جامعة الملك فهد للبترول والمعادن
King Fahd University of Petroleum & Minerals

Hartree–Fock Theory Implementation in Python

Introduction to Quantum–Chemical Calculations

Ibraheem F. Al-Yousef
Department of Physics, KFUPM

SCF Procedure

Mathematical Objects

Hartree–Fock–Roothan Equation:

$$\sum_{\nu} F_{\mu\nu} C_{\nu i} = \sum_{\nu} S_{\mu\nu} C_{\nu i} E_i$$
$$FC = SCE$$

Where:

F: Fock matrix, for closed shell systems F = H + 2J - K.

C: The orbital coefficient matrix.

S: Overlap matrix.

E: The orbital energy diagonal matrix.

H, J, K: Matrices representing one and two electron integrals.

SCF Procedure

1. Specify molecule, basis functions.
2. Form the overlab matrix **S** from the basis functions.
3. Provide an initial guess for the orbitals **C**.
4. Form the density matrix $\mathbf{D_{pq} = \sum_i^{occ} c_{pi}^* c_{qi}}$.
5. Form the Fock matrix $\mathbf{F = H + 2J - K}$
6. Solve the Hartree-Fock-Roothan equation: **FC** $=$ **SCE** to get a new orbital matrix **C**.
7. Calculate the Fock matrix **F** using the new orbital matrix **C**.
8. Repeat from step 4 until the orbital matrix **C** and the orbital energies converges.

Computing The Matrices:

The overlap matrix **S** contains the information about how the basis functions ϕ overlaps. Each element in the matrix is the Bra-ket of those basis functions. We shall form it using a matrix **B** that contains the basis vectors as its columns:

$$S_{ij} = \langle \phi_i | \phi_j \rangle$$

$$B = \begin{pmatrix} | & | & | \\ |\phi_1\rangle & |\phi_2\rangle & |\phi_3\rangle \\ | & | & | \end{pmatrix}$$

$$B^\dagger = \begin{pmatrix} - & \langle\phi_1| & - \\ - & \langle\phi_2| & - \\ - & \langle\phi_3| & - \end{pmatrix}$$

$$B^\dagger B = \begin{pmatrix} - & \langle\phi_1| & - \\ - & \langle\phi_2| & - \\ - & \langle\phi_3| & - \end{pmatrix} \begin{pmatrix} | & | & | \\ |\phi_1\rangle & |\phi_2\rangle & |\phi_3\rangle \\ | & | & | \end{pmatrix} \equiv S.$$

S to S’:

S Has to be orthonormal. This is not always the case, to ensure this orthonormality, we will construct an orthogonalization matrix A to orthogonalize S. Then, we will use A to transform S to be represented in the orthonormal basis S'

$$A = S^{-1/2}$$
$$B' = BA$$
$$S' = B'^{\dagger} B',$$
$$= (BA)^{\dagger}(BA),$$
$$= A^{\dagger} B^{\dagger} BA,$$
$$= A^{\dagger} SA.$$
$$S' = ASA$$

Transforming HFR Equation:

Now we will use **A** and some mathematical tricks to transform Hartree–Fock–Roothan (HFR) equation to the new orthonormal basis set:

$$FC = SCE$$

$$F(\mathbf{1})C = S(\mathbf{1})CE$$

$$FAA^{-1}C = SAA^{-1}CE$$

Mutliply by A from the left

$$AFAA^{-1}C = ASAA^{-1}CE$$

We can recognize $S' = ASA$, we can define $F' = AFA$ and $C' = A^{-1}C$

$$F'C' = S'C'E,$$

Since $S' = 1$ in the orthonormal basis set.

$$F'C' = C'E.$$

We will now start with an initial guess of F then we will get C by solving HFR Eq.

Density Matrix and I, J, K:

Recall $F = H + 2J - K$, in order to move forwars, we need J, K tp form F. We will define a density matrix D from the occupied orbitals in C, and a repulsive tensor I that will help us obtain these matrices.

$$D_{pq} = \sum_{i}^{occ} c^*_{pi} c_{qi}$$

Where $c^*_{pi} c_{qi}$ are the probability of some basis function p contributing to the MO i

$$I_{pqrs} = \int \mathrm{d}\tau \; \phi^*_p(1)\phi_q(1)\frac{1}{r_{ij}}\phi^*_r(2)\phi_s(2)$$

Using D and I we can now get J and K

$$J_{pq} = \sum_{rs} D_{rs} I_{pqrs}$$

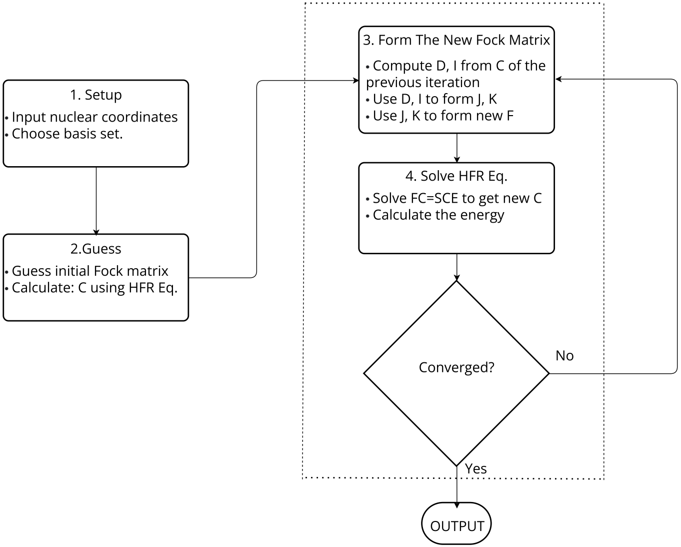
$$K_{ps} = \sum_{rq} D_{rq} I_{pqrs}$$

Calculating the energy and starting the iterative procedure:

Now we have everything we need, we will obtain the energy using this expression: $E = E_{nuc} + \sum_{pq}(H_{pq} + F_{pq})D_{pq}$, where E_{nuc} is the nuclear repulsion energy.

However, since $F_{Guessed} \rightarrow C \rightarrow D, J, K \rightarrow F_{New} \rightarrow C \rightarrow D...$, F (which depends on C) will yield a new C. This forms a Self-Consistent-Field SCF which we will solve iteratively until our orbital coefficients and our orbital energies converge.

Summary:


1. Setup
• Input nuclear coordinates
• Choose basis set.
2.Guess
• Guess initial Fock matrix
• Calculate: C using HFR Eq.
3. Form The New Fock Matrix
• Compute D, I from C of the previous iteration
• Use D, I to form J, K
• Use J, K to form new F
4. Solve HFR Eq.
• Solve FC=SCE to get new C
• Calculate the energy
Converged?
No
Yes
OUTPUT

Python Implementation

PSI4 initialization

```
[ ]: import psi4
     import numpy as np
     import datetime
     from scipy import linalg as splinalg
```

```
[ ]: psi4.set_memory('500 MB')
     numpy_memory = 2
     psi4.core.set_output_file('output.dat', False)
     basis = 'sto-3g'
     psi4.set_options({'basis': basis,
                       'scf_type': 'pk',
                       'e_convergence': 1e-8})
     mol = psi4.geometry("""
     O
     H 1 1.1
     H 1 1.1 2 104
     symmetry c1
     """)
     SCF_E_psi = psi4.energy('scf')
     psi4.core.clean()
     print(f"The Hartree-Fock ground state energy of the water is: {SCF_E_psi:.6f} Eh")
```

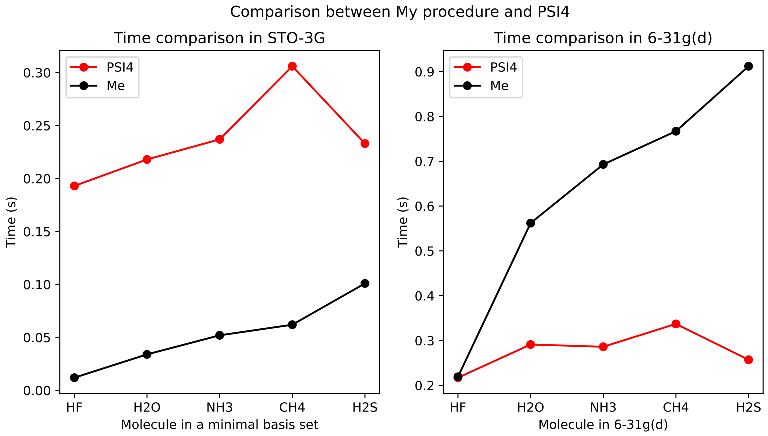
Pre-iterative Calculations

```
[ ]: wfn = psi4.core.Wavefunction.build(mol, psi4.core.get_global_option('basis'))
     ndocc = wfn.nalpha()    #Number of doubly occupied orbitals
     nbf = wfn.basisset().nbf()
     mints = psi4.core.MintsHelper(wfn.basisset())   #Molecular integrals object
     S_matrix = mints.ao_overlap()
     S = np.asarray(S_matrix)
     #Testing for orthonormality
     def isBasisOrthonormal(S):
         size_S = S.shape[0]
         identity_matrix = np.eye(size_S)
         orthonormal_check = np.allclose(S, identity_matrix)
         print(F'Q:(T/F) The AO basis is orthonormal? A: {orthonormal_check}')
         return orthonormal_check
     A = splinalg.sqrtm(np.linalg.inv(S))    #A=S^-1/2
     S_p = A.dot(S.dot(A))   #S'=ASA
     T = np.asarray(mints.ao_kinetic())
     V = np.asarray(mints.ao_potential())
     H = T + V
     F_p = A.dot(H.dot(A))   #Initial guess of F = Core hamiltonian
     vals, vecs = np.linalg.eigh(F_p)    #Solve HFR to get C
     C=A.dot(vecs)
     C_occ = C[:, :ndocc]
     D = np.einsum('pi,qi->pq',C_occ ,C_occ )    #Using einsum to compute D, J, K
     I = np.asarray(mints.ao_eri())
     J = np.einsum('rs,pqrs->pq',D,I )
     K = np.einsum('rq,pqrs->ps',D,I )
     F = H + 2*J - K
     isBasisOrthonormal(S_p)
```

SCF Procedure

```
[ ]: start_time = datetime.datetime.now()
     # ==> Nuclear Repulsion Energy <==
     E_nuc = mol.nuclear_repulsion_energy()
     # ==> SCF Iterations <==
     SCF_E = 0.0
     E_old = 0.0
     MAXITER = 40
     E_conv = 1.0e-8
     print('==> Starting SCF Iterations <==\n')
     # Begin Iterations
     for scf_iter in range(1, MAXITER + 1):
         I = np.asarray(mints.ao_eri())
         J = np.einsum('rs,pqrs->pq',D,I )
         K = np.einsum('rq,pqrs->ps',D,I )
         F = H + 2*J - K
         SCF_E = E_nuc + np.einsum('pq->',(H+F)*(D))
         print(F'SCF Iteration {scf_iter}: Energy = {SCF_E:.8f} dE = {SCF_E - E_old:.8f}')
         if (abs(SCF_E - E_old) < E_conv):
             break
         E_old = SCF_E
         F_p=A.dot(F.dot(A))
         vals, vecs = np.linalg.eigh(F_p)
         C=A.dot(vecs)
         C_occ = C[:, :ndocc]
         D = np.einsum('pi,qi->pq',C_occ ,C_occ )
         if (scf_iter == MAXITER):
             psi4.core.clean()
             raise Exception("Maximum number of SCF iterations exceeded.")
     # Post iterations
     print('\nSCF converged.')
     print(F'Final RHF Energy: {SCF_E:.6f} [Eh]')
     end_time = datetime.datetime.now()
     print("This took %f seconds" %(end_time-start_time).total_seconds())
```

Time Comparison



Thank you!

Questions?

References

Alenizan, A.
CHEM313 Computational Chemistry Lecture Notes, 2022.
https://github.com/alenaizan/CHEM_313/

McDonald, A.
Psi4Education Hartree-Fock Lab, 2020.
https://github.com/Psi4Education/psi4education/tree/master/labs/Hartree_Fock