Configuration Interaction Singles
Part 2

March 12, 2018

- Symmetric matrix diagonalization

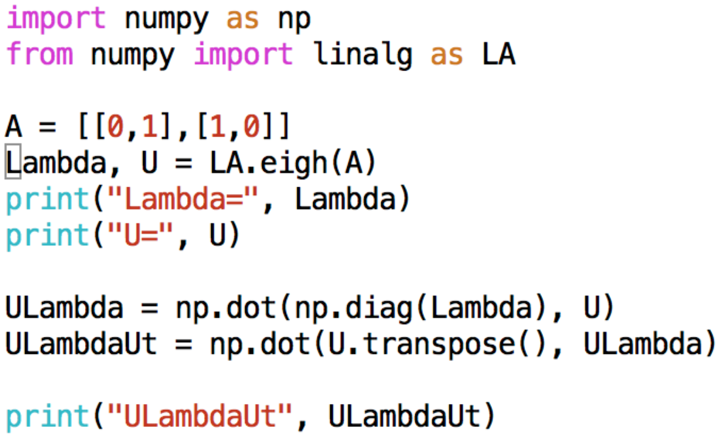
$$\mathbf{A}\mathbf{U} = \mathbf{U}\lambda$$

- 2-by-2 symmetric matrix

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

- Example

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$



```python
import numpy as np
from numpy import linalg as LA

A = [[0,1],[1,0]]
Lambda, U = LA.eigh(A)
print("Lambda=", Lambda)
print("U=", U)

ULambda = np.dot(np.diag(Lambda), U)
ULambdaUt = np.dot(U.transpose(), ULambda)

print("ULambdaUt", ULambdaUt)
```

Singlet-Reference Methods

- Singlet-reference methods for electron excited states include:
  - Configuration Interaction Singlets (CIS)
  - Time–Dependent Density Functional Theory (TDDFT)

Singlet-Reference Methods

- Singlet-reference methods for electron excited states include:
  - Configuration Interaction Singlets (CIS)
  - Time–Dependent Density Functional Theory (TDDFT)

Ground-State: Hartree-Fock DFT

Singlet-Reference Methods

- Singlet-reference methods for electron excited states include:
  - Configuration Interaction Singlets (CIS)
  - Time–Dependent Density Functional Theory (TDDFT)

$$\begin{tikzcd}
\text{Ground-State:} & \text{Hartree-Fock} \arrow[d, Rightarrow] & \text{DFT} \arrow[d, Rightarrow] \\
\text{Excited-States:} & \text{CIS} & \text{TDDFT}
\end{tikzcd}$$

Singlet-Reference Methods

- Singlet-reference methods for electron excited states include:
  - Configuration Interaction Singlets (CIS)
  - Time–Dependent Density Functional Theory (TDDFT)

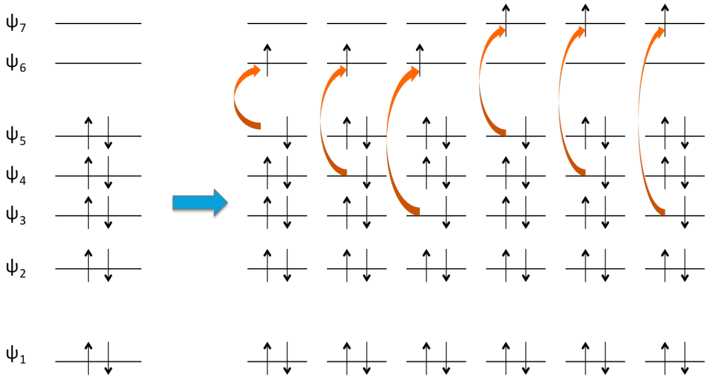
$$\begin{tikzcd}
\text{Ground-State:} & \text{Hartree-Fock} \arrow[d, Rightarrow] & \text{DFT} \arrow[d, Rightarrow] \\
\text{Excited-States:} & \text{CIS} & \text{TDDFT}
\end{tikzcd}$$

- The CIS wavefunction can be written as

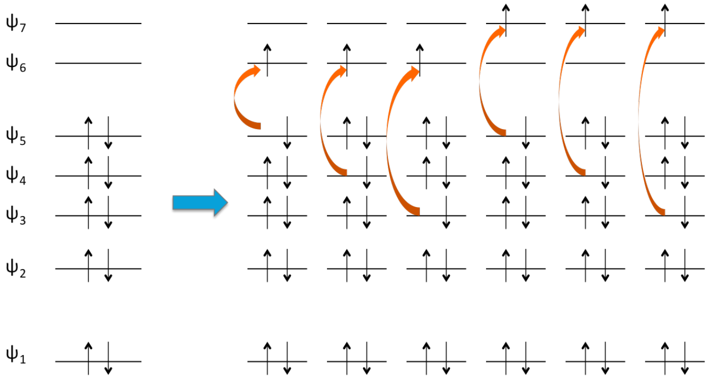
$$\Psi_{\text{CIS}} = \sum_{ai} X_{ai} \Phi_i^a$$

where X_{ai} are called excitation amplitudes.

Configuration Interaction Singles (CIS)



Configuration Interaction Singles (CIS)

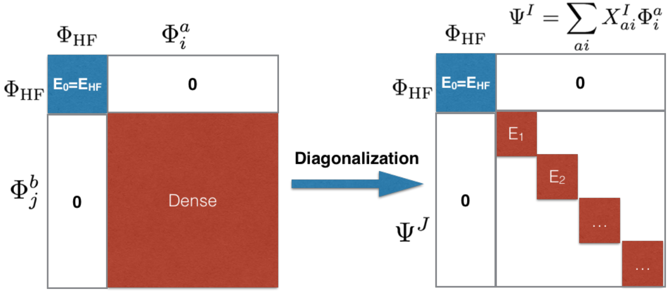


$$\Psi_{\text{CIS}}^{\text{singlet}} = \sum_{ai} X_{ai} \left[\frac{1}{\sqrt{2}} \left(\Phi_i^a + \Phi_{\bar{i}}^{\bar{a}} \right) \right] \qquad \Psi_{\text{CIS}}^{\text{triplet}} = \sum_{ai} X_{ai} \left[\frac{1}{\sqrt{2}} \left(\Phi_i^a - \Phi_{\bar{i}}^{\bar{a}} \right) \right]$$

- The CIS wavefunction

$$\Psi_{\text{CIS}} = \sum_{ai} X_{ai} \Phi_i^a$$

are obtained by diagonalizing the Hamiltonian in the subspace spanned by all singly-excited configurations.



- Given the Hamiltonian,

$$\hat{H} = \sum_i \left(-\frac{1}{2} \nabla_i^2 - \sum_A \frac{Z_A}{\left| \vec{r}_i - \vec{R}_A \right|} \right) + \sum_{i<j} \frac{1}{|\vec{r}_i - \vec{r}_j|}$$

- The Hamiltonian matrix elements

$$\left\langle \Phi_i^a \middle| \hat{H} \middle| \Phi_j^b \right\rangle = E_0 + (\varepsilon_a - \varepsilon_i)\, \delta_{ij}\delta_{ab} + (ai|bj) - (ab|ij)$$

- Given the Hamiltonian,

$$\hat{H} \quad = \quad \sum_i \left(-\frac{1}{2} \nabla_i^2 - \sum_A \frac{Z_A}{\left| \vec{r}_i - \vec{R}_A \right|} \right) + \sum_{i<j} \frac{1}{|\vec{r}_i - \vec{r}_j|}$$

- The Hamiltonian matrix elements

$$\left\langle \Phi_i^a \middle| \hat{H} \middle| \Phi_j^b \right\rangle = E_0 + (\varepsilon_a - \varepsilon_i)\, \delta_{ij}\delta_{ab} + (ai|bj) - (ab|ij)$$

- For singlet configurations

$$\left\langle \frac{\Phi_i^a + \Phi_{\bar{i}}^{\bar{a}}}{\sqrt{2}} \middle| \hat{H} \middle| \frac{\Phi_j^b + \Phi_{\bar{j}}^{\bar{b}}}{\sqrt{2}} \right\rangle = E_0 + (\varepsilon_a - \varepsilon_i)\, \delta_{ij}\delta_{ab} + 2(ai|bj) - (a2b|ij)$$

- Given the Hamiltonian,

$$\hat{H} = \sum_i \left(-\frac{1}{2} \nabla_i^2 - \sum_A \frac{Z_A}{\left| \vec{r}_i - \vec{R}_A \right|} \right) + \sum_{i<j} \frac{1}{|\vec{r}_i - \vec{r}_j|}$$

- The Hamiltonian matrix elements

$$\left\langle \Phi_i^a \middle| \hat{H} \middle| \Phi_j^b \right\rangle = E_0 + (\varepsilon_a - \varepsilon_i)\, \delta_{ij}\delta_{ab} + (ai|bj) - (ab|ij)$$

- For singlet configurations

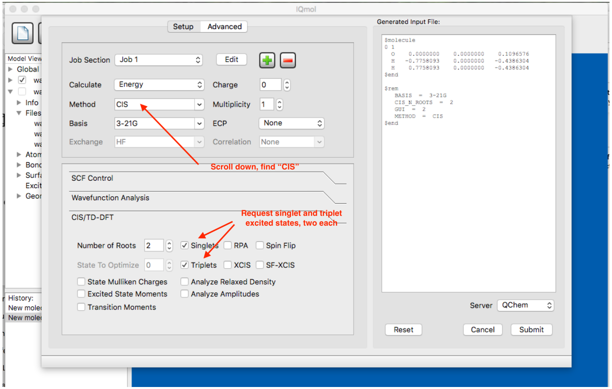
$$\left\langle \frac{\Phi_i^a + \Phi_{\bar{i}}^{\bar{a}}}{\sqrt{2}} \middle| \hat{H} \middle| \frac{\Phi_j^b + \Phi_{\bar{j}}^{\bar{b}}}{\sqrt{2}} \right\rangle = E_0 + (\varepsilon_a - \varepsilon_i)\, \delta_{ij}\delta_{ab} + 2(ai|bj) - (a2b|ij)$$

- For triplet configurations

$$\left\langle \frac{\Phi_i^a - \Phi_{\bar{i}}^{\bar{a}}}{\sqrt{2}} \middle| \hat{H} \middle| \frac{\Phi_j^b - \Phi_{\bar{j}}^{\bar{b}}}{\sqrt{2}} \right\rangle = E_0 + (\varepsilon_a - \varepsilon_i)\, \delta_{ij}\delta_{ab} - (ab|ij)$$

Case Study: Water Molecule

- Use IQmol to build water molecule (O-H: 0.95 Å; H-O-H: 105°).
- Perform CIS calcn’s to find two lowest singlet and triplet excited states.



Job Section: Job 1
Calculate: Energy
Charge: 0
Method: CIS
Multiplicity: 1
Basis: 3-21G
ECP: None
Exchange: HF
Correlation: None

Number of Roots: 2
State To Optimize: 0
[✓] Singlets
[ ] RPA
[ ] Spin Flip
[✓] Triplets
[ ] XCIS
[ ] SF-XCIS
[ ] State Mulliken Charges
[ ] Analyze Relaxed Density
[ ] Excited State Moments
[ ] Analyze Amplitudes
[ ] Transition Moments

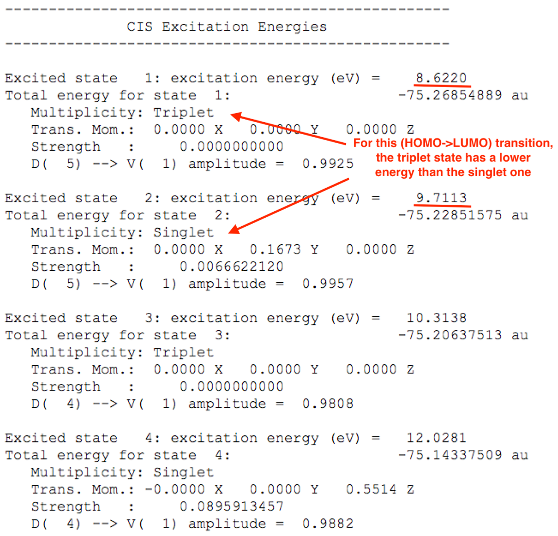
Server: QChem

Generated Input File:

```
$molecule
0 1
  O    0.0000000    0.0000000    0.1096576
  H   -0.7758093    0.0000000   -0.4386304
  H    0.7758093    0.0000000   -0.4386304
$end

$rem
   BASIS  =  3-21G
   CIS_N_ROOTS  =  2
   GUI  =  2
   METHOD  =  CIS
$end
```

Q-Chem Output File



```
 ---------------------------------------------------
            CIS Excitation Energies
 ---------------------------------------------------

 Excited state   1: excitation energy (eV) =    8.6220
 Total energy for state  1:                  -75.26854889 au
    Multiplicity: Triplet
    Trans. Mom.:  0.0000 X   0.0000 Y   0.0000 Z
    Strength   :     0.0000000000
    D(  5) --> V(  1) amplitude =  0.9925

 Excited state   2: excitation energy (eV) =    9.7113
 Total energy for state  2:                  -75.22851575 au
    Multiplicity: Singlet
    Trans. Mom.:  0.0000 X   0.1673 Y   0.0000 Z
    Strength   :     0.0066622120
    D(  5) --> V(  1) amplitude =  0.9957

 Excited state   3: excitation energy (eV) =   10.3138
 Total energy for state  3:                  -75.20637513 au
    Multiplicity: Triplet
    Trans. Mom.:  0.0000 X   0.0000 Y   0.0000 Z
    Strength   :     0.0000000000
    D(  4) --> V(  1) amplitude =  0.9808

 Excited state   4: excitation energy (eV) =   12.0281
 Total energy for state  4:                  -75.14337509 au
    Multiplicity: Singlet
    Trans. Mom.: -0.0000 X   0.0000 Y   0.5514 Z
    Strength   :     0.0895913457
    D(  4) --> V(  1) amplitude =  0.9882
```

For this (HOMO->LUMO) transition, the triplet state has a lower energy than the singlet one

```
from read_fcidump import *
from read_amplitudes import *
from matrix_print import *
import numpy as np
from numpy import linalg as LA

integrals = ElectronIntegrals()
integrals.read_from_fcidump("FCIDUMP")
NOrb = integrals.NOrb
NEle = integrals.NEle
NOcc = NEle/2
NVir = NOrb - NOcc
NOV = NVir * NOcc
print("NOrb=", NOrb, "NEle=", NEle, "NOcc=", NOcc, "NVir=", NVir, "NOV=", NOV)

EOrb = np.zeros(NOrb)    #EOrb[0] = EOrb[1] = ... = EOrb[12] = 0
for p in range(0, NOrb):
        EOrb[p] += integrals.one_e_integrals[p,p]
        for i in range(0, NOcc):
                EOrb[p] += 2.0*integrals.two_e_integrals[p,p,i,i]
                EOrb[p] -= integrals.two_e_integrals[p,i,p,i]
print("EOrb:", EOrb)

A = np.zeros((NOV, NOV))
for i in range(0, NOcc):
        for a in range(NOcc, NOrb):
                ai = (a-NOcc) + i*NVir
                A[ai, ai] = EOrb[a]- EOrb[i]
                for j in range(0, NOcc):
                        for b in range(NOcc, NOrb):
                                bj = (b-NOcc) + j*NVir
                                A[ai, bj] += 2.0 * integrals.two_e_integrals[a, i, b, j]
                                A[ai, bj] -= 1.0 * integrals.two_e_integrals[a, b, i, j]
matrix_print_2d(A, 6, "A")

Lambda, U = LA.eigh(A)
print("Lambda=", Lambda*27.211)
```