Title:	Hartree-Fock Programming Project
System:	H_2O molecule, STO-3G basis
Language:	Matlab
Assignment:	CHEM 6485 Final Project
Submitted to:	David Sherrill (sherrill@gatech.edu)
Author:	Rebecca Han (rhan39@gatech.edu)
Date:	April 24, 2015

1. Hartree-Fock method

Developed in the first part of the 20th century, Hartree-Fock theory offers a way to approximately solve the Schrodinger equation for atoms or molecules (i.e. many-electron problems) by invoking three key assumptions and formulating an iterative algorithm for determining the system's wave function and energy.

The three major simplifications are as follows:

1. the Born-Oppenheimer approximation implies that the nuclei coordinates are fixed relative to electronic motions, and the total molecular wavefunction can be separated as products of the nuclear and electronic wavefunctions
2. the variational theorem is invoked, and a set of basis functions are defined to approximate the electron spin orbitals
3. the anti-symmetrized product of these orbitals is a single Slater determinant that approximates the energy eigenfunction

The last is also effectively equivalent to a mean-field approximation, where each electron only interacts with the average or mean potential field generated by all the other electrons. Therefore, the goal is to minimize the electronic energy by varying the orbitals – the resulting Slater determinant at the energy minimum is the closest approximation to the true wavefunction of the system.

Mathematically, it is convenient to define a Fock operator which includes the electronic Hamiltonian (composed of the one-electron kinetic energy and nuclear attraction potential terms), the Coulomb repulsion between an electron and the average electron density, and a two-electron exchange term that enforces antisymmetry.

At the system's energy minimum, the spin orbitals are eigenfunctions of the Fock operator, with corresponding orbital energy eigenvalues. Iteratively solving this eigenvalue equation for the orbitals that minimize energy is at the heart of Hartree-Fock routines such as the one implemented below.

2. Implementation in Matlab

One-electron and two-electron integrals for the H_2O molecule were generated in PSI4. The one-electron integrals (overlap, kinetic energy, potential energy) were treated as square matrices, with rows corresponding to atomic orbitals (AOs) and columns corresponding to the molecular orbitals (MOs); for water, the minimal basis set STO-3G requires seven basis functions. All 140 unique two-electron integrals were read into Matlab as a list and converted to a 4 index tensor, with the complete set of permutations specified for each unique integral, i.e.:

$$[pq|rs] = [qp|rs] = [pq|sr] = [qp|sr] = [rs|pq] = [sr|pq] = [rs|qp] = [sr|qp]$$

An initial guess density matrix was used to generate the Fock operator, assuming a Restricted Hartree-Fock (RHF) method, and calculate a total electronic energy. While the change in energy remained greater than convergence tolerance of 1e-6 (default PSI4 tolerance), the process would be repeated. The eigenvectors of the diagonalized Fock operator define a set of new orbitals which could be used to construct a new density matrix and new Fock operator.

Matlab was chosen as the programming language because it is well suited to linear algebra operations, with many built in subroutines for diagonalizing, transposing, and otherwise manipulating matrices, e.g.:

- **`[V,D] = eig(A)`** diagonalizes matrix A by returning the right eigenvectors in matrix V and corresponding eigenvalues in a diagonal matrix D, such that A*V = V*D
- **`[V,D] = eigs(A)`** also diagonalizes matrix A, but only returns the 6 largest eigenvalues and their corresponding right eigenvectors
- **`eigs(A,k)`** will return the k largest eigenvalues of matrix A as an array

2.1. Sorting the eigenvalues/eigenvectors

Since there were seven orbitals, eigs( ) was not suitable for returning the seven evals/evecs of a diagonalized matrix. Instead, the subroutine eig( ) was used, although it was not initially apparent that eig( ) returns the evals/evecs in random order. The order of the evals/evecs is ultimately irrelevant, but it is convenient to sort them in ascending order since the density matrix D is constructed by taking a double sum over the doubly occupied molecular orbitals (columns) of the SCF matrix. Negative eigenvalues correspond to the doubly occupied MO's – therefore, sorting the eigenvalues/eigenvectors allows for a straightforward double sum over the first five columns of the SCF matrix. This was very easily corrected by sorting the eigenvalues in ascending order and assigning a permutation index to each eigenvalue that was out of order. The eigenvectors were also rearranged by the same permutation.

2.2. Improving convergence time

One of Matlab's drawbacks is its relatively slower runtime compared to Python or FORTRAN. Implementing a direct inversion of the iterative subspace (DIIS) extrapolation technique would be one strategy to accelerate convergence on a solution. Alternatively, since the SCF iterations are rate-limiting and define the computation time, another strategy for more efficient code could be implementing composite indices for the two-electron integrals (naively written as four nested for-loops over p, q, r, and s).

However, for a molecule as small as H_2O, Matlab was able to converge on a solution in 13 iterations (see Table 3.1.), about 1.5 times as many iterations as required for PSI4, with two decimal points of precision compared to the PSI4 solution. Therefore, for this first pass program, neither DIIS nor fancier summation techniques appeared to be necessary. Perhaps those would be good additions to future programs designed to tackle larger systems.

3. Results and Discussion

Considering that the Matlab algorithm requires integrals generated by PSI4, it was affirming to see that given that input, Matlab returns the same converged energy as PSI4 up to two decimal points of precision. Both programs differ from the Dunning coverged value by a little more than one hartree. However, this could be explained by the choice of basis set – PSI4 used a STO-3G basis, which is a minimal basis that lacks any correlation or polarization. Dunning et al used both a (5s4p1d/3s1p) Slater basis and a [6s5p2d/3s1p] contracted Gaussian basis; the converged energy value is expected to be much more accurate.

PSI4 version 4.0b5 (http://sirius.chem.vt.edu/psi4manual/4.0b5/autodir_options_c/module__scf.html) uses DIIS extrapolation by default, so it is also not surprising that it has slightly faster convergence than the Matlab routine. As mentioned previously, however, the computational performance for both programs is similarly cheap, at least for the water molecule with minimal basis set.

Table 3.1. Comparison of converged energies for Matlab routine, PSI4, and literature

Source	Energy (*hartree*)	Iterations
Matlab routine	-74.9659	13
PSI4	-74.96479	9
Dunning±	-76.063	--

±T.H. Dunning, R.M. Pitzer, and S. Aung. (1972) *J Chem Phys* **57**, 12, 5044-5051.

In conclusion, the Matlab program provided here is a good example routine that demonstrates how easily and accurately the Hartree-Fock method can be implemented. Recasting the Fock operator and Hartree-Fock equations in terms of a linear algebra eigenvalue problem made the abstract physics problem more practical significance, especially in terms of gaining intuition for which steps/algorithms/computations are rate-limiting in the iterative calculation.

3.1. Example output for H2O

```
>> han_hartree_fock_noDIIS_0422
%%%% This is iteration number 0 %%

E =

 -70.986893715982490


dE =

   2.254021681730535

%%%% This is iteration number 1 %%

E =

 -75.461284828015494


dE =

   0.530234638082078

%%%% This is iteration number 2 %%

E =

 -74.907455303076290


dE =

   0.056989667331180

%%%% This is iteration number 3 %%

E =

 -74.973008343646200


dE =

   0.007273782534753

%%%% This is iteration number 4 %%

E =

 -74.965784938738352


dE =

     8.689556062790871e-05

%%%% This is iteration number 5 %%

E =

 -74.966218893223129


dE =

     3.234043795998787e-04

%%%% This is iteration number 6 %%
```

```
E =

 -74.966009318310142

dE =

     1.093431592948946e-04

%%%% This is iteration number 7 %%

E =

 -74.965953459102110

dE =

     5.261151730451275e-05

%%%% This is iteration number 8 %%

E =

 -74.965924052262864

dE =

     2.303382993318337e-05

%%%% This is iteration number 9 %%

E =

 -74.965911321579085

dE =

     1.026962087280481e-05

%%%% This is iteration number 10 %%

E =

 -74.965905611001858

dE =

     4.552461177809164e-06

%%%% This is iteration number 11 %%

E =

 -74.965903078989214

dE =

     2.019155914467774e-06

%%%% This is iteration number 12 %%

E =

 -74.965901955158557

dE =
     8.950713947797340e-07

%%%% Converged in 13 iterations %%
%%%% Optimized H-F energy: -7.496590e+01 %%
```