

Hartree-Fock Theory

Part I. Total and Orbital Energies

February 19, 2018

Schrödinger equation for a molecular system

- The Schrödinger equation is

$$\hat{H}\Psi(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_N) = E\Psi(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_N)$$

- The Hamiltonian is

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

which includes

- kinetic energy, $-\frac{1}{2} \nabla_i^2$, of each electron
- nuclear attraction $-\sum_A \frac{Z_A}{|\vec{r}_i - \vec{R}_A|}$
- electron-electron repulsion, $\sum_{i < j} \frac{1}{|\vec{r}_i - \vec{r}_j|}$

Hartree-Fock Energy for a Slater Determinant

- For N electrons in N **orthonormal** molecular orbitals, the Slater determinant is

$$\Psi(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_N) = \frac{1}{\sqrt{(N)!}} \begin{vmatrix} \psi_1(\vec{r}_1) & \psi_2(\vec{r}_1) & \cdots & \psi_N(\vec{r}_1) \\ \psi_1(\vec{r}_2) & \psi_2(\vec{r}_2) & \cdots & \psi_N(\vec{r}_2) \\ \cdots & \cdots & \cdots & \cdots \\ \psi_1(\vec{r}_N) & \psi_2(\vec{r}_N) & \cdots & \psi_N(\vec{r}_N) \end{vmatrix}$$

The energy of this Slater determinant is

$$E_{HF} = V_{NN} + \sum_{i=1}^N h_i + \sum_{i < j}^N [(ii|jj) - (ij|ij)]$$

where

$$\begin{aligned} h_i &= \int d\vec{r} \psi_i^*(\vec{r}) \left[-\frac{1}{2} \nabla^2 - \sum_A \frac{Z_A}{|\vec{r} - \vec{R}_A|} \right] \psi_i(\vec{r}) \\ (ii|jj) &= \iint d\vec{r}_1 d\vec{r}_2 \psi_i^*(\vec{r}_1) \psi_i^*(\vec{r}_1) \frac{1}{r_{12}} \psi_j(\vec{r}_2) \psi_j(\vec{r}_2) \\ (ij|ij) &= \iint d\vec{r}_1 d\vec{r}_2 \psi_i^*(\vec{r}_1) \psi_j^*(\vec{r}_1) \frac{1}{r_{12}} \psi_i(\vec{r}_2) \psi_j(\vec{r}_2) \end{aligned}$$

- Hartree-Fock energy (of a Slater determinant) is

$$\begin{aligned}
 E_{HF} &= V_{NN} + \sum_{i=1}^{N_e} h_i + \sum_{i < j}^{N_e} [(ii|jj) - (ij|ij)] \\
 &= V_{NN} + \sum_{i=1}^{N_e} h_i + \frac{1}{2} \sum_{i=1}^{N_e} \sum_{j=1}^{N_e} [(ii|jj) - (ij|ij)]
 \end{aligned}$$

- For closed-shell molecules

$$\begin{aligned}
 E_{HF} &= V_{NN} + 2 \sum_{i=1}^{N_e/2} h_i + \sum_{i < j}^{N_e/2} [2(ii|jj) - (ij|ij)] \\
 &= V_{NN} + \sum_{i=1}^{N_e/2} (h_i + \varepsilon_i)
 \end{aligned}$$

- We search for an optimal set of occupied molecule orbitals, $\psi_i(\vec{r})$, to minimize this energy. In practice, this is done through the **self-consistent field** (SCF) procedure.
- In DFT calculations, the $i = j$ term of Coulomb energy is no longer cancelled by its Hartree Fock exchange counterpart. This is called the **self-interaction error**.

- Notation for molecular orbitals
 - i, j, \dots , **occupied** molecular orbitals
 - a, b, \dots , **unoccupied (virtual)** molecular orbitals
 - p, q, \dots , occupied or unoccupied molecular orbitals
- **Orthonormality** condition

$$\langle \psi_p | \psi_q \rangle = \int \psi_p^*(\vec{r}) \psi_q(\vec{r}) d\vec{r} = \delta_{pq}$$

- Orbital energies

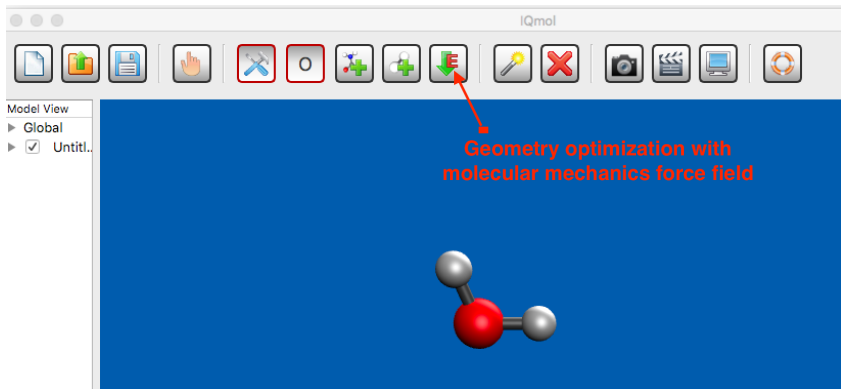
$$\varepsilon_p = h_{pp} + \sum_{i=1}^{N_e/2} [2(pp|ii) - (pi|pi)]$$

- **Electron density**

$$\rho(\vec{r}) = \sum_{i=1}^{N_{occ}} (\psi_i(\vec{r}))^2$$

Case Study: Water Molecule

- Use IQmol to draw a water molecule
- Use molecular mechanics to optimize its geometry



Case Study: Water Molecule

- Use IQmol to run Hartree-Fock/3-21G calculation

The screenshot displays the IQmol software interface. The 'Setup' tab is active, showing the following configuration:

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

Below these settings is the 'SCF Control' section:

- Algorithm:** DIIS
- Convergence:** 5
- Guess:** SAD
- Max Cycles:** 50
- Second Basis:** None
- Guess Mix:** 0%
- ☐ Dual Basis Energy
- ☐ Unrestricted

The 'Wavefunction Analysis' section is currently empty.

On the right, the 'Generated Input File:' window shows the input for the calculation:

```
!molecule
O 1
O -1.7238612 1.4159825 0.0032834
H -0.7738613 1.4157764 0.0031597
H -2.0407511 2.1838973 0.4641389
send

$read
BASIS = 3-21G
$end

$method = HF
$end
```

At the bottom right, the 'Server' is set to 'QChem'. The 'Job submission' buttons are 'Reset', 'Cancel', and 'Submit'.

Red arrows point from the text labels to the corresponding settings: 'Hartree-Fock' points to the Method dropdown, '3-21G' points to the Basis dropdown, and 'Job submission' points to the Submit button.

On the left side, a 'History' panel shows a list of actions: 'Add atom', 'Change atom t', 'Add hydrogen', and 'Minimize energy'. The bottom status bar indicates 'Chemical energy: 0.0000'.

Case Study: Water Molecule

- In IQmol, open the output file to draw a water molecule
- Find Hartree-Fock total and orbital energies

Model View

- Global
- h2o_321G
- Info
- Files
 - h2o_321G.out
 - h2o_321G.inp
 - h2o_321G.Fchk
- Atoms
- Bonds
- Surfaces
- Geometries

History:

- Add atoms/bonds
- Change atom type
- Add hydrogens
- Minimize energy

Computing

h2o_321G.out

```
6 -75.5854004632 3.74e-05
7 -75.5854005100 2.99e-06 Convergence criterion met
-----
SCF time: CPU 0.05s wall 0.00s
Total energy in the final basis set = -75.5854005100
Wall 1 = 0
Clock 1 = 0.0454859994
=====
Done GEN_SCFMAN
=====
-----
Orbital Energies (a.u.) and Symmetries
-----
Alpha MOs, Restricted
-- Occupied --
-20.422 -1.327 -0.698 -0.928 -0.477
1 A1 2 A1 1 B1 3 A1 1 B2
-- Virtual --
0.267 0.367 1.227 1.286 1.785 1.856 2.028 3.110
4 A1 2 B1 3 B1 5 A1 2 B2 6 A1 4 B1 7 A1

Beta MOs, Restricted
-- Occupied --
```


- Go to schooner, and make a new folder
- Copy
 - /home/yihan/qm_tutorial/qchem_files/FCIDump
 - /home/yihan/qm_tutorial/qchem_files/read_fcidump.pyto the new folder.
- Compute the Hatree-Fock total and orbital energies

```

import numpy as np
from read_fcidump import *

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

E1 = 0.0
for i in range(0, NOcc):
    E1 += 2*integrals.one_e_integrals[i,i]

print("E1=", E1)

E2 = 0.0
for j in range(0, NOcc):
    for i in range(0, NOcc):
        E2 += 2.0 * integrals.two_e_integrals[i,i,j,j]
        E2 -= 1.0 * integrals.two_e_integrals[i,j,i,j]

print("E2=", E2)

ETot = ENuc + E1 + E2
print("ETot=", ETot)

orbital_energies = np.zeros(NOrb)
for p in range(0, NOrb):
    orbital_energies[p] = integrals.one_e_integrals[p,p]
    for i in range(0, NOcc):
        orbital_energies[p] += 2.0 * integrals.two_e_integrals[p, p, i, i]
        orbital_energies[p] -= 1.0 * integrals.two_e_integrals[p, i, p, i]

print("orbital_energies=", orbital_energies)

```

~