Physics 514 Final Project

Sam Cochran

December 7, 2022

1 Hartree Fock Implementation

1.1 Introducing the physics

See my presentation pdf, submitted along with this file and the source code.

1.2 Implementing the iterative procedure

The first thing I coded up was the iterative program to update the density matrix P. I then tested this code by inputting the values given in Szabo-Ostlund for the integrals at equilibrium distance and checked to ensure I was getting the correct Fock matrix and eigen-energies, again just by comparing with the given values in the text. This part was fairly straightforward. I just used a lot of nested loops to carry out all of the sums and double sums to implement the equations given in the text. The code to carry out this procedure is found in the SCF method in the hf_solver class in hartree_fock.py.

1.3 Calculating integrals

I then wrote another class to calculate all the necessary integrals for Hartree-Fock. The first appendix in Szabo-Ostlund gives all the simplified formulas for computing each integral type, and all of these end up having either a simple closed form expression or require only the evaluation of a single one-dimensional Gaussian (implemented using scipy's erf function). Each of the integrals was implemented using two functions: a primitive function that evaluates a single integral of a single product of Gaussians and the other terms in the integrand, and a function that calls the primitive function with the coefficients corresponding to the basis function multiple times. This second function evaluates the sum of integrals over primitive gaussians to get the integral for inputs that are contracted gaussian functions. The input for this class is also a set of basis coefficients, which were taken from basissetexchange.org.

I would note that the implementation of these integrals is not optimized at all, and redundantly calculates a lot of values multiple times. If speed were a concern, it would be straightforward to optimize a few things to make the integrations more efficient, but for this project the code runs more than fast enough, so I left it as is.

1.4 Computing the energy

Up to this point, the process was fairly straightforward coding and debugging using the equilibrium distance values from Szabo-Ostlund as a test set. In this step, I hit the first major snag that really prevented me from achieving my goal of testing things out using a different basis set (I had hoped to use cc-pvdz, but never got there).

The issue was that as I wrote the code, I tested it along the way, and at each milestone I was able to reproduce the results in the book. But when I went to run the final test, this time varying the internuclear distance, I got some very strange results.

For most of the values I tested, I did indeed get good results, but for some values of R, my algorithm inexplicably failed to converge, and even worse, I would get a nice smooth energy curve for one grid of R values, and then I would simply change the number of grid points and suddenly huge discontinuities began to appear.

I asked about this issue in office hours and then went to double check my integral code. I found that as I varied R, my integral calculations were still nice and continuous, so I concluded that the integral class was not the issue.

Next, I went through the SCF procedure and found that the values in my Fock matrix F were causing the discontinuities as R changed. After testing out each piece individually, starting from F and working backward through all of the computations that went into producing F, I found that the issue seemed to be coming from the calculation of C' (the variable C_p in the code), which is the matrix of eigenvectors of F', where $F' = X^T F X$, $X = U s^{\frac{1}{2}} U^T$, U is the matrix of eigenvalues of the overlap matrix S, and $s^{\frac{1}{2}}$ is the diagonal matrix whose diagonal entries are the eigenvalues of S raised to the $-1/2$ power.

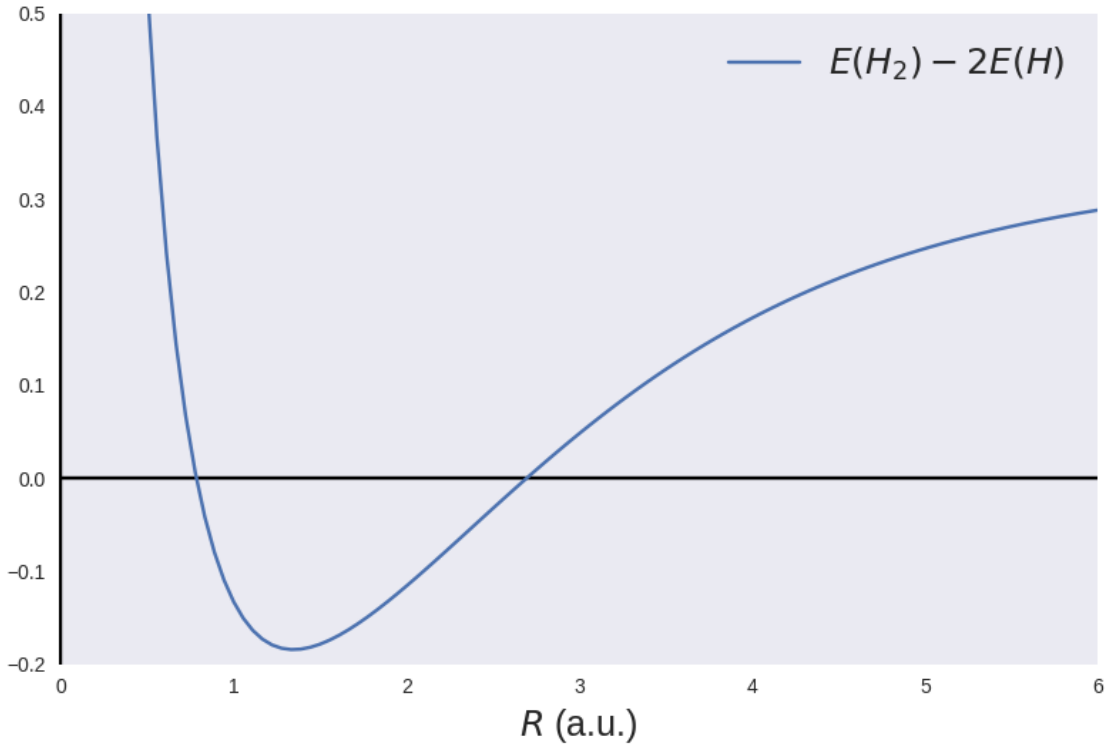
The issue here was a simple one, and a pretty dumb one. It turns out that numpy's eig function does not sort the eigenvalues for you. This is fine if you are just trying to diagonalize a matrix, because if you multiply on the left by the matrix of eigenvectors and on the right by the matrix of eigenvectors transposed, the order of the columns (the order of the eigenvectors) won't matter. But the ordering becomes an issue when computing C. When we compute C' as the matrix of eigenvectors of F', we then use this to define $C = XC'$. So since we only multiply by C' on the right, the order of the columns of C' very much matters.

Resolving the issue was simply a matter of sorting the eigenvalues of F', and then using the same sorting for the columns of C'. Once I made this very simple change, everything worked as expected, and I got nice smooth energy curves as I varied R. The results are plotted below using all the functions described above.

[1]:
```
from hartree_fock import *
from plotting import *

# STO-3G coefficients for hydrogen (from basissetexchange)
basis_coeffs = [(.4446345422, .1688554040), (.5353281423, .6239137298), (.
↪1543289673, 3.425250914)]
hf = hf_solver(basis_coeffs)
distances = []
energy_vals = []
for x in np.linspace(.5, 6, 100):
    R_A = np.array([0, 0, 0])
    R_B = np.array([x, 0, 0])
    distances.append(la.norm(R_A - R_B))
    hf.compute_integrals(R_A, R_B)
    hf.SCF(maxiter=100000)
    energy_vals.append(hf.compute_energy())
```

```
plot_energy(distances, energy_vals)
```



1.5 Interpreting results

As we can see in the plot above, there is a clear global minimum between 1 and 2 (it is at 1.346 a.u.). This corresponds to the bound state, meaning the equilibrium internuclear distance for the hydrogen molecule is estimated to be 1.346 a.u., which is very close to the actual value of 1.4 a.u., and so we see that our simulation works well for distances near the equilibrium value.

As expected, the energy blows up as R approaches zero, which makes sense when we think about the repulsion between the two nuclei. But we note that we've actually plotted, as a function of internuclear distance R, the difference in energy between an H_2 molecule with internuclear distance R and two standalone hydrogen atoms. So the bound state is negative, corresponding to a hydrogen molecule having lower ground state energy than two separate hydrogen atoms, as the energy goes into the bond. But there is an issue here as we let R become big. As R increases, we would expect that the two hydrogen atoms in the molecule would dissociate and become standalone hydrogen atoms, i.e. as the internuclear distance grows, the energy of the molecule should approach twice the energy of a single hydrogen, and thus the curve plotted above should go to zero. It does not, and in fact it seems to grow more and more positive as R increases.

This is a shortcoming of our model. The way it is set up, we are essentially forcing atoms to occupy orbitals in pairs, and so we are unable to properly model behavior for atoms that do not have closed shells. Specifically, in this case of hydrogen, each hydrogen atom has only one electron, i.e. one valence electron, and so as R increases, in order to properly show the dissociation, a model

would have to be able to describe atoms with single electrons in the outermost shell. So our model produces inaccurate results in this regime of R large. Nevertheless, we still get good results near the equilibrium value, so this shortcoming of our model does not detract from its usefulness in modeling the bound state, which is what we set out to do in the first place.

1.6 Possible expansions

One way that I had hoped to go further with this project was to use different basis functions (cc-pvdz was discussed). Though I didn't have time this semester, the would be one way to improve the model, since STO-3G as a minimal basis isn't likely to be used in practice, and is more a useful tool conceptually to explain the idea of Hartree-Fock.

Other generalizations of Hartree-Fock that I read about deal with electron correlation. As we've done it here, Hartree-Fock is a mean-field theory, meaning that for each electron we measure the average repulsion of all the other electrons (for hydrogen there are actually only two). Post-Hartree-Fock methods find more accurate ways to deal with electron-electron interactions.

Code available at github.com/samco7/hartree-fock