Project 1 report: Hartree Fock with Integrals

Benjamin Peyton

December 20, 2017

1 Overview

For this project, a code was developed according to the directions given in the Final Project 1 Description pdf. The code for the project spans across three files: `ints.py`, `hf.py`, and `opt.py`. The integral evaluation code is split into functions within `ints.py`, the Hartree Fock method is split into functions (and a total HF function is also implemented) within `hf.py` which calls on the integral functions from `ints.py` for the required integral matrices, and finally `opt.py` calls the HF procedure and uses it to perform a geometry optimization on H_2 using Newton-Raphson steps with approximated derivatives. The complete code can be found on my github page at `github.com/bgpeyton/HF_Int` along with a `README` which outlines the code and describes dependencies (`math`, `numpy`, and `scipy` packages).

2 Derivations and Procedure

2.1 Integral Evaluation

Before integral evaluation, a closed expression for the integral in Eq (1) was derived to be used later for Gaussian-type integrals (derivations separate).

$$R_k = \int_{-\infty}^{\infty} x^k e^{-\alpha x^2} dx \tag{1}$$

Next, using the Gaussian product theorem, a closed expression for the overlap integral between two primitive non-normalized Gaussians was derived. The function `gen_S()` was coded in `ints.py` to evaluate these overlap integrals, and another function `S_mat()` returns a matrix of such integrals given a molecule and basis set. A `normalize()` function was also implemented,

which returns the inverse square root of the self-overlap of a given basis function to be used as a normalization constant for S and all future matrices.

For kinetic energy integrals, again a closed form expression was derived. This expression was implemented in `gen_K()` within `ints.py` and then `K_mat()` generates a matrix of these integrals given a molecule and basis set.

An expression for nuclear attraction integrals was given, and this was implemented as `gen_V()` within `ints.py` and followed by `V_mat()`. Similarly, the expression for the electron repulsion integrals was given and implemented as `gen_eri()`, followed by `eri_mat()` which returns the later-discussed **G**-matrix necessary for the Hartree Fock procedure.

2.2 Hartree Fock Procedure

Evaluation of the Hartree Fock equations (Eq (2)) is made relatively simple using the functions defined in `ints.py`. The core Hamiltonian matrix **H** is directly solved from the **T** and **V** matrices solved using the corresponding generating functions fron `ints.py`. An orthogonalization matrix $\mathbf{S}^{-1/2}$ is generated and used to diagonalize the Hamiltonian, and the eigenvectors c are used to form the density matrix according to Eq (3).

$$\mathbf{Hc}_i^{(0)} = e_i \mathbf{Sc}_i^{(0)} \tag{2}$$

$$P_{ij} = \sum_{m=1}^{N/2} c_{mi} c_{mj} \tag{3}$$

The **G** matrix is then computed using the before-mentioned `eri_mat()` and the Fock matrix **F** is computed as the sum of **G** and **H**, to be used in the next HF step. An initial energy is computed (first using **H**, then using **F** in subsequent iterations) using Eq (4), and this process is repeated until convergence of the energy is reached. A single-HF-loop function `loop_hf()` was designed to take a molecule, basis, **H**, and **F** matrices. This function is then called in a loop within the `full_hf()` function, which computes the entire HF energy from start to finish with only a molecular coordinate array and basis set array as arguments (with optional convergence argument, the default being 10^{-12}).

$$E = \sum_{i=1}^{M} \sum_{j=1}^{M} (2H_{ih} + G_{ij}) P_{ij} \tag{4}$$

2.3 Geometry Optimization

A geometry optimization was implemented in `opt.py`. The first (E') and second (E'') derivatives of the energy are approximated using small $0.001au$ steps in the geometry (R), then used to take Newton-Raphson steps according to Eq (5). This is repeated until the absolute value of E'' was less than $10^{-6}au$.

$$R^{new} = R - \frac{E'(R)}{E''(R)} \tag{5}$$

3 Results

The Hartree Fock geometry optimization procedure produced values comparable (though not exactly equivalent to) a `Psi4` output using similar methods and starting values. The input and output are included as `input.dat` and `output.dat`. The bond distance, energies, derivatives, and second derivatives of each step are printed by `opt.py` and are included in Table (1) for an H_2 molecule starting with the atoms $1au$ apart and using the basis set provided in the project pdf.

Table 1: Energies and derivatives

R (au)	**E (au)**	**E' (au)**	**E" (au)**
1.0	-1.07263860249	-0.32892177304	1.53767876387
1.21390798961	-1.11524236275	-0.097629478193	0.735004021202
1.34673648493	-1.12257966082	-0.0185942960007	0.474442480858
1.38592837172	-1.12295919264	-0.00114918519656	0.416997325114
1.3886842292	-1.1229607803	-5.20953924443e-06	0.413221389106

`Psi4` predicts an equilibrium bond distance of 0.734828003532Å, or roughly $1.388623575au$, with an energy of $-1.122965336026au$. These are in good (though not perfect) agreement with the calculated values from `opt.py`, with discrepancies small enough to be attributed to slight differences in the basis set and procedure used in the calculation.

The NIST webbook[1] lists the equilibrium bond distance of H_2 as 0.74144Å, or roughly $1.40112au$. This discrepancy is not surprising- Hartree Fock ignores a lot of contributions, in-

cluding electron correlation, and this is hardly an ideal HF calculation considering the small uncontracted basis used. The literature[2] shows a bond dissociation energy (BDE) of $-1.12296au$, while this calculation predicts $BDE = 2(-0.5)au - (-1.12296au) = 0.12296au$, using the analytical solution of $0.5au$ per hydrogen atom. Again, this discrepancy is most likely due to shortcomings in the HF procedure.

References

[1] NIST, *NIST Webbook*, `http://webbook.nist.gov/cgi/cbook.cgi?ID=C1333740&Mask=1000`.

[2] G. Herzberg, *Phys. Rev. Lett.*, 1969, **23**, 1081–1083.