NWChem Tutorial

Jeff Hammond

Leadership Computing Facility
Argonne National Laboratory

5 November 2010

NWChem 6.0 Overview

- **Open Source License:** ECL 2.0 — Apache 2.0 with patent modifications for academic users.
- **Wiki:** http://www.nwchem-sw.org
- **Capability:** The most diverse collection of quantum chemical methodology of any code. Notable missing features: TDDFT and CC analytic gradients.
- **Portability:** Runs on laptops/workstations (Linux, Mac and Cygwin), clusters (e.g. Fusion) and supercomputers (e.g. Intrepid and Jaguar).

Prerequisites

Required:

- **Software:**
  GNU/Linux environment or equivalent.
- **Hardware:**
  A real computer (not a cell phone).

If you want performance:

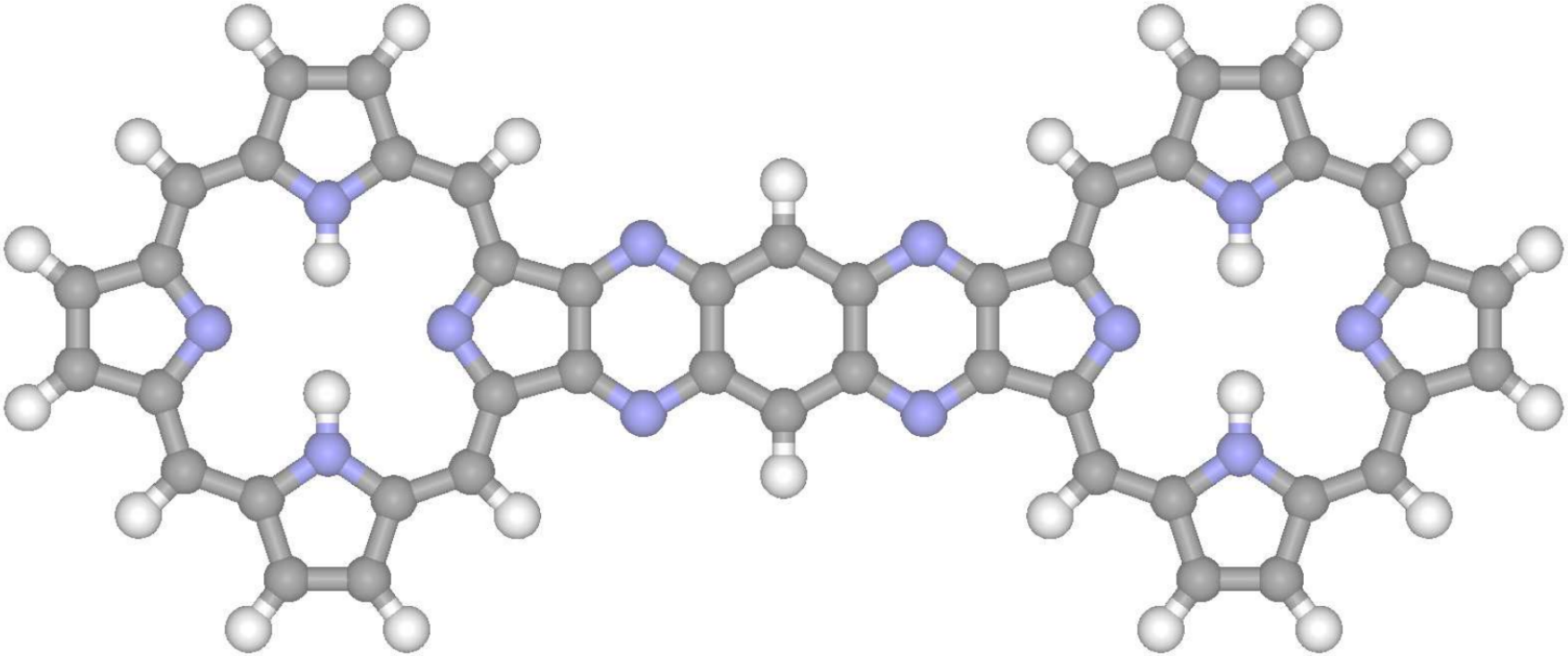
- **Software:**
  MPI, BLAS, LAPACK, ScaLAPACK.
- **Hardware:**
  Infiniband cluster or supercomputer.

Why Use NWChem?

- **Open Source:** Modify the code as needed.
- **Free:** Use anywhere.
- **Scalable:** No need to switch codes from laptop to supercomputer.
- **Functionality:** No need to use N codes for 1 project.
- **Support:** Responsive developers and user community via nwchem-users@emsl.pnl.gov.
- **Potential:** Actively developed by multiple DOE labs and user community.

NWChem Highlight Reel

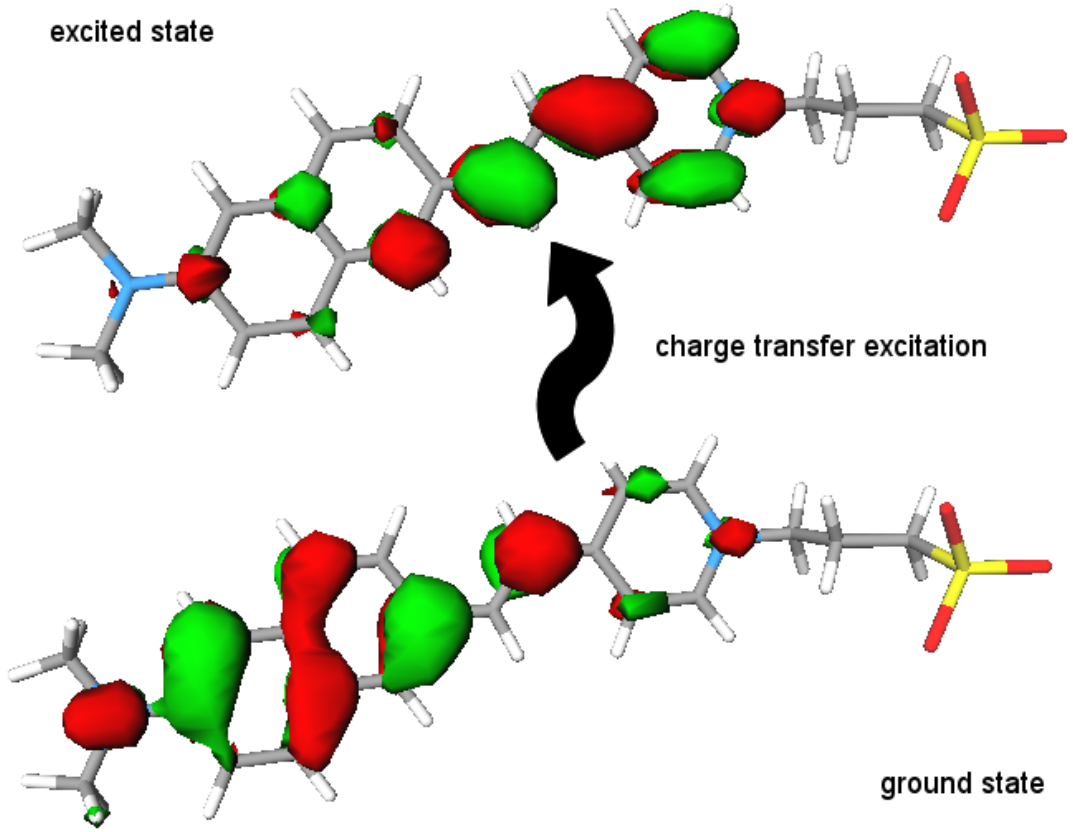
Largest Coupled-cluster Excited-state Calculation



J. Chem. Phys. **132**, 154103 (2010).

Charge-transfer Excited-states of Biomolecules

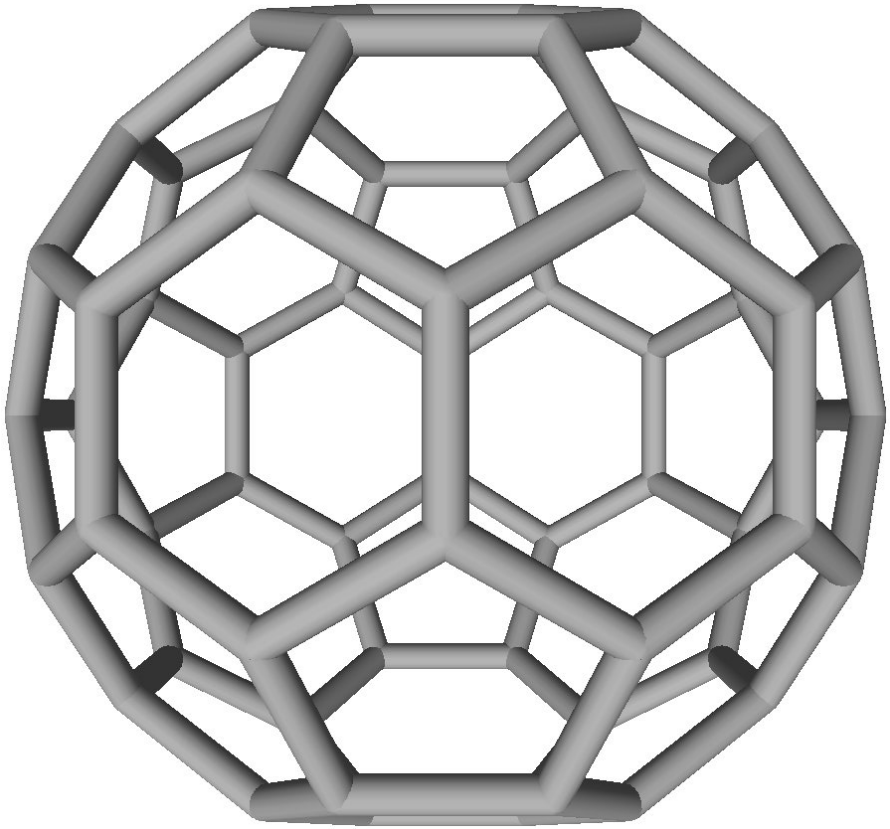
CR-EOMCCSD(T)/6-31G* — 1 hour on 256 cores



Joint work with Karol Kowalski (PNNL) and Benoît Roux (UC/Argonne).

CCSD-LR Dynamic Polarizability

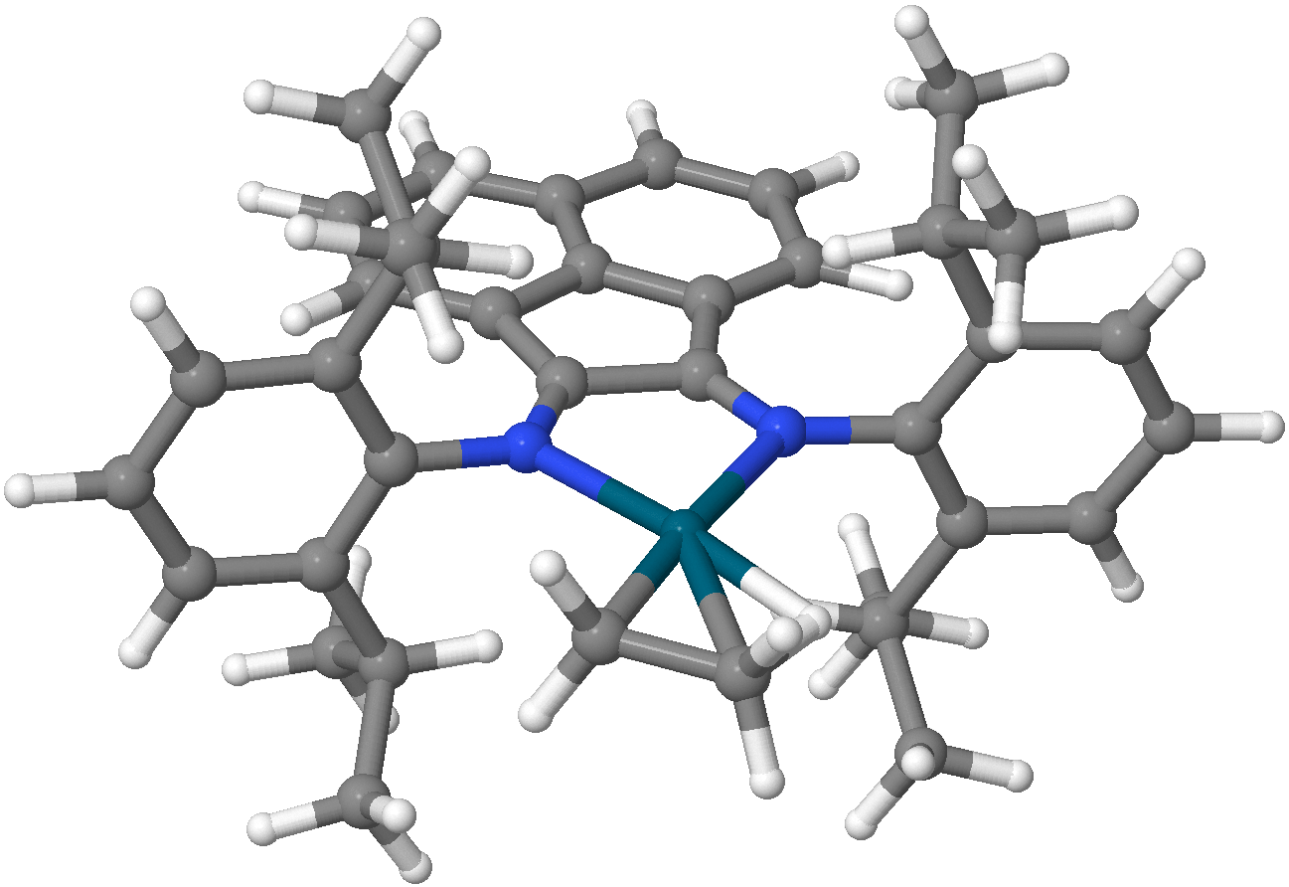
1080 b.f. — 40 hours on 1024 processors



J. Chem. Phys. **129**, 226101 (2008).

Large Fifth-rung DFT (B2PLYP)

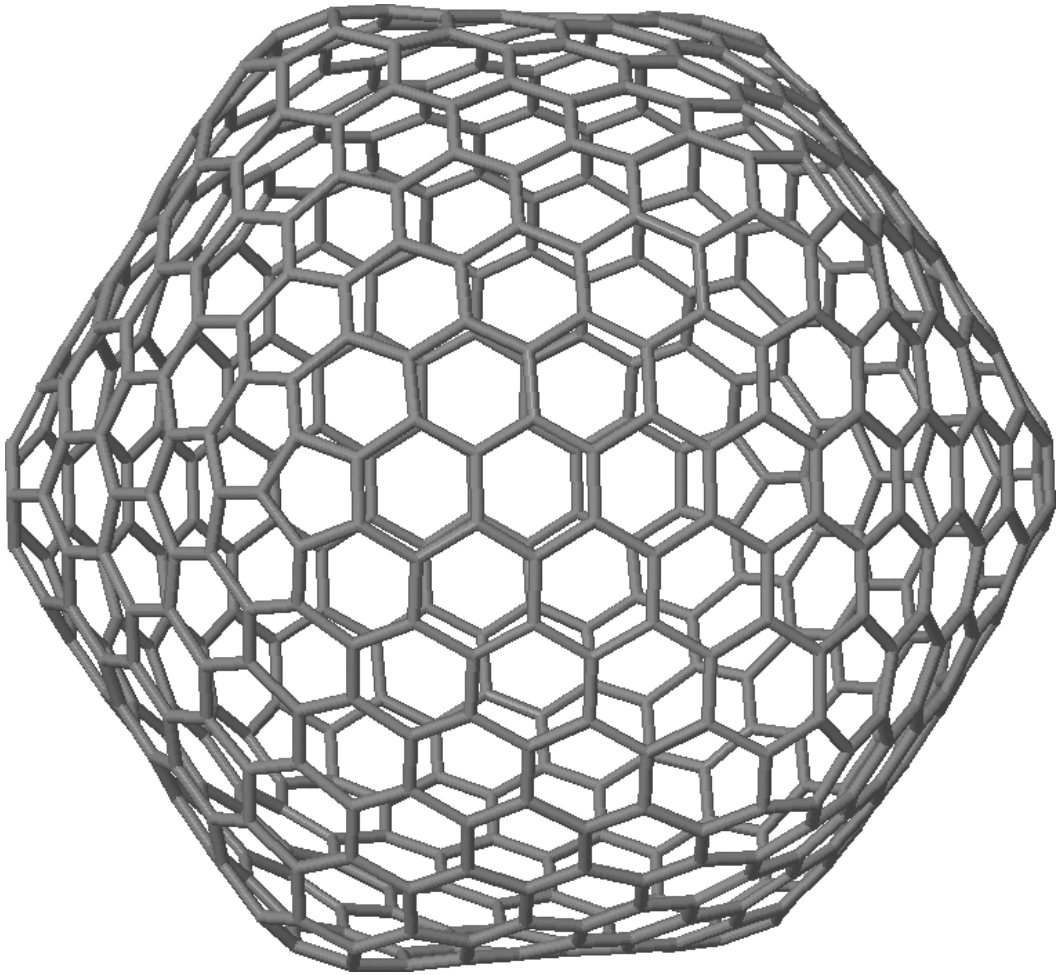
2154 b.f. — 7 hours on 256 cores



Organometallics **29**, 1750-1760 2010.

Large Hybrid DFT (PBE0)

no symmetry, 7560 b.f. — 6 hours on 1024 cores



Creating NWChem Input Files

Input File Structure

```
echo

start water

geometry
  O  0.00  0.00  0.12
  H  0.75  0.00 -0.47
  H -0.75  0.00 -0.47
end

basis
  * library 6-31G
end

task scf optimize
```

echo

Copies input into output.

start <prefix>

Names job, creates/overwrites existing files.

geometry and basis

Basic input blocks.

task <method> <task>

Initiates a calculation.

Input File Structure

```
echo

start water

geometry
  O  0.00  0.00  0.12
  H  0.75  0.00 -0.47
  H -0.75  0.00 -0.47
end

basis
  * library 6-31G
end

task scf optimize
```

echo

Copies input into output.

start <prefix>

Names job, creates/overwrites existing files.

geometry and basis

Basic input blocks.

task <method> <task>

Initiates a calculation.

Input File Structure

```
echo

start water

geometry
  O   0.00   0.00   0.12
  H   0.75   0.00  -0.47
  H  -0.75   0.00  -0.47
end

basis
  * library 6-31G
end

task scf optimize
```

echo

Copies input into output.

start <prefix>

Names job, creates/overwrites existing files.

geometry and basis

Basic input blocks.

task <method> <task>

Initiates a calculation.

Input File Structure

```
echo

start water

geometry
  O   0.00   0.00   0.12
  H   0.75   0.00  -0.47
  H  -0.75   0.00  -0.47
end

basis
  * library 6-31G
end

task scf optimize
```

echo

Copies input into output.

start <prefix>

Names job, creates/overwrites existing files.

geometry and basis

Basic input blocks.

task <method> <task>

Initiates a calculation.

Input File Structure

```
echo

start water

geometry
  O   0.00   0.00   0.12
  H   0.75   0.00  -0.47
  H  -0.75   0.00  -0.47
end

basis
  * library 6-31G
end

task scf optimize
```

echo

Copies input into output.

start <prefix>

Names job, creates/overwrites existing files.

geometry and basis

Basic input blocks.

task <method> <task>

Initiates a calculation.

Geometry Input

Geometry Input — Cartesian

```
geometry units angstroms autosym
  O  0.00  0.00  0.12
  H  0.75  0.00 -0.47
  H -0.75  0.00 -0.47
end
```

Other keywords of interest

`noautosym`, `units <bohr, pm, nm, atomic>`

Geometry Input — Cartesian

```
geometry units angstroms autosym
  O  0.00  0.00  0.12
  H  0.75  0.00 -0.47
  H -0.75  0.00 -0.47
end
```

Other keywords of interest

`noautosym`, `units <bohr, pm, nm, atomic>`

Geometry Input — Simple Z-Matrix

```
geometry units angstrom
 symmetry c1
 zmatrix
  O
  H1 O 0.9572
  H2 O 0.9572 H1 104.52
 end
end
```

Geometry Input — Detailed Z-Matrix

```
geometry units angstrom autosym
 zmatrix
  O
  H1 O r
  H2 O r H1 theta
  constants
   theta 104.0
  variables
   r 0.95
 end
end
```

Geometry Input — Constrained Internal Coordinates

```
geometry units angstroms autosym
  O  0.00  0.00  0.12
  H  0.75  0.00 -0.47
  H -0.75  0.00 -0.47
  zcoord
    angle 2 1 3 90.0 theta constant
  end
end
```

Geometry Input — Forcing Symmetry

TCE only handles subgroups of D_{2h}, hence you need to forcibly reduce symmetry:

```
geometry units angstroms autosym
  symmetry d2h
  Ne 0.0 0.0 0.0
end
```

On the other hand, using symmetry "fill in" helps to construct e.g. C_{60}.

Geometry Input — Other Considerations

Consult the documentation regarding:

- symmetry group keywords
- adding point charges
- periodic geometries
- changing mass, charge and extent of nuclei
- naming geometries (useful for e.g. BSSE)
- other obscure options

Basis Set Input

Basis Set Input — Using the Library

```
basis cartesian
 * library 6-31G
end

basis spherical
 C library cc-pVQZ
 H library cc-pVTZ
end
```

spherical or cartesian

Pople → cartesian.

Dunning → spherical.

library

Specify all or by element.

NWChem library is extensive but missing some basis sets present in Gaussian.

Basis Set Input — Using the Library

```
basis cartesian
 * library 6-31G
end

basis spherical
 C library cc-pVQZ
 H library cc-pVTZ
end
```

spherical or cartesian

Pople → cartesian.
Dunning → spherical.

library

Specify all or by element.

NWChem library is extensive but missing some basis sets present in Gaussian.

Basis Set Input — Using the Library

```
basis cartesian
 * library 6-31G
end

basis spherical
 C library cc-pVQZ
 H library cc-pVTZ
end
```

`spherical` or `cartesian`

Pople → `cartesian`.
Dunning → `spherical`.

`library`

Specify all or by element.

NWChem library is extensive but missing some basis sets present in Gaussian.

Basis Set Input — Explicit Definition

```
basis
  H S
   33.8650140 0.0060680
    5.0947880 0.0453160
  H S
    0.1027410 1.0000000
  H P
    1.1588000 0.1884400
    0.3258000 0.8824200
end
```

Cut-and-paste from bse.pnl.gov/bse/portal!

Basis Set Input — RI-MP2 Fitting Basis Sets

```
basis "ao basis" spherical
 * library aug-cc-pVTZ
end

basis "ri-mp2 basis" spherical
 * library cc-pVTZ-RI
 * library aug-cc-pVTZ-RI_diffuse
end
```

This is how you activate RI-MP2 in NWChem.

Basis Set Input — DFT Fitting Basis Sets

```
basis "ao basis" cartesian
 * library 6-31+G*
end

basis "cd basis" spherical
 * library "Ahlrichs Coulomb Fitting"
end
```

When to Use Density-Fitting

Useful for GGA functionals (e.g. PBE).
Less useful for hybrid DFT (e.g. B3LYP).

Basis Set Input — DFT Fitting Basis Sets

```
basis "ao basis" cartesian
 * library 6-31+G*
end

basis "cd basis" spherical
 * library "Ahlrichs Coulomb Fitting"
end
```

When to Use Density-Fitting

Useful for GGA functionals (e.g. PBE).
Less useful for hybrid DFT (e.g. B3LYP).

Basis Set Input — Using ECPs

```
basis
 Ca library "lanl2dz ecp"
 F library "6-31g"
 H library "sto-3g"
end

ecp
 Ca library "lanl2dz ecp"
end
```

Task Input

Task Input — What is Available?

- energy
- gradient/optimize/saddle/dynamics
- hessian/frequencies
- property
- dplot
- *python*
- *shell*

Method Input

Method Input — What is Available?

- SCF/DFT energy, gradient, hessian, property
- TDDFT excited-state energy
- MP2 energy and gradient (semidirect algorithm)
- DHDF and RI-MP2 energy
- CCSD/CCSD(T) energy (direct algorithm)
- TCE energy (+properties), gradient (sort-of)
- MCSCF energy, gradient and property

SCF Input

SCF Input — Disk Usage

```
scf
  semidirect memsize <words> filesize 0
end
```

```
scf
  direct
end
```

semidirect

Use memory but not disk.

direct

Compute integrals on-the-fly.

A word is 8 bytes. Divide stack (in bytes) by 10.

SCF Input — Disk Usage

```
scf
  semidirect memsize <words> filesize 0
end
```

```
scf
  direct
end
```

semidirect

Use memory but not disk.

direct

Compute integrals on-the-fly.

A word is 8 bytes. Divide stack (in bytes) by 10.

SCF Input — Disk Usage

```
scf
  semidirect memsize <words> filesize 0
end
```

```
scf
  direct
end
```

semidirect

Use memory but not disk.

direct

Compute integrals on-the-fly.

A word is 8 bytes. Divide stack (in bytes) by 10.

SCF Input — Basic MOVEC I/O

This is the default behavior:

```
scf
  vectors input atomic output <prefix>.movecs
end
```

Read input guess:

```
scf
  vectors input <file>
end
```

For geometry optimizations and restart, the second option is automatic.

You can use SCF, DFT and MCSCF movecs interchangeably.

SCF Input — Advanced MOVEC I/O

```
basis "small" ; * library 3-21G ; end
basis "large" ; * library cc-pVTZ ; end

set "ao basis" "small"
scf
  vectors output <file>
end
task scf

set "ao basis" "large"
scf
  vectors input project "small" <file>
end
task scf
```

DFT Input

DFT Input — Predefined Functionals

NWChem defines dozens of functionals.
See documentation for a complete list.

```
dft
  xc <functional>
end
```

Popular choices: `b3lyp`, `pbe0`, `m06`.

DFT Input — Custom Functionals

NWChem permits any linear combination of exchange-correlation functionals:

This is PBE:

```
dft
  xc xpbe96 1.0 \
     pw91lda local 1.0 \
     cpbe96 nonlocal 1.0
end
```

DFT Input — Double-Hybrid Functionals

Adding `mp2` as a correlation functional leads to double-hybrid functionals.

This is B2PLYP:

```
dft
  xc HFexch 0.53 \
     becke88 0.47 \
     lyp 0.73 \
     mp2 0.27
  dftmp2 <semidirect, direct, ri>
end
```

Define RI basis if `dftmp2 ri` used.

DFT Input — Disk Usage

```
dft
  direct
  noio
end
```

direct

USE THIS OPTION!!!

noio

This option is pointless.

DFT Input — Disk Usage

```
dft
  direct
  noio
end
```

direct

USE THIS OPTION!!!

noio

This option is pointless.

DFT Input — Disk Usage

```
dft
  direct
  noio
end
```

direct

USE THIS OPTION!!!

noio

This option is pointless.

DFT Input — Convergence

1. default (atomic density guess)
2. if hybrid, project from HF vectors
3. run with tiny basis and project
4. use `smear` if metallic
5. email nwchem-users@emsl.pnl.gov

Do not waste your time with convergence voodoo!

SCF solver is often better than the DFT solver.

MP2 Input

MP2 Input — Important Options

```
mp2
  freeze atomic
  tight
  scratchdisk <limit>
end
```

freeze atomic

Freeze core.

tight

Use with gradients.

scratchdisk

The semidirect approach will make multiple passes if disk is limited.

MP2 Input — Important Options

```
mp2
  freeze atomic
  tight
  scratchdisk <limit>
end
```

freeze atomic

Freeze core.

tight

Use with gradients.

scratchdisk

The semidirect approach will make multiple passes if disk is limited.

MP2 Input — Important Options

```
mp2
  freeze atomic
  tight
  scratchdisk <limit>
end
```

freeze atomic

Freeze core.

tight

Use with gradients.

scratchdisk

The semidirect approach will make multiple passes if disk is limited.

MP2 Input — Important Options

```
mp2
  freeze atomic
  tight
  scratchdisk <limit>
end
```

freeze atomic

Freeze core.

tight

Use with gradients.

scratchdisk

The semidirect approach will make multiple passes if disk is limited.

CCSD(T) Input

CCSD(T) Input — Important Options

```
ccsd
  freeze atomic
  maxiter 20
  thresh 1e-6
  nodisk
end
```

Method Input

```
task ccsd
task ccsd(t)
```

freeze atomic

Freeze core.

nodisk

Recompute integrals each iteration.

Symmetry

Most of this code does not use symmetry.

CCSD(T) Input — Important Options

```
ccsd
  freeze atomic
  maxiter 20
  thresh 1e-6
  nodisk
end
```

Method Input

```
task ccsd
task ccsd(t)
```

freeze atomic

Freeze core.

nodisk

Recompute integrals each iteration.

Symmetry

Most of this code does not use symmetry.

CCSD(T) Input — Important Options

```
ccsd
  freeze atomic
  maxiter 20
  thresh 1e-6
  nodisk
end
```

Method Input

```
task ccsd
task ccsd(t)
```

freeze atomic

Freeze core.

nodisk

Recompute integrals each iteration.

Symmetry

Most of this code does not use symmetry.

CCSD(T) Input — Important Options

```
ccsd
  freeze atomic
  maxiter 20
  thresh 1e-6
  nodisk
end
```

Method Input

```
task ccsd
task ccsd(t)
```

freeze atomic

Freeze core.

nodisk

Recompute integrals each iteration.

Symmetry

Most of this code does not use symmetry.

CCSD(T) Input — Important Options

```
ccsd
  freeze atomic
  maxiter 20
  thresh 1e-6
  nodisk
end
```

Method Input

```
task ccsd
task ccsd(t)
```

freeze atomic

Freeze core.

nodisk

Recompute integrals each iteration.

Symmetry

Most of this code does not use symmetry.

TCE Input

Disclaimer

TCE input is cryptic. Do not attempt to build input files from scratch using the documentation. Start from the examples in the QA directory or search the user list archives, especially as it pertains to maximizing performance.

TCE Input — Method Input

```
tce; <method>; end
```

method

CC2, QCISD, LCCSD, CCSD, CCSDT, CCSDTQ, CISD, CISDT, CISDTQ, CCSD(T), CCSD(2)_T, CCSD(2), CR-CCSD(T), CREOMSD(T), CCSDT(2)_Q, MP2, MP3, MP4, MP4SDQ(T), ...

Useful methods

CR-CCSD(T) breaks bonds, CCSD and CREOMSD(T) for excited-states, MP4SDQ(T) for G4 and CCSD(T) otherwise.

TCE Input — Method Input

```
tce; <method>; end
```

method

CC2, QCISD, LCCSD, CCSD, CCSDT, CCSDTQ, CISD, CISDT, CISDTQ, CCSD(T), CCSD(2)_T, CCSD(2), CR-CCSD(T), CREOMSD(T), CCSDT(2)_Q, MP2, MP3, MP4, MP4SDQ(T), ...

Useful methods

CR-CCSD(T) breaks bonds, CCSD and CREOMSD(T) for excited-states, MP4SDQ(T) for G4 and CCSD(T) otherwise.

TCE Input — Method Input

```
tce; <method>; end
```

method

CC2, QCISD, LCCSD, CCSD, CCSDT, CCSDTQ, CISD, CISDT, CISDTQ, CCSD(T), CCSD(2)_T, CCSD(2), CR-CCSD(T), CREOMSD(T), CCSDT(2)_Q, MP2, MP3, MP4, MP4SDQ(T), ...

Useful methods

CR-CCSD(T) breaks bonds, CCSD and CREOMSD(T) for excited-states, MP4SDQ(T) for G4 and CCSD(T) otherwise.

TCE Input — Algorithm Options

```
tce
  io ga
  2eorb
  2emet 13
  tilesize <n>
end
```

io ga

ALWAYS USE THIS!

2eorb/2emet

Use if RHF/ROHF.

tilesize

n=32 for methods ending in SD.
n=20 for methods ending in (T).

TCE Input — Algorithm Options

```
tce
  io ga
  2eorb
  2emet 13
  tilesize <n>
end
```

io ga

ALWAYS USE THIS!

2eorb/2emet

Use if RHF/ROHF.

tilesize

n=32 for methods ending in SD.
n=20 for methods ending in (T).

TCE Input — Algorithm Options

```
tce
  io ga
  2eorb
  2emet 13
  tilesize <n>
end
```

io ga

ALWAYS USE THIS!

2eorb/2emet

Use if RHF/ROHF.

tilesize

n=32 for methods ending in SD.
n=20 for methods ending in (T).

TCE Input — Algorithm Options

```
tce
  io ga
  2eorb
  2emet 13
  tilesize <n>
end
```

io ga

ALWAYS USE THIS!

2eorb/2emet

Use if RHF/ROHF.

tilesize

n=32 for methods ending in SD.
n=20 for methods ending in (T).

TCE Input

Warning

If you do not heed the advice to follow, you shall be doomed to endless misery. Your jobs will fail frequently and sysadmins everywhere will curse your existence.

Memory Input

`memory stack <S> heap <H> global <G>`

Each of these allocations is **PER PROCESS**!

`global` is for GA, `stack` is local.

`heap` usage appears to be system-size invariant.

`stack+heap` is statically allocated at boot.

`global` is allocated on-demand.

Memory Input

1. Divide memory per node by process per node.
2. OS and MPI/ARMCI use 100-200 MB.
3. Set `heap` to 100 MB.

4a. If SCF/DFT/MP2/*(T), set `stack` to 50% of total.

4b. If *SD, set `stack` = `global`.

Example — DFT on Fusion

```
memory stack 2000 mb heap 100 mb global 1000 mb
```

Example — CCSD on Fusion

```
memory stack 1500 mb heap 100 mb global 1500 mb
```

Memory Input

1. Divide memory per node by process per node.
2. OS and MPI/ARMCI use 100-200 MB.
3. Set `heap` to 100 MB.

4a. If SCF/DFT/MP2/*(T), set `stack` to 50% of total.

4b. If *SD, set `stack` = `global`.

Example — DFT on Fusion

```
memory stack 2000 mb heap 100 mb global 1000 mb
```

Example — CCSD on Fusion

```
memory stack 1500 mb heap 100 mb global 1500 mb
```

Memory Input

1. Divide memory per node by process per node.
2. OS and MPI/ARMCI use 100-200 MB.
3. Set `heap` to 100 MB.

4a. If SCF/DFT/MP2/*(T), set `stack` to 50% of total.

4b. If *SD, set `stack` = `global`.

Example — DFT on Fusion

```
memory stack 2000 mb heap 100 mb global 1000 mb
```

Example — CCSD on Fusion

```
memory stack 1500 mb heap 100 mb global 1500 mb
```

Running Jobs on Fusion

Memory Input

Simple examples from the tutorial:
`/soft/nwchem/tutorial`

Complete with PBS for each compiler:
`/soft/nwchem/examples`

I will upload or create examples upon request.