FYS4411 Project 1

Wilhelm Holmen

April 9, 2015

Monte-Carlo integration for Helium atom

In this exercise I will use Monte Carlo integration to compute the expectation value for the energy.

$$\langle E \rangle = \frac{\int d\mathbf{r_1} d\mathbf{r_2} \psi_T^*(r_1, r_2) \hat{H} \psi_T(r_1, r_2)}{\int d\mathbf{r_1} d\mathbf{r_2} \psi_T^*(r_1, r_2) \psi_T(r_1, r_2)}$$

I need a wavefunction for this, so first I choose the ground state wavefunction solution the hydrogen atom, $e^{-\alpha r}$, for both the particles. This gives the total wavefunction

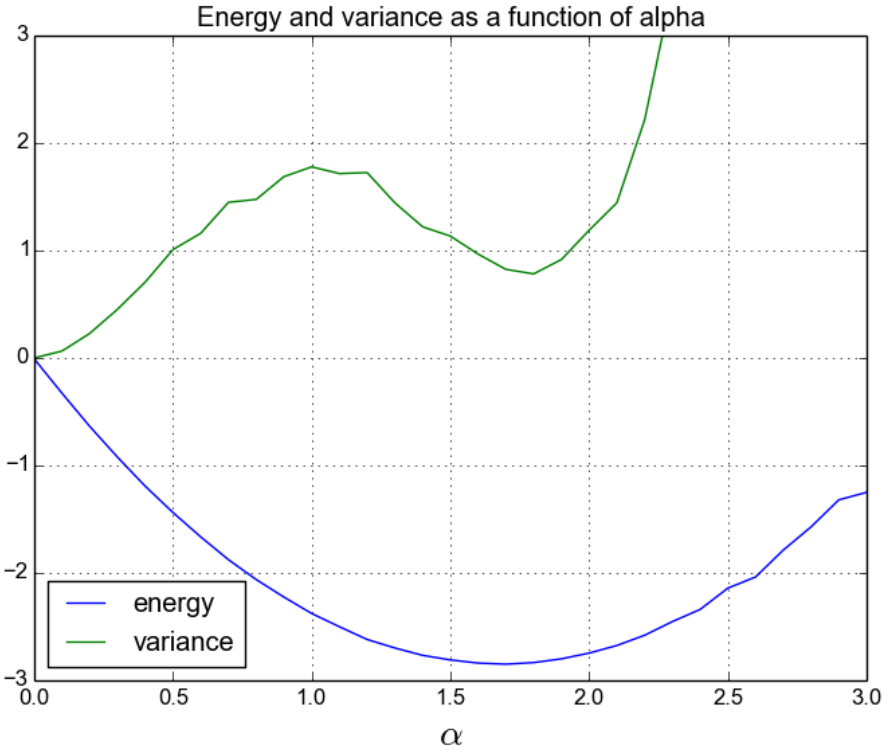
$$\Psi_T(r_1, r_2) = e^{-\alpha(r_1 + r_2)}$$

This is not the correct wavefunction, but it can give a decent first evaluation.

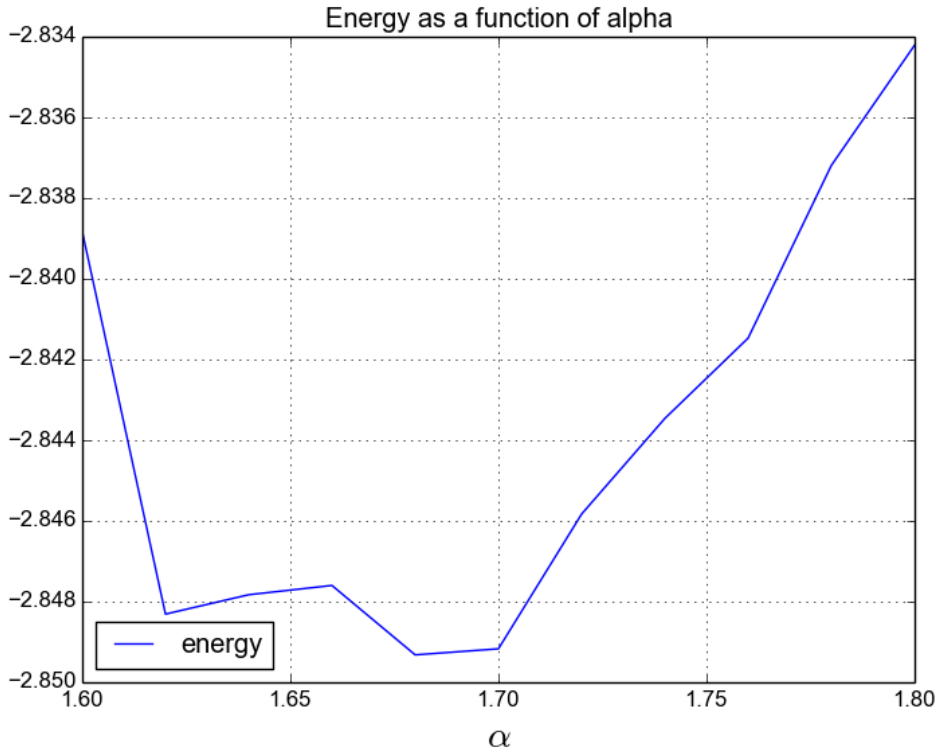
In the Metropolis algorithm, I will check wether a move given by, $\mathbf{R}' = \mathbf{R} + \delta \cdot r$, will be accepted. r is a random number in $[0, 1]$. The acceptance criteria is given by

$$\frac{P(\mathbf{R}')}{P(\mathbf{R})} = \frac{\int d\mathbf{R}' \psi_T^*(\mathbf{R}') \psi_T(\mathbf{R}')}{\int d\mathbf{R} \psi_T^*(\mathbf{R}) \psi_T(\mathbf{R})} \geq r$$

Because of the variational principle, the trial energy will always be higher than the true energy. Therefore we can adjust α and the steplength δ until we get the lowest possible energy. The best steplength δ is a steplength that will accept around 50% of the proposed steps. Using an algorithm that loops over different steplengths, I adjust α and plot the energy and variance as a function of α. I start the steplength at $\delta = 1.2$ and increase it by $0.5 * r$, $r \in [0, 1]$ for each step. Testing if the number of accepted steps are within $[0.49, 0.51]$ I either accept the step length or increase it.



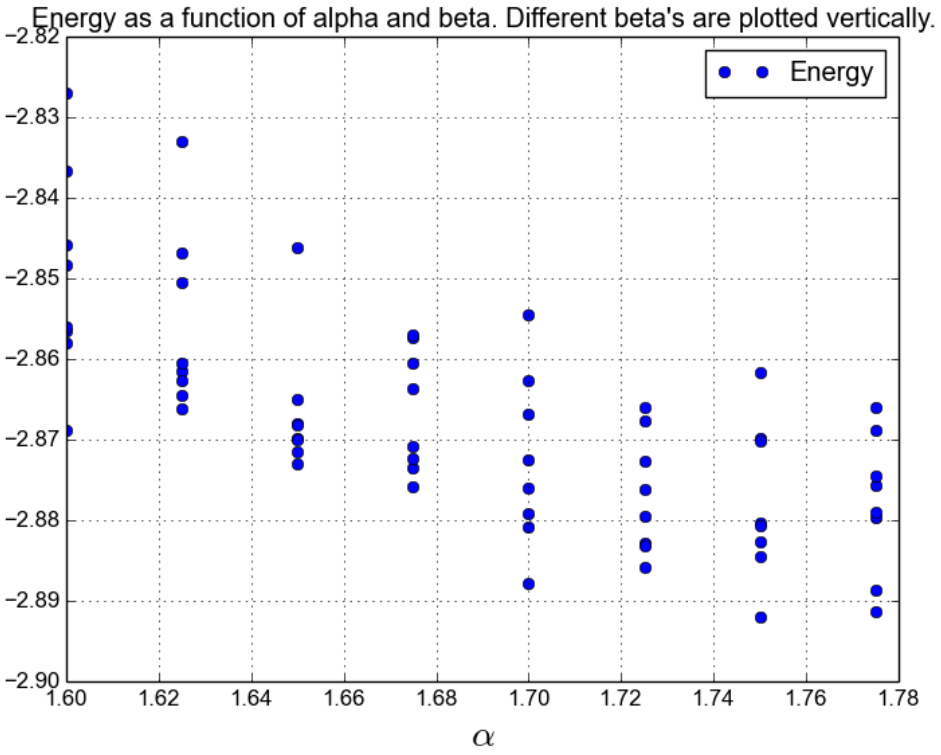
By looking closer at the energy and variance around $\alpha = 1.68$, we see that we get an energy of $\langle E \rangle = -2.84997$, with a variance of $\sigma^2 = 0.932912$. This is pretty close to the real value of $E_{exact} = -2.903$. We see however that the variance is smaller for $\alpha = 1.8$, 1.9 and 1.6.



We can get a better result by improving our test function, Ψ_T. We can introduce a Jastrow factor. Here $r_{12} = |r_1 - r_2|$.

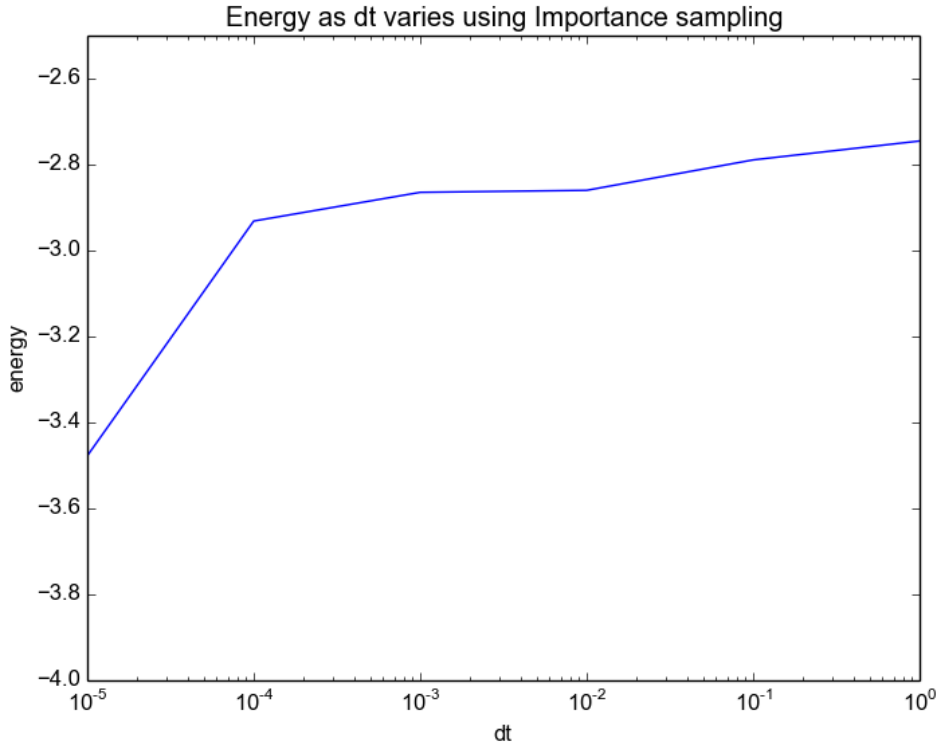
$$\Psi_T(r_1, r_2) = e^{-\alpha(r_1+r_2)} e^{\frac{r_{12}}{2(1+\beta r_{12})}}$$

By varying α and β to find the lowest energy, we see that we can get a lower energy. Namely $E = -2.89207$ for $\alpha = 1.75$ and $\beta = 0.3$. I will later use these values when I do an optimal Monte Carlo integration with statistical analysis.



Importance Sampling

Introducing importance sampling. We can see from the plot below that using $dt = 10^-3$, we get stable results. A smaller dt probably gives rise to different kinds of errors, like truncation errors etc. Using this dt, we get -2.86462 as the energy. This is a slightly higher energy than the energy obtained in



Statistical Analysis

A Monte Carlo simulation gives rise to two kinds of error. Systematic errors and statistical errors. Systematic errors are due to limitations of the applied models and faulty implementations. Here I will explore methods to estimate the statistical error.

Given a set of local energies, the variance is a measure of the spread from the true mean. The definition is

$$\mathrm{Var}(E) = \left\langle E^2 \right\rangle - \left\langle E \right\rangle^2$$

Unfortunatly we do not know the true mean in a Monte-Carlo simulation. The computed average $\overline{E}$ is an approximation to the exact mean, and we do the following approximation

$$\mathrm{Var}(E) \approx \overline{E^2} - \overline{E}^2$$

For the case where we have the exact wave function, the variance becomes 0, so the variance is an excellent measure of how close we are to the exact wave function. The variance however is *not* a direct measure of the error. The standard deviation, the square root of the variance is related of the *spread* in the sampled value.

$$\sigma^2(x) = \mathrm{Var}(x) \tag{1}$$

This does not account for the samples being corrolated. Two samples, x and y are corrolated if the *Covariance* is non-zero.

$$\mathrm{Cov}(x, y) = \langle xy \rangle - \langle x \rangle \langle y \rangle$$

The diagonal elements of the Covariance is the Variance. By ignoring the corrolations, one get an error estimate that is generally too small. Given the true deviation, σ_e and σ from (1) we have

$$\sigma_c \geq \sigma$$

The most interesting quantity when doing statistical analysis is not the error of single samples. It is the error of the mean value. One can show that the variance for the mean value, m is

$$\sigma^2(m) = \frac{1}{n}\mathrm{Cov}(x) \tag{2}$$

Where n is the number of samples used to calculate m and

$$m = \frac{1}{n}\sum_i^n x_i$$

Calculating the Covariance is an expensive process for a large sample set, so we need a better way to calculate this.

Blocking

There is no need to do statistical analysis within the Monte-Carlo simulation. By storing the data set, one can estimate the error post process. An efficient algorithm for this is called blocking.

Given a set of N samples from a single Monte-Carlo process. This set is divided into n blocks of size $n_b = N/n$. Now we can treat each block as an individual simulation to calculate the variance of a calculated mean, m. This will in turn give us the covariance by (2).

$$\sigma^2(m) = \langle m^2 \rangle - \langle m \rangle^2$$

One can not know beforehand which block size is optimal to compute the exact error. One can, however, plot the variance against different block sizes. The curve should be stable over a span of block sizes. This plateau will serve as a reasonable approximation to the covariance.

One-body density

The one-body density is defined as

$$\rho(r_1) = \int_{r_2} .. \int_{r_N} |\Phi(r_1 r_2 .. r_N)|^2 dr_2 .. dr_N$$

The distribution $|\Phi(r)|^2$ describes the distribution of a particle in the system. The one-body density $\rho(r_1)$ describes the simultaneous distribution of every particle in the system. $\rho(r_1)dr_1$ represents the probability of finding *any* particle in the volume element dr_1 as oppsed to $|\Phi(r)|^2$ giving the probability of finding *particle* r_1 in the volume element dr_1. Due to the indistinguishable nature of the particles, any of the coordinates contain information about all the particles. I will therefore normalize this density to the number of particles N and not 1 which is common for the one-particle density.

The way this is computed in a Monte Carlo simulation is by taking snapshots of all positions for every timestep, then constructing a histogram to construct an approximation to the wave function.

The charge density for a single particle is given by

$$\rho_q(r) = q|\Psi(r)|^2$$

However, since we already have computed the many-body probability density, we can use this instead to get the many-body charge density of the system.

$$\rho_q(r) = Q\rho(r_1)$$

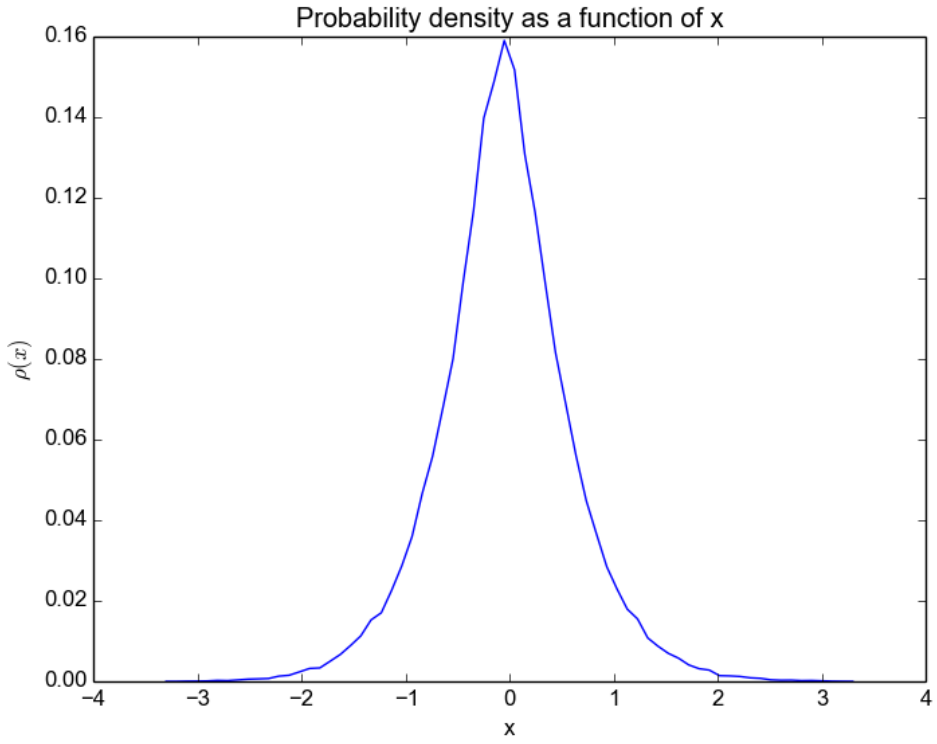
Final variational Monte-Carlo on Helium

I have found the optimal parametres for calculating on Helium. I set $\alpha = 1.75$, $\beta = 0.3$, step $= 1.5$. This computation gave an energy of

$$E = -2.89207$$

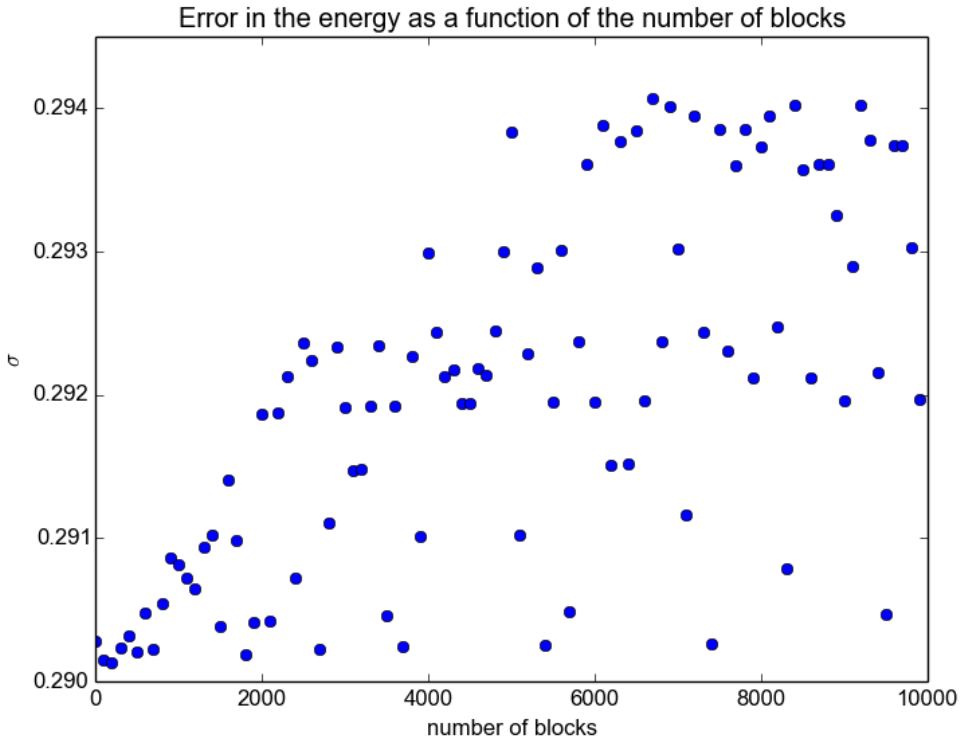
Which is pretty close to the best value -2.903.

A plot of the probability density:



Since we only have the 1s-states, the density should be close in shape to the 1s-shape. This density will be more spread out when one take the electron-electron repulsion into account, because the repulsion can be seen as a reduction in the core's charge.

By using the blocking analysis, the following plot shows how variance of the mean E-value changes when the block size changes. The y-axis shows the total error in the calculation. One see that the error is estimated to be around 0.29. The spread seams to be pretty uniform and I conclude that the corrolation is low.



Note on programs

I deliver this project as a project on Helium without analytical calculations. I will finish up the next project as fast as possible. And there I will focus on the analytical local energy to compute both Helium, Beryllium and Neon.

The python program blocking_analysis.py computes the blocking graph and the probability density.

Helium_plots.py generates the plots for different α.

The c++ code will have a pretty different structure in project 2.