cclib: Program-agnostic quantum chemical parsing, representation, and analysis

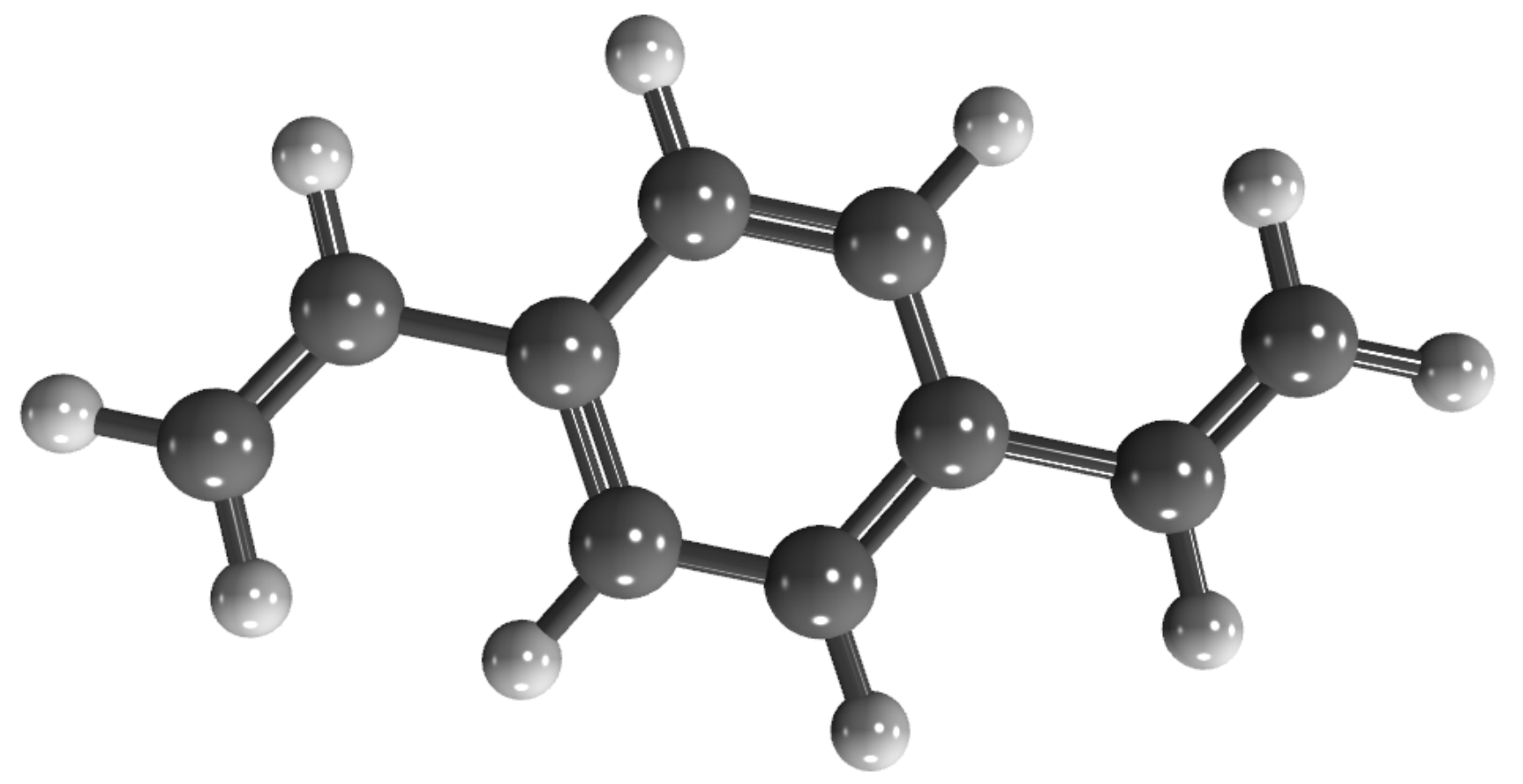
Eric Berquist, Geoff Hutchison, Karol Langner, Noel O’Boyle, Adam Tenderholt

erb74@pitt.edu

Go to:
- https://cclib.github.io to see our documentation
- https://github.com/cclib/cclib to see our code
- `pip install cclib` to install the latest stable version

Testing

Development is driven by extensive testing suites:
- Unit tests for each calculation type run for multiple versions of each quantum program on divinylbenzene (DVB, C_{2h}, below)
- Regression tests from our own research (https://github.com/cclib/cclib-data)



Available Properties

- Atoms: coordinates, partial charges, spin densities
- Energies: SCF, MP, CC
- Electronic excitations (energies, osc. strengths, ... from CIS/TD-DFT)
- Molecular orbitals (eigenvalues, coefficients)
- (Nuclear) Gradients, Hessians
- Vibrations (frequencies, IR intensities, Raman intensities, displacements)
- Potential energy scans (energies, coordinates)
- Multipole moments (arbitrary order)

Available Methods

- Population analysis (C squared, Mulliken, Löwdin, overlap)
- Density matrix calculation
- Isosurface generation and integration (molecular orbitals and densities)
- Charge decomposition analysis
- Mayer bond orders
- Fragment analysis (with custom fragments)

Available Programs

ADF
Dalton
GAMESS-UK
GAMESS-US
Gaussian
Jaguar
Molcas
Molpro
MOPAC
NWChem
ORCA
Psi4
Q-Chem
Turbomole

Reference

N. M. O’Boyle, A. L. Tenderholt, K. M. Langner, *cclib: a library for package-independent computational chemistry algorithms*, J. Comp. Chem. 29 (5), pp. 839-845, **2008**

Examples

Calculate the AO-basis density matrix

```
                     RESTRICTED (RHF) MOLECULAR ORBITAL COEFFICIENTS
                           1         2         3         4         5         6
eigenvalues:         -10.018   -10.018   -10.008   -10.008   -10.007   -10.007
  1  C 1   s         0.69897   0.69891  -0.03421   0.03463  -0.02640   0.01449
  2  C 1   s         0.03190   0.03167   0.00338  -0.00331   0.00776  -0.00335
  3  C 1   px       -0.00138  -0.00104   0.00408  -0.00405  -0.00329   0.00221
  4  C 1   py       -0.00028  -0.00021   0.00066  -0.00062   0.00184  -0.00464
  5  C 1   pz       -0.00000  -0.00000   0.00000   0.00000   0.00000  -0.00000
  6  C 2   s        -0.01980  -0.02107   0.01419  -0.02036  -0.66726   0.65727
  7  C 2   s        -0.00816  -0.00729   0.00088  -0.00104  -0.03195   0.03156
  8  C 2   px       -0.00268  -0.00308  -0.00002  -0.00005   0.00180  -0.00185
```

$$\psi_i = \sum_{\mu}^{\mathrm{AO}} C_{\mu i}\phi_{\mu}$$

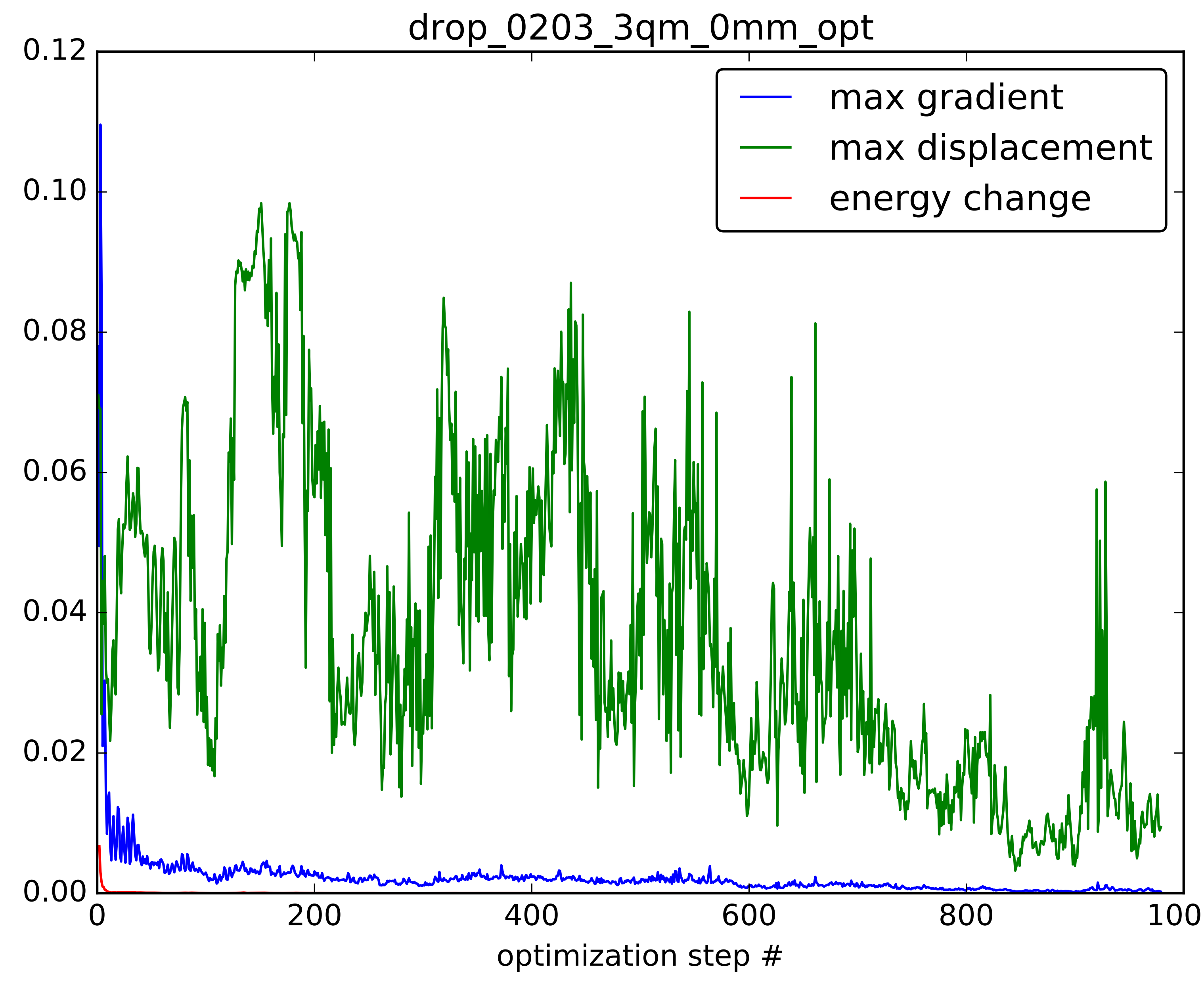
$$P_{\mu\nu} = \sum_{i}^{\mathrm{d.o.}} C_{\mu i}C_{\nu i}$$

```
$ ccget mocoeffs dvb_sp.out
Attempting to read dvb_sp.out
mocoeffs:
[array([[  6.98968000e-01,   3.18974000e-02,  -1.38370000e-03, ...,
          3.77600000e-04,  -3.08000000e-05,  -1.49000000e-04],
       [  6.98911600e-01,   3.16727000e-02,  -1.03560000e-03, ...,
          3.91900000e-04,  -2.97000000e-05,  -1.36300000e-04],
       [ -3.42051000e-02,   3.37620000e-03,   4.08370000e-03, ...,
          6.36590000e-03,  -2.02800000e-04,  -1.70800000e-04],
       ...,
```

```
# Need to do this...
C = data.mocoeffs[0].T
# ...so we can write
# the correct equation:
P = np.dot(C[:, :nocc],
           C[:, :nocc].T)
```

Plot geometry optimization convergence

```
data = ccread(compchemfilename)

fig, ax = plt.subplots()

if type(job) == cclib.parser.qchemparser.QChem:

    scfenergies = [utils.convertor(scfenergy, 'eV', 'hartree') for
scfenergy in data.scfenergies]
    gradients = [geovalue[0] for geovalue in data.geovalues]
    displacements = [geovalue[1] for geovalue in data.geovalues]
    energy_changes = [(geovalue[2] * args.scaling_energy_change) for
geovalue in data.geovalues]

    # If this isn't true, something funny happened during the
    # parsing, so fail out.
    assert len(scfenergies) == len(gradients)

    steps = range(1, len(scfenergies) + 1)

    # ax.plot(steps, scfenergies, label='SCF energy')
    ax.plot(steps, gradients, label='max gradient')
    ax.plot(steps, displacements, label='max displacement')
    ax.plot(steps, energy_changes, label='energy change')

    ax.set_title(stub)
    ax.set_xlabel('optimization step #')
```



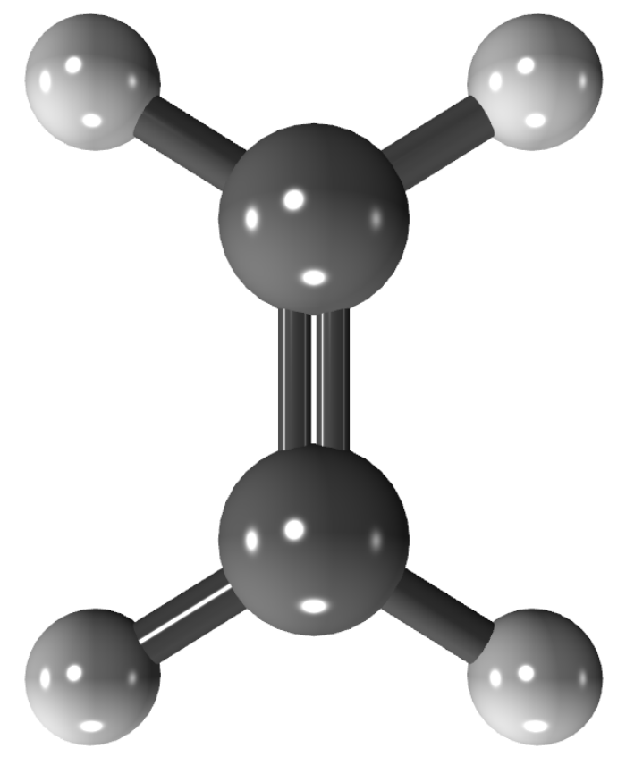
Basic Usage

Q-Chem TD-DFT input

```
$rem
 jobtype = sp
 method = b3lyp
 basis = 6-311++g(d,p)
 cis_n_roots = 10
 cis_singlets = true
 cis_triplets = true
 rpa = false
$end

$molecule
0 1
C   0.6680  0.0000  0.0000
C  -0.6680  0.0000  0.0000
H   1.2437  0.9222  0.0000
H  -1.2437  0.9222  0.0000
H   1.2437 -0.9222  0.0000
H  -1.2437 -0.9222  0.0000
$end
```

Quickly check what was parsed:

```
$ ccget --list --verbose qchem_tddft.out
Attempting to read qchem_tddft.out
Identified logfile to be in QChem format
cclib can parse the following attributes
from qchem_tddft.out:
  atomcharges
  atomcoords
  atomnos
  charge
  coreelectrons
  etenergies
  etoscs
  etsecs
  etsyms
  homos
  moenergies
  moments
  mosyms
  mult
  natom
  nbasis
  nmo
  scfenergies
  scftargets
  scfvalues
```



Q-Chem TD-DFT output

```
 ---------------------------------------------------
        TDDFT/TDA Excitation Energies
 ---------------------------------------------------

 Excited state   1: excitation energy (eV) =    4.4006
    Total energy for state   1:    -78.453711566821
    Multiplicity: Triplet
    Trans. Mom.:  0.0000 X   0.0000 Y   0.0000 Z
    Strength   :     0.0000000000
    D(  8) --> V(  1) amplitude =  0.9798

 Excited state   2: excitation energy (eV) =    6.6174
    Total energy for state   2:    -78.372245053445
    Multiplicity: Triplet
    Trans. Mom.:  0.0000 X   0.0000 Y   0.0000 Z
    Strength   :     0.0000000000
    D(  8) --> V(  2) amplitude =  0.9928

 Excited state   3: excitation energy (eV) =    6.7062
    Total energy for state   3:    -78.368981782723
    Multiplicity: Singlet
    Trans. Mom.: -0.0000 X   0.0000 Y   0.5940 Z
    Strength   :     0.0579747854
    D(  8) --> V(  2) amplitude =  0.9974

 Excited state   4: excitation energy (eV) =    7.2024
    Total energy for state   4:    -78.350745871569
    Multiplicity: Triplet
    Trans. Mom.:  0.0000 X   0.0000 Y   0.0000 Z
    Strength   :     0.0000000000
    D(  8) --> V(  3) amplitude =  0.9933
...
```

```
from cclib.io import ccread
data = ccread("qchem_tddft.out")
print(data.etenergies)
print(data.etoscs)
print(data.etsecs)
print(data.etsyms)
```

Printing the Python representation:

```
$ ccget etenergies etoscs etsecs etsyms qchem_tddft.out
Attempting to read qchem_tddft.out
etenergies:
[ 35493.06025806  53372.89325032  54089.09838909
58091.41826732
  58390.4607959   58799.84817293  58874.78149699
60404.37174076
  63297.70711771  64565.8298838   66393.71048982
67291.30174949
  67371.4463312   69681.37586254  70505.03004838
70917.07881868
  71082.02827352  71852.43337395  72466.75468411
76593.78450849]
etoscs:
[ 0.          0.          0.05797479  0.          0.
0.          0.
  0.          0.44719476  0.          0.          0.
0.          0.
  0.          0.          0.00136019  0.          0.
0.09169852]
etsecs:
[[[(7, 0), (8, 0), 0.9798]], [[(7, 0), (9, 0), 0.9928]],
[[(7, 0), (9, 0), 0.9974]], [[(7, 0), (10, 0), 0.9933]],
[[(6, 0), (8, 0), 0.2461], [(7, 0), (11, 0), 0.9581]],
[[(7, 0), (10, 0), 0.9974]], [[(7, 0), (11, 0), 0.9921]],
[[(6, 0), (8, 0), 0.9609], [(7, 0), (11, 0), -0.2504]],
[[(7, 0), (8, 0), 0.9242], [(7, 0), (16, 0), 0.3075]],
[[(6, 0), (8, 0), 0.9937]], [[(7, 0), (13, 0), 0.9984]],
[[(7, 0), (12, 0), 0.9952]], [[(7, 0), (12, 0), 0.9968]],
[[(7, 0), (13, 0), 0.9826]], [[(6, 0), (9, 0), 0.986]],
[[(7, 0), (14, 0), 0.9939]], [[(7, 0), (14, 0), 0.9986]],
[[(6, 0), (9, 0), 0.9972]], [[(5, 0), (8, 0), -0.986]],
[[(6, 0), (10, 0), 0.9973]]]
etsyms:
['Triplet', 'Triplet', 'Singlet', 'Triplet', 'Triplet',
'Singlet', 'Singlet', 'Triplet', 'Singlet', 'Singlet',
'Triplet', 'Triplet', 'Singlet', 'Singlet', 'Triplet',
'Triplet', 'Singlet', 'Singlet', 'Triplet', 'Singlet']
```

Future Work

In progress:
- MCSCF (CASSCF, RASSCF) attributes: energies, state information (CSF weights, occupations, densities)
- NMR spectroscopy (shielding tensors and principal values)
- Symmetry handling (+ `libmsym` integration)
- QCJSON generation (github.com/MolSSI/QC_JSON_Schema)

Desired:
- CFOUR/ACES parsers
- CI energies and excitation methods beyond TD-HF/TD-DFT
- MP/CI/CC amplitudes
- EPR spectroscopy (hyperfine tensors, *g*-tensors)
- Nonlinear spectroscopies (first hyperpolarizability, ...)
- Fragments (ALMO, SAPT, CP for BSSE, MBE, FMO, ...)