CASSCF
&
psi4numpy

Daniel Smith

CASSCF

- Master branch:
  - Psi3 DETCAS module ported to Psi4
  - CASSCF and SA-CASSCF test cases

CASSCF

- Master branch:
  - Psi3 DETCAS module ported to Psi4
  - CASSCF and SA-CASSCF test cases
- CASSCF branch:
  - DETCAS and DETCI are now merged
- Future:
  - 1-step CASSCF
  - Linear response

PSI4NUMPY

- Numpy (Numerical Python):
  - Written in C++
  - Base class: N-dimensional array
  - Arbitrary tensor manipulation (einsum)
  - Accesses system BLAS
- Psi Interface:
  - Matrix and Vector classes
  - MintsHelper and NumpyHelper

RHF.dat

Import Numpy —>

Set Molecule —>

Grab Integrals —>

Core guess —>

Roothan loops —>

Build Fock Matrix —>

Check convergence —>

Diagonalize Fock Matrix —>

Compare to Psi4 —>

```
import numpy as np
from scipy import linalg as SLA
np.set_printoptions(precision=5, linewidth=200, suppress=True)

molecule mol {
O
H 1 1.1
H 1 1.1 2 104
symmetry c1
}

set {
basis cc-pVDZ
scf_type pk
e_convergence 1e-8
}

mints = MintsHelper()
T = mints.ao_potential()
V = mints.ao_kinetic()
I = np.array(I).reshape(nbf, nbf, nbf, nbf)

# Build H core
H = np.matrix(T) + np.matrix(V)
# Orthogonalizer A = S^(-1/2)
A = np.matrix(SLA.sqrtm(S)).I.real

# Calculate initial core guess
Hp = A * H * A
e, C2 = SLA.eigh(Hp)
C = A*C2
Cocc = C[:, :ndocc]
D = Cocc * Cocc.T

print('\nStart RHF iterations:\n')
E = 0.0
Enuc = mol.nuclear_repulsion_energy()
Eold = 0.0

for SCF_ITER in range(1, maxiter + 1):

    # Build fock matrix
    J = np.einsum('pqrs,rs->pq', I, D)
    K = np.einsum('pqrs,qs', I, D)
    F = H + J*2 - K

    # SCF energy and update
    SCF_E = np.einsum('ij,ij->', F+H, D) + Enuc

    print 'RHF Iteration %3d: Energy = %4.16f   dE = % 1.5E   dRMS = %1.5E' %
                                        (SCF_ITER, SCF_E, (SCF_E - Eold), dRMS)
    if (abs(SCF_E - Eold) < E_conv):
        break

    Eold = SCF_E
    Dold = D

    # Diagonalize Fock matrix
    Fp = A*F*A
    e, C2 = SLA.eigh(Fp)
    C = A*C2
    Cocc = C[:, :ndocc]
    D = Cocc*Cocc.T

print 'Final SCF energy: %.8f hartree' % SCF_E
SCF_E_psi = energy('scf')
compare_values(SCF_E_psi, SCF_E, E_conv, 'SCF Energy')
```

PSI4NUMPY – SCF

- Form H_{core}

```
mints = MintsHelper()
T = mints.ao_potential()
V = mints.ao_kinetic()
H = np.array(T) + np.array(V)
```

- Build Fock matrix

```
J = np.einsum('pqrs,rs->pq', I, D)
K = np.einsum('pqrs,qs', I, D)
F = H + J*2 - K
```

- Run it! (H_2O RHF/6-31G)

```
RHF Iteration  24: Energy = -75.9525284810213748   dE = -6.68640E-07   dRMS = 1.3
RHF Iteration  25: Energy = -75.9525287868844430   dE = -3.05863E-07   dRMS = 8.8
Total time taken for SCF iterations: 0.027 seconds

Final SCF energy: -75.95252879 hartree
        SCF Energy.....................................................PASSED
```

PSI4NUMPY – POST HF

- Use Psi4 wavefunction object

```
energy('RHF')
wfn = wavefunction()
C = np.array(wfn.Ca())
```

PSI4NUMPY – POST SCF

- Use Psi4 wavefunction object

```
energy('RHF')
wfn = wavefunction()
C = np.array(wfn.Ca())
```

- CCSD

$$W_{mbej} = \langle mb||ej\rangle + \sum_f t_j^f \langle mb||ef\rangle - \sum_n t_n^b \langle mn||ej\rangle - \sum_{nf} (\frac{1}{2} t_{jn}^{fb} + t_j^f t_n^b) \langle mn||ef\rangle$$

```
def build_Wmbej(t1, t2):
    Wmbej = MO[o, v, v, o].copy()
    Wmbej += np.einsum('jf,mbef->mbej', t1, MO[o, v, v, v])
    Wmbej -= np.einsum('nb,mnej->mbej', t1, MO[o, o, v, o])

    tmp = (0.5 * t2) + np.einsum('jf,nb->jnfb', t1, t1)
    Wmbej -= np.einsum('jnfb,mnef->mbej', tmp, MO[o, o, v, v])
    return Wmbej
```

PSI4NUMPY

Current

- SCF
- MP2, MP3, MPn
- Electron Propagator
- CCSD(T)
- CIS
- SAPT0

Beta

- Density Fitting
- DFT
- Multireference
- Response functions

github.com/dgasmith/psi4numpy