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

86. Basis Set `"Partridge Uncontr. 2"` (number of atoms 34)
    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

87. Basis Set `"Partridge Uncontr. 3"` (number of atoms 28)
    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

88. Basis Set `"Ahlrichs VDZ"` (number of atoms 36)
    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

89. Basis Set `"Ahlrichs pVDZ"` (number of atoms 36)
    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

90. Basis Set `"Ahlrichs VTZ"` (number of atoms 36)
    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

91. Basis Set `"Ahlrichs TZV"` (number of atoms 34)
    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

92. Basis Set `"Binning/Curtiss SV"` (number of atoms 6)
    Ga Ge As Se Br Kr

93. Basis Set `"Binning/Curtiss VTZ"` (number of atoms 6)
    Ga Ge As Se Br Kr

94. Basis Set `"Binning/Curtiss SVP"` (number of atoms 6)
    Ga Ge As Se Br Kr

95. Basis Set `"Binning/Curtiss VTZP"` (number of atoms 6)
    Ga Ge As Se Br Kr

96. Basis Set `"McLean/Chandler VTZ"` (number of atoms 8)
    Na Mg Al Si P S Cl Ar

97. Basis Set `"SV + Rydberg (Dunning-Hay)"` (number of atoms 9)
    H Li Be B C N O F Ne

98. Basis Set `"SVP + Rydberg (Dunning-Hay)"` (number of atoms 9)
    H Li Be B C N O F Ne

99. Basis Set `"SVP + Diffuse + Rydberg"` (number of atoms 9)
    H Li Be B C N O F Ne
100. Basis Set `"DZ + Rydberg (Dunning)"` (number of atoms 12)
    H B C N O F Ne Al Si P S Cl
101. Basis Set `"DZP + Rydberg (Dunning)"` (number of atoms 12)
    H B C N O F Ne Al Si P S Cl
102. Basis Set `"DZ + Double Rydberg (Dunning-Hay)"` (number of atoms 12)
    H B C N O F Ne Al Si P S Cl
103. Basis Set `"SV + Double Rydberg (Dunning-Hay)"` (number of atoms 9)
    H Li Be B C N O F Ne
104. Basis Set `"Wachters+f"` (number of atoms 9)
    Sc Ti V Cr Mn Fe Co Ni Cu
105. Basis Set `"Bauschlicher ANO"` (number of atoms 9)
    Sc Ti V Cr Mn Fe Co Ni Cu
106. Basis Set `"NASA Ames ANO"` (number of atoms 12)
    H B C N O F Ne Al P Ti Fe Ni
107. Basis Set `"NASA Ames cc-pVTZ"` (number of atoms 1)
    Ti
108. Basis Set `"NASA Ames cc-pVQZ"` (number of atoms 1)
    Ti
109. Basis Set `"NASA Ames cc-pV5Z"` (number of atoms 1)
    Ti
110. Basis Set `"NASA Ames cc-pCVTZ"` (number of atoms 1)
    Ti
111. Basis Set `"NASA Ames cc-pCVQZ"` (number of atoms 1)
    Ti
112. Basis Set `"NASA Ames cc-pCV5Z"` (number of atoms 1)
    Ti
113. Basis Set `"Sadlej pVTZ"` (number of atoms 19)
    H Li Be C N O F Na Mg Si P S Cl K Ca Br Rb Sr I
114. Basis Set `"WTBS"` (number of atoms 84)
    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 Ce Pr Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn

Resolution of Identity (RI) fitting basis sets:

1. Basis Set `"cc-pVDZ-fit2-1"` (number of atoms 10)
   H He Li Be B C N O F Ne
2. Basis Set `"cc-pVTZ-fit2-1"` (number of atoms 10)
   H He Li Be B C N O F Ne

Density functional specific basis sets:

1. Basis Set `"DZVP (DFT Orbital)"` (number of atoms 54)
   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
2. Basis Set `"DZVP2 (DFT Orbital)"` (number of atoms 25)
   H He Li Be B C N O F Al Si P S Cl Ar Sc Ti V Cr Mn Fe Co Ni Cu Zn
3. Basis Set `"TZVP (DFT Orbital)"` (number of atoms 11)
   H C N O F Al Si P S Cl Ar

Density functional Coulomb and Exchange fitting basis sets:

1. Basis Set `"DGauss A1 DFT Coulomb Fitting"` (number of atoms 54)
   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
2. Basis Set `"DGauss A1 DFT Exchange Fitting"` (number of atoms 54)
   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
3. Basis Set `"DGauss A2 DFT Coulomb Fitting"` (number of atoms 28)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Sc Ti V Cr Mn Fe Co Ni Cu Zn
4. Basis Set `"DGauss A2 DFT Exchange Fitting"` (number of atoms 64)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Sc Ti V Cr Mn Fe Co Ni Cu Zn
5. Basis Set `"Ahlrichs Coulomb Fitting"` (number of atoms 70)
   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 Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At
6. Basis Set `"DeMon Coulomb Fitting"` (number of atoms 54)
   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

Effective core potentials and their respective basis sets:

1. Basis Set `"Hay-Wadt MB (n+1) ECP"` (number of atoms 32)
   K Ca Sc Ti V Cr Mn Fe Co Ni Cu Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cs Ba La Ta W Re Os Ir Pt Au
2. ECP `"Hay-Wadt MB (n+1) ECP"` (number of atoms 32)
   K Ca Sc Ti V Cr Mn Fe Co Ni Cu Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cs Ba La Ta W Re Os Ir Pt Au
3. Basis Set `"Hay-Wadt VDZ (n+1) ECP"` (number of atoms 32)
   K Ca Sc Ti V Cr Mn Fe Co Ni Cu Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cs Ba La Ta W Re Os Ir Pt Au
4. ECP `"Hay-Wadt VDZ (n+1) ECP"` (number of atoms 32)
   K Ca Sc Ti V Cr Mn Fe Co Ni Cu Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cs Ba La Ta W Re Os Ir Pt Au

5. Basis Set `"LANL2DZ ECP"` (number of atoms 67)
H 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 U Np Pu

6. ECP `"LANL2DZ ECP"` (number of atoms 58)
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 U Np Pu

7. Basis Set `"LANL2DZdp ECP"` (number of atoms 19)
H C N O F Si P S Cl Ge As Se Br Sn Sb Te I Pb Bi

8. ECP `"LANL2DZdp ECP"` (number of atoms 13)
Si P S Cl Ge As Se Sn Sb Te I Pb Bi

9. Basis Set `"SBKJC VDZ ECP"` (number of atoms 73)
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 Ce Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn

10. ECP `"SBKJC VDZ ECP"` (number of atoms 71)
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 Ce Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn

11. Basis Set `"CRENBL ECP"` (number of atoms 116)
H 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 Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu Hf Ta W Re Os Ir Pt Au Hg Pb Bi Po At Rn Fr Ra Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr Rf Db Sg Bh Hs Mt Un Uu Ub Ut Uq Up Uh Us Uo

12. ECP `"CRENBL ECP"` (number of atoms 115)
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 Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu Hf Ta W Re Os Ir Pt Au Hg Pb Bi Po At Rn Fr Ra Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr Rf Db Sg Bh Hs Mt Un Uu Ub Ut Uq Up Uh Us Uo

13. Basis Set `"CRENBS ECP"` (number of atoms 50)
Sc Ti V Cr Mn Fe Co Ni Cu Zn Y Zr Nb Mo Tc Ru Rh Pd Ag Cd La Hf Ta W Re Os Ir Pt Au Hg Pb Bi Po At Rn Rf Db Sg Bh Hs Mt Un Uu Ub Ut Uq Up Uh Us Uo

14. ECP `"CRENBS ECP"` (number of atoms 50)
Sc Ti V Cr Mn Fe Co Ni Cu Zn Y Zr Nb Mo Tc Ru Rh Pd Ag Cd La Hf Ta W Re Os Ir Pt Au Hg Pb Bi Po At Rn Rf Db Sg Bh Hs Mt Un Uu Ub Ut Uq Up Uh Us Uo

15. Basis Set `"Stuttgart RLC ECP"` (number of atoms 57)
Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca Zn Ga Ge As Se Br Kr Rb Sr In Sn Sb Te I Xe Cs Ba Hg Tl Pb Bi Po At Rn Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr

16. ECP `"Stuttgart RLC ECP"` (number of atoms 57)
Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca Zn Ga Ge As Se Br Kr Rb Sr In Sn Sb Te I Xe Cs Ba Hg Tl Pb Bi Po At Rn Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr

17. Basis Set `"Stuttgart RSC 1997 ECP"` (number of atoms 64)
K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd Cs Ba Ce Pr Nd Pm

Sm Eu Gd Tb Dy Ho Er Tm Yb Hf Ta W Re Os Ir Pt Au Hg Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr Db

18. ECP `"Stuttgart RSC 1997 ECP"` (number of atoms 64)
    K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd Cs Ba Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Hf Ta W Re Os Ir Pt Au Hg Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr Db
19. Basis Set `"Stuttgart RSC Segmented/ECP"` (number of atoms 15)
    La Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu
20. ECP `"Stuttgart RSC Segmented/ECP"` (number of atoms 15)
    La Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu
21. Basis Set `"Stuttgart RSC ANO/ECP"` (number of atoms 15)
    La Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu
22. ECP `"Stuttgart RSC ANO/ECP"` (number of atoms 15)
    La Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu
23. Basis Set `"SDB-cc-pVTZ"` (number of atoms 12)
    Ga Ge As Se Br Kr In Sn Sb Te I Xe
24. ECP `"SDB-cc-pVTZ"` (number of atoms 12)
    Ga Ge As Se Br Kr In Sn Sb Te I Xe
25. Basis Set `"SDB-cc-pVQZ"` (number of atoms 12)
    Ga Ge As Se Br Kr In Sn Sb Te I Xe
26. ECP `"SDB-cc-pVQZ"` (number of atoms 12)
    Ga Ge As Se Br Kr In Sn Sb Te I Xe
27. Basis Set `"SDB-aug-cc-pVTZ"` (number of atoms 10)
    Ga Ge As Se Br In Sn Sb Te I
28. ECP `"SDB-aug-cc-pVTZ"` (number of atoms 30)
    Ga Ge As Se Br In Sn Sb Te I Ga Ge As Se Br In Sn Sb Te I
29. Basis Set `"SDB-aug-cc-pVQZ"` (number of atoms 10)
    Ga Ge As Se Br In Sn Sb Te I
30. ECP `"SDB-aug-cc-pVQZ"` (number of atoms 10)
    Ga Ge As Se Br In Sn Sb Te I

Douglas-Kroll (DK) all-electron basis sets:

1. Basis Set `"cc-pVDZ DK"` (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
2. Basis Set `"cc-pVTZ DK"` (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
3. Basis Set `"cc-pVQZ DK"` (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
4. Basis Set `"cc-pV5Z DK"` (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr

Dyall's Modified Dirac (DmD) all-electron basis sets:

1. Basis Set "cc-pvdz fi sf fw" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
2. Basis Set "cc-pvdz fi sf lc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
3. Basis Set "cc-pvdz fi sf sc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
4. Basis Set "cc-pvdz pt sf fw" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
5. Basis Set "cc-pvdz pt sf lc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
6. Basis Set "cc-pvdz pt sf sc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
7. Basis Set "cc-pvtz fi sf fw" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
8. Basis Set "cc-pvtz fi sf lc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
9. Basis Set "cc-pvtz fi sf sc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
10. Basis Set "cc-pvtz pt sf fw" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
11. Basis Set "cc-pvtz pt sf lc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
12. Basis Set "cc-pvtz pt sf sc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
13. Basis Set "cc-pvqz fi sf fw" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
14. Basis Set "cc-pvqz fi sf lc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
15. Basis Set "cc-pvqz fi sf sc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
16. Basis Set "cc-pvqz pt sf fw" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
17. Basis Set "cc-pvqz pt sf lc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
18. Basis Set "cc-pvqz pt sf sc" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr
19. Basis Set "cc-pv5z fi sf fw" (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr

20. Basis Set "cc-pv5z fi sf lc" (number of atoms 20)
    H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr

21. Basis Set "cc-pv5z fi sf sc" (number of atoms 20)
    H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr

22. Basis Set "cc-pv5z pt sf fw" (number of atoms 20)
    H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr

23. Basis Set "cc-pv5z pt sf lc" (number of atoms 20)
    H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr

24. Basis Set "cc-pv5z pt sf sc" (number of atoms 20)
    H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr

Polarization functions:

1. Basis Set "STO-3G* Polarization" (number of atoms 8)
   Na Mg Al Si P S Cl Ar

2. Basis Set "3-21G* Polarization" (number of atoms 8)
   Na Mg Al Si P S Cl Ar

3. Basis Set "6-31G* Polarization" (number of atoms 27)
   Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca Sc Ti V Cr Mn Co Ni Cu Zn

4. Basis Set "6-31G** Polarization" (number of atoms 29)
   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 Co Ni Cu Zn

5. Basis Set "6-311G* Polarization" (number of atoms 23)
   Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr I

6. Basis Set "6-311G** Polarization" (number of atoms 25)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr I

7. Basis Set "Pople (2d,2p) Polarization" (number of atoms 26)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca Ga Ge As Se Br Kr

8. Basis Set "Pople (2df,2pd) Polarization" (number of atoms 12)
   H He Li Be B C N O F Ne K Ca

9. Basis Set "Pople (3df,3pd) Polarization" (number of atoms 18)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar

10. Basis Set "HONDO7 Polarization" (number of atoms 17)
    H He Be B C N O F Ne Na Mg Al Si P S Cl Ar

11. Basis Set "Huzinaga Polarization" (number of atoms 22)
    K Ca Ga Ge As Se Br Kr Rb Sr In Sn Sb Te I Xe Tl Pb Bi Po At Rn

12. Basis Set "DHMS Polarization" (number of atoms 15)
    H He Li Be B C N O F Ne Al Si P S Cl

13. Basis Set "Binning-Curtiss (1d) Polarization" (number of atoms 6)
    Ga Ge As Se Br Kr

14. Basis Set `"Binning-Curtiss (df) Polarization"` (number of atoms 6)
    Ga Ge As Se Br Kr

15. Basis Set `"Ahlrichs Polarization"` (number of atoms 36)
    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

16. Basis Set `"Glendening Polarization"` (number of atoms 3)
    K Rb Cs

17. Basis Set `"Blaudeau Polarization"` (number of atoms 2)
    K Ca

18. Basis Set `"LANL2DZdp ECP Polarization"` (number of atoms 19)
    H C N O F Si P S Cl Ge As Se Br Sn Sb Te I Pb Bi

19. Basis Set `"Dunning-Hay Rydberg"` (number of atoms 12)
    B C N O F Ne Al Si P S Cl Ar

20. Basis Set `"Dunning-Hay Double Rydberg"` (number of atoms 12)
    B C N O F Ne Al Si P S Cl Ar

Diffuse functions:

1. Basis Set `"Pople-style Diffuse"` (number of atoms 19)
   H Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca

2. Basis Set `"Dunning-Hay Diffuse"` (number of atoms 9)
   H Li Be B C N O F Ne

3. Basis Set `"aug-cc-pVDZ Diffuse"` (number of atoms 24)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr

4. Basis Set `"aug-cc-pVTZ Diffuse"` (number of atoms 23)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br

5. Basis Set `"aug-cc-pVQZ Diffuse"` (number of atoms 24)
   H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar Ga Ge As Se Br Kr

6. Basis Set `"aug-cc-pV5Z Diffuse"` (number of atoms 20)
   H He B C N O F Ne Al Si P S Cl Ar Ga Ge As Se Br Kr

7. Basis Set `"aug-cc-pV6Z Diffuse"` (number of atoms 14)
   H He B C N O F Ne Al Si P S Cl Ar

8. Basis Set `"d-aug-cc-pVDZ Diffuse"` (number of atoms 8)
   H He B C N O F Ne

9. Basis Set `"d-aug-cc-pVTZ Diffuse"` (number of atoms 8)
   H He B C N O F Ne

10. Basis Set `"d-aug-cc-pVQZ Diffuse"` (number of atoms 8)
    H He B C N O F Ne

11. Basis Set `"d-aug-cc-pV5Z Diffuse"` (number of atoms 8)
    H He B C N O F Ne

12. Basis Set `"d-aug-cc-pV6Z Diffuse"` (number of atoms 5)
    H B C N O
13. Basis Set `"aug-cc-pV(D+d)Z Diffuse"` (number of atoms 6)
    Al Si P S Cl Ar
14. Basis Set `"aug-cc-pV(T+d)Z Diffuse"` (number of atoms 6)
    Al Si P S Cl Ar
15. Basis Set `"aug-cc-pV(Q+d)Z Diffuse"` (number of atoms 6)
    Al Si P S Cl Ar
16. Basis Set `"aug-cc-pV(5+d)Z Diffuse"` (number of atoms 6)
    Al Si P S Cl Ar
17. Basis Set `"aug-cc-pV(6+d)Z Diffuse"` (number of atoms 6)
    Al Si P S Cl Ar
18. Basis Set `"SDB-aug-cc-pVTZ Diffuse"` (number of atoms 10)
    Ga Ge As Se Br In Sn Sb Te I
19. Basis Set `"SDB-aug-cc-pVQZ Diffuse"` (number of atoms 10)
    Ga Ge As Se Br In Sn Sb Te I

Core-valence functions:

1. Basis Set `"Core/val. Functions (cc-pCVDZ)"` (number of atoms 12)
   Li Be B C N O F Ne Na Mg Si Cl
2. Basis Set `"Core/val. Functions (cc-pCVTZ)"` (number of atoms 13)
   Li Be B C N O F Ne Na Mg Si P Cl
3. Basis Set `"Core/val. Functions (cc-pCVQZ)"` (number of atoms 13)
   Li Be B C N O F Ne Na Mg Si P Cl
4. Basis Set `"Core/val. Functions (cc-pCV5Z)"` (number of atoms 8)
   B C N O F Ne Na Mg
5. Basis Set `"Core/val. Functions (cc-pCV6Z)"` (number of atoms 1)
   O

Appendix B

Sample input files

B.1 Water SCF calculation and geometry optimization in a 6-31g basis

The input file in section 2 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).

B.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.

B.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.

B.2 Compute the polarizability of Ne using finite field

B.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.

B.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 (section 6).

B.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    1 1.00
   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
```

B.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 1*s* 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)
```

Appendix C

Examples of geometries using symmetry

Below are examples of the use of the SYMMETRY directive in the compound GEOMETRY directive (Section 6). 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.

C.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
```

C.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
```

C.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
```

C.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
```

C.5 T_d methane

For ease of use, the primary C_3 axis should be the x=y=z axis. The 3 C_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
```

C.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
```

C.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
```

C.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
```

C.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
```

C.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
```

C.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
```

C.12 C_{3h} BO_3H_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
```

C.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
```

C.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
```

C.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
```

C.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
```

C.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
```

C.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
```

Appendix D

Running NWChem

The command required to invoke NWChem is machine dependent, whereas most of the NWChem input is machine independent[1] .

D.1 Sequential execution

To run NWChem sequentially on nearly all UNIX-based platforms simply use the command `nwchem` and provide the name of the input file as an argument (See section 2.1 for more information). This does assume that either `nwchem` is in your path or you have set an alias of `nwchem` to point to the appropriate executable.

Output is to standard output, standard error and Fortran unit 6 (usually the same as standard output). Files are created by default in the current directory, though this may be overridden in the input (section 5.2).

Generally, one will run a job with the following command:

```
nwchem input.nw >& input.out &
```

D.2 Parallel execution on UNIX-based parallel machines including workstation clusters using TCGMSG

These platforms require the use of the TCGMSG[2] `parallel` command and thus also require the definition of a process-group (or procgroup) file. The process-group file describes how many processes to start, what program to run, which machines to use, which directories to work in, and under which userid to run the processes. By convention the process-group file has a `.p` suffix.

The process-group file is read to end-of-file. The character `#` (hash or pound sign) is used to indicate a comment which continues to the next new-line character. Each line describes a cluster of processes and consists of the following whitespace separated fields:

```
userid hostname nslave executable workdir
```

[1] Machine dependence within the input arises from file names, machine specific resources, and differing services provided by the operating system.

[2] Where required TCGMSG is automatically built with NWChem.

- `userid` – The user-name on the machine that will be executing the process.
- `hostname` – The hostname of the machine to execute this process. If it is the same machine on which parallel was invoked the name must match the value returned by the command hostname. If a remote machine it must allow remote execution from this machine (see man pages for rlogin, rsh).
- `nslave` – The total number of copies of this process to be executing on the specified machine. Only "clusters" of identical processes specified in this fashion can use shared memory to communicate. If no shared memory is supported on machine `<hostname>` then only the value one (1) is valid.
- `executable` – Full path name on the host `<hostname>` of the image to execute. If `<hostname>` is the local machine then a local path will suffice.
- `workdir` – Full path name on the host `<hostname>` of the directory to work in. Processes execute a chdir() to this directory before returning from pbegin(). If specified as a "." then remote processes will use the login directory on that machine and local processes (relative to where parallel was invoked) will use the current directory of parallel.

For example, if your file `"nwchem.p"` contained the following

```
d3g681 pc 4 /msrc/apps/bin/nwchem /scr22/rjh
```

then 4 processes running NWChem would be started on the machine `pc` running as user `d3g681` in directory `"/scr22/rjh"`. To actually run this simply type:

```
parallel nwchem big_molecule.nw
```

N.B. : The first process specified (process zero) is the only process that

- opens and reads the input file, and
- opens and reads/updates the database.

Thus, if your file systems are physically distributed (e.g., most workstation clusters) you must ensure that process zero can correctly resolve the paths for the input and database files.

N.B. In releases of NWChem prior to 3.3 additional processes had to be created on workstation clusters to support remote access to shared memory. This is no longer the case. The TCGMSG process group file now just needs to refer to processes running NWChem.

D.3 Parallel execution on UNIX-based parallel machines including workstation clusters using MPI

To run with MPI, `parallel` should not be used. The way we usually run nwchem under MPI are the following

- using mpirun:

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

- If you have all nodes connected via shared memory and you have installed the ch_shmem version of MPICH, you can do

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

D.4 Parallel execution on MPPs

All of these machines require use of different commands in order to gain exclusive access to computational resources.

D.5 IBM SP

If using POE (IBM's Parallel Operating Environment) interactively, simply create the list of nodes to use in the file `"host.list"` in the current directory and invoke NWChem with

```
nwchem <input_file> -procs <n>
```

where `n` is the number of processes to use. Process 0 will run on the first node in `"host.list"` and must have access to the input and other necessary files. Very significant performance gains may be had by setting the following environment variables before running NWChem (or setting them using POE command line options).

- `setenv MP_EUILIB us` — dedicated user space communication over the switch (the default is IP over the switch which is much slower).
- `setenv MP_CSS_INTERRUPT yes` — enable interrupts when a message arrives (the default is to poll which significantly slows down global array accesses).

In addition, if the IBM is running PSSP version 3.1, or later

- `setenv MP_MSG_API lapi`, or
- `setenv MP_MSG_API mpi,lapi` (if using both GA and MPI)

For batch execution, we recommend use of the `llnw` command which is installed in `/usr/local/bin` on the EMSL/PNNL IBM SP. If you are not running on that system, the `llnw` script may be found in the NWChem distribution directory contrib/loadleveler. Interactive help may be obtained with the command `llnw -help`. Otherwise, the very simplest job to run NWChem in batch using Load Leveller is something like this

```
#!/bin/csh -x
# @ job_type         =    parallel
# @ class            =    small
# @ network.lapi     = css0,not_shared,US
# @ input            =    /dev/null
# @ output           =    <OUTPUT_FILE_NAME>
# @ error            =    <ERROUT_FILE_NAME>
# @ environment      =    COPY_ALL; MP_PULSE=0; MP_SINGLE_THREAD=yes; MP_WAIT_MODE=yield; restart=no
# @ min_processors   =    7
# @ max_processors   =    7
# @ cpu_limit        =    1:00:00
# @ wall_clock_limit =    1:00:00
# @ queue
#
```

```
cd /scratch

nwchem <INPUT_FILE_NAME>
```

Substitute `<OUTPUT_FILE_NAME>`, `<ERROUT_FILE_NAME>` and `<INPUT_FILE_NAME>` with the *full* path of the appropriate files. Also, if you are using an SP with more than one processor per node, you will need to substitute

```
# @ network.lapi     = css0,shared,US
# @ node             = NNODE
# @ tasks_per_node   = NTASK
```

for the lines

```
# @ network.lapi     = css0,not_shared,US
# @ min_processors   =    7
# @ max_processors   =    7
```

where `NNODE` is the number of physical nodes to be used and `NTASK` is the number of tasks per node.

These files and the NWChem executable must be in a file system accessible to all processes. Put the above into a file (e.g., `"test.job"`) and submit it with the command

```
llsubmit test.job
```

It will run a 7 processor, 1 hour job in the queue `small`. It should be apparent how to change these values.

Note that on many IBM SPs, including that at EMSL, the local scratch disks are wiped clean at the beginning of each job and therefore persistent files should be stored elsewhere. PIOFS is recommended for files larger than a few MB.

D.6 Cray T3E

```
mpprun -n <npes> $NWCHEM_TOP/bin/$NWCHEM_TARGET/nwchem <input_file>
```

where `npes` is the number of processors and `input_file` is the name of your input file.

D.7 Linux

If running in parallel across multiple machines you should consider applying this patch to your kernel to boost the performance of TCP/IP

- http://www.icase.edu/coral/LinuxTCP.html

D.8 Alpha systems with Quadrics switch

```
prun -n <npes> $NWCHEM_TOP/bin/$NWCHEM_TARGET/nwchem <input_file>
```

where `npes` is the number of processors and `input_file` is the name of your input file.

D.9 Windows 98 and NT

```
$NWCHEM_TOP/bin/win32/nw32 <input_file>
```

where and `input_file` is the name of your input file. If you use WMPI, uou must have a file named `nw32.pg` in the `$NWCHEM_TOP/bin/win32` directory; the file must only contains the following single line

```
local 0
```

D.10 Tested Platforms and O/S versions

- IBM SP with P2SC nodes, AIX 4.2.1, and PSSP 2.3
- IBM SP with silver nodes (SMP nodes with two 604e processors), AIX 4.3.2, and PSSP 3.1.
- IBM RS6000 workstation, AIX 3.2 and 4.1.
- Cray T3E, 2.0.4.61 UNICOSMK
- SGI R8000/10000, IRIX 6.2, 6.5
- SGI R4000, IRIX 5.3
- SUN workstations, SunOS 4.1.3 and Solaris 5.5, 5.6
- Compaq DEC alpha workstion
  - (600 MHz EV6), Digital UNIX V4.0E Rev. 1091, DEC C V5.8-009, Digital Fortran V5.2
  - (667 MHz EV6), Digital UNIX V4.0F Rev. 1229, DEC C V5.9-005, Digital Fortran V5.2
- Linux. Since there are at least 8 popular distributions of the Linux operating system and numerous others in existence, including downloading everything and building your own Linux OS, it is impossible to test all possible versions of Linux with NWChem. NWChem Release 3.3 has been tested on Slackware 3.4, 3.5, 4.0, 7.0 RedHat 5.1, 5.2, and 6.0, 6.1, 6.2 Mandrake based on RedHat 6.0, RedHat 6.0 for the Power PC Macintosh, and Black Lab 1.2 or Yellow Dog 1.2 Linux for Power PC Macintosh. These all use the EGCS compilers at different levels. Those distributed from Slackware are somewhat different than those distributed from RedHat but the code is configured to run on all of them.
- Linux on Alpha cpus has been tested with RedHat 6.1 and 6.2. Usage of the Compaq Fortran and C compilers is necessary for compiling.
- HP 9000/800 workstations with HPUX B.11.00. f90 must be used for compilation.
- FreeBSD. Just use the Linux target and NWChem will build for FreeBSD version 4.0 and the pre-release 5.0 versions of the Operating System.
- Intel x86 with Windows 98 and NT has been tested with Compaq Visual Fortran 6.0 and 6.1 with WMPI 1.3 or NT-MPICH. NT-MPICH is available from http://www-unix.mcs.anl.gov/~ashton/mpich.nt/