FYS4411 - Project 2
The restricted Boltzmann machine applied to the quantum many-body problem

Kristoffer Langstad [*]
krilangs@uio.no

Abstract

In this project we study fermions in a harmonic oscillator potential using a restricted Boltzmann machine (RBM) to approximate the wave function as a neural network quantum state (NQS) combined with a variational Monte Carlo (VMC) method. To compute the ground state of the system, a stochastic gradient descent (SGD) method is used, and to do a statistical analysis of the results we use blocking. The computation of the system is tested with three sampling methods; a standard brute-force Metropolis method, Importance sampling and Gibbs sampling.

The computed energies in atomic units (a.u.) are benchmarked against analytical ground state energies for systems of one and two electrons in one and two dimensions with and without interaction. The computed results are found to have a strong dependence on the initial guesses of the variational/RBM parameters of the system, together with the sampling and learning rate parameters. For both the non-interacting and interacting cases the Gibbs method is found to give the best solutions and on average the best errors with the found optimal parameters, but all three sampling methods produce good energies close to the expected analytical energies.

1 Introduction

Machine learning is a more and more used method in physics to simulate and study quantum systems. One of the categorizations in machine learning where we consider the desired output of a system, is Neural Networks (NN) and Boltzmann machines. Boltzmann machines are a type of unsupervised learning method. One application is to represent a wave function with a restricted Boltzmann machine (RBM). Such a wave function/network is named a neural network quantum state (NQS) after Carleo and Troyer [2], which used this on quantum mechanical spin lattice systems of the Ising model and Heisenberg mode.

In this project we will test the applicability of the RBM's to quantum mechanics on a system with electrons confined to move in a harmonic oscillator trap. The system will be studied in one and two dimensions for one and two particles with different number of hidden nodes for non-interacting particles. Here, we can derive the ground state energy

[*]https://github.com/krilangs/FYS4411/tree/master/Project2 [1]

analytically. For the interacting case we only study two electrons in a quantum dot in two dimensions with various number of hidden nodes. In this case, we will compare our computed energies with analytical ground state energies from Taut [5].

To do the ground state energy calculation, we will use Variational Monte Carlo (VMC) methods with three different sampling methods: a standard "brute-force" Metropolis sampling, an improved Metropolis-Hastings/Importance sampling and Gibbs sampling. We will use a Gaussian-Binary RBM as the neural networks (NN) with visible and hidden nodes to represent the wave function. The weights of these nodes are what we will use as variational parameters to approximate the ground state. We also use a Stochastic Gradient Descent (SGD) with a fixed learning rate to find the parameters that minimize the energy function. Then we do a statistical error analysis on the ground state energies by using the blocking method.

For the layout of this project, we first take a look at the theory of the physics and the numerics we will use in this project. Here get an overview of the physical system, neural networks, restricted Boltzmann machine, the analytical derivations of the ground state energies and then the theory of the numerical methods we are using. In the methods section, we explain what we do with the implementation of the numerical tools we use and the parameters we use for running the code. Here we will use the three sampling methods we have mentioned to sample the ground state energies for varying parameters and various cases. The computed energies, with a statistical analysis, will be benchmarked against the analytically derived ground state energies to see how well our sampling methods can compare with the analytical results. In the results section, we present and discuss the results we get from the VMC computation. Lastly, we come up with a conclusion to the project with possible aspects towards future work.

2 Theory

2.1 The System

We will consider a system of $N_p = 2$ electrons as function of the oscillator frequency ω, confined in a pure 2D isotropic harmonic oscillator. With two electrons we can exclude Pauli's exclusion principle, which states that no more than two fermions can occupy the same quantum state simultaneously, and the Slater determinant. This system is considered as a quantum dot with frequency $\hbar\omega = 1$. For this system we have exact closed form expressions for the ground state energy from Taut [5] for selected values for ω to use as benchmarks. The total Hamiltonian, with natural units ($\hbar = c = e = m_e = 1$), of this system is given as

$$\hat{H} = \hat{H}_0 + \hat{H}_1 = \sum_{i=1}^{N_p} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right) + \sum_{i<j} \frac{1}{r_{ij}}. \tag{1}$$

Here $\hat{H}_0$ is the standard harmonic oscillator part, which is the Hamiltonian we will use without interaction between the electrons, and $\hat{H}_1$ is the repulsive interaction given by the Coulomb potential. The energies are in atomic units (a.u.), while the modulus of the position for an electron i is $r_i = \sqrt{r_{i_x}^2 + r_{i_y}^2}$ and the distance between the electrons are given as $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$.

2.2 Neural Networks (NN)

The main the differences between this project and the first is that we now look at fermions and that we want to use machine learning and neural networks to represent the wave function. We are still using a variation of the variational Monte Carlo (VMC) calculation to do the simulation.

Neural networks are computational models that are supposed to mimic biological systems, and they consist of layers of nodes that are connected. Neural networks can solve a various of problems like supervised, unsupervised and reinforcement learning. Neural networks have commonly, as seen in Figure 1, an input/visible layer, a hidden layer and an output layer. This is a simple case with two input variables/nodes, one hidden layer with four nodes and one node in the output layer. This is called a feed-forward neural network (FFNN), since the information only moves in one direction. Neural networks can contain many hidden layers, and the number of nodes in each layer have to be decided by us and may vary from layer to layer. The input and output layers contain as many nodes as there are input and output variables respectively. The number of output variables vary depending on the problem and may differ from the number of input variables. From the figure we see that the input variables are sent to the hidden layer nodes where they are processed with an activation function before sent to the output layer. The connection between the nodes in the different layers are affected by weight variables **W**, which gives weighted sums to be passed through an activation function. The outputted weighted sum have to pass a certain threshold to not give zero output. The visible and hidden nodes are usually affected by biases respectively as well. In neural networks a training process is normally used to find the optimal biases and weights that minimize a chosen cost function. Neural Networks have many other aspects which we will not cover in this project, since they are not used.

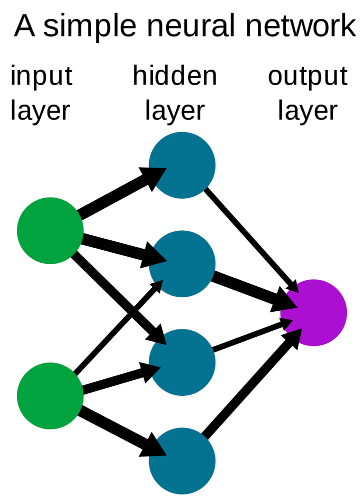

Figure 1: Schematic diagram of a simple single hidden layer, feed-forward neural network with two input nodes, four nodes in the single hidden layer and one output node.

2.3 Restricted Boltzmann Machine (RBM)

We will use a restricted Boltzmann machine as our neural network of choice for this project. This type of NN consists of one layer with input/visible nodes and one layer consisting of hidden nodes (similar to Figure 1). The visible and hidden nodes are also affected by separate biases and weights. There is only a connection between a visible and a hidden node, meaning that there are no connections between nodes in the same layer. With the RBM (generated) network, we want to learn or produce a probability distribution of the simulated system which then is used to generate an output. In this case the wave function Ψ is the probability distribution, and the generated output is the positions of the electrons. Here the cost function is given by the gradient of the expectation value of the energy of the wave function with respect to the parameters $\alpha = [\mathbf{a}, \mathbf{b}, \mathbf{W}]$. This minimization will lead to the ground state of the system. This gradient will be used in the stochastic gradient descent method, which is explained more later (sect. 3.3).

The RBM network is classified as reinforcement learning under unsupervised learning in neural networks. This is because we are not using training data, but use the quantum variational principle where the neural network quantum state (NQS) wave function is used to represent the ground state after the quantum mechanical energy is minimized with respect to the parameters **a**, **b** and **W**. This is then used to optimize the weights and biases of the network.

For the RBM we only have the joint probability distribution between the visible and hidden nodes:

$$F_{rbm}(\mathbf{X}, \mathbf{H}) = \frac{1}{Z} e^{-\frac{1}{T_0} E(\mathbf{X}, \mathbf{H})} \tag{2}$$

X is the visible nodes, which in this case is the output of the system (particle positions), **H** is the hidden nodes and Z is the partition function and is a normalization constant:

$$Z = \int\int \frac{1}{Z} e^{-\frac{1}{T_0} E(\mathbf{x}, \mathbf{h})} d\mathbf{x} d\mathbf{h} \tag{3}$$

For simplicity, we will just ignore T_0 by setting it to one. E in equation 2 and 3, is an energy function describing the relations between the visible and hidden nodes called the energy of the node configuration. Not to be confused with the local energy of the system. The choice of energy function will determine what kind of RBM version we are using.

For a case where both the visible and hidden nodes only takes on binary values, the most common version of the RMB is used, which is called "Binary-Binary". For our case with fermions, we want continuous values for the particle positions (visible nodes). So we will use an energy function for the RBM called the "Gaussian-Binary":

$$E(\mathbf{X}, \mathbf{H}) = \sum_{i}^{M} \frac{(X_i - a_i)^2}{2\sigma_i^2} - \sum_{j}^{N} b_j H_j - \sum_{i,j}^{M,N} \frac{X_i w_{ij} H_j}{\sigma_i^2} \tag{4}$$

If $\sigma_i = \sigma$:

$$E(\mathbf{X}, \mathbf{H}) = \frac{||\mathbf{X} - \mathbf{a}||^2}{2\sigma^2} - \mathbf{b}^T \mathbf{H} - \frac{\mathbf{X}^T \mathbf{W} \mathbf{H}}{\sigma^2} \tag{5}$$

X is a vector containing the visible nodes with length M, **a** is a vector containing the visible biases with length M, **H** is a vector containing the hidden nodes with length N, **b** is a vector containing the hidden biases with length N and $\mathbf{W}$ is a weight matrix containing the weights characterizing the connection of each visible node to a hidden node of size $M \times N$. We have the number of visible nodes $M = P \cdot D$, where P is the number of particles and D is the number of dimensions. For the number of hidden nodes N, we will choose and experiment with. The RBM parameters $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{W}$ will act as variational parameters like in the standard VMC calculation. The total number of parameters will then be $M + N + M \cdot N$, from the dimensions of the RBM parameters.

To represent the wave function we will use the marginal probability distribution function (PDF) by summing over the hidden nodes, changing $F_{rbm}(\mathbf{X}, \mathbf{H})$ in equation 2 to

$$\Psi(\mathbf{X}) = F_{rbm}(\mathbf{X}) = \sum_{\mathbf{h}} F_{rbm}(\mathbf{X}, \mathbf{h}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{X},\mathbf{h})}. \tag{6}$$

Using the Gaussian-Binary RBM, the wave function becomes:

$$\begin{aligned} \Psi(\mathbf{X}) &= \frac{1}{Z} \sum_{\{h_j\}} e^{-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_j^N b_j h_j + \sum_{i,j}^{M,N} \frac{X_i w_{ij} h_j}{\sigma^2}} \\ &= e^{-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2}} \prod_j^N \left(1 + e^{b_j + \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}\right) \end{aligned} \tag{7}$$

The partition function will not affect the result in any way, which is why we will ignore the partition function by setting $Z = 1$ as we did with T_0.

2.4 Local Energy and Analytical Expressions

2.4.1 Evaluation of Local Energy

To find the ground state energy of the fermions, we have from the variational principle that the exact energy have to be smaller or equal to the expectation value of the local energy as a function of the variational parameters $\alpha = [\mathbf{a}, \mathbf{b}, \mathbf{W}]$ given a trial wave function (eq. 7), $E_{exact} \leq \langle E_L \rangle$. The expectation value of the energy with a trial wave function Ψ_T and the Hamiltonian is computed as

$$\langle E \rangle = \langle H \rangle = \frac{\int \Psi_T^*(\mathbf{X}, \alpha) \hat{H} \Psi_T(\mathbf{X}, \alpha) d\mathbf{X}}{\int \Psi_T^*(\mathbf{X}, \alpha) \Psi_T(\mathbf{X}, \alpha) d\mathbf{X}}, \tag{8}$$

where the local energy is defined as

$$E_L(\mathbf{X}, \alpha) = \frac{1}{\Psi_T(\mathbf{X}, \alpha)} \hat{H} \Psi_T(\mathbf{X}, \alpha). \tag{9}$$

We define the probability density function (PDF) as

$$P(\mathbf{X}, \alpha) = \frac{|\Psi_T(\mathbf{X}, \alpha)|^2}{\int |\Psi_T(\mathbf{X}, \alpha)|^2 d\mathbf{X}}. \tag{10}$$

Since we will study large numbers in the Monte Carlo simulations, we apply Bernoullis law and use the PDF (eq. 10) on the expectation value (eq. 8) for the local energy, which gives

$$E_{exact} \leq \langle E_L \rangle = \int P(\mathbf{X}, \alpha) E_L(\mathbf{X}, \alpha) d\mathbf{X} \approx \frac{1}{MC} \sum_{i=1}^{MC} E_L(X_i, \alpha), \tag{11}$$

where MC is the number of Monte Carlo cycles.

The full expression for the local energy with the expression for the Hamiltonian (eq. 1) and the NQS wave function (eq. 7) is

$$E_L = \frac{1}{\Psi(\mathbf{X})} \hat{H} \Psi(\mathbf{X}) = \frac{1}{\Psi(\mathbf{X})} \left(\sum_{i=1}^{N_p} \left[-\frac{1}{2} \nabla_i^2 + \frac{1}{2} \omega^2 r_i^2 \right] + \sum_{i<j} \frac{1}{r_{ij}} \right) \Psi(\mathbf{X}) \tag{12}$$

By looking at this expression we can also represent the local energy in terms of kinetic, potential and interaction energy:

$$E_L = \sum_{k=1}^{M} (E_k^{kin} + E_k^{pot}) + \sum_{i<j} E_{ij}^{int} \tag{13}$$

The potential and interaction energies are pretty straight forward to implement numerically. For the kinetic energy we have to compute $\nabla^2 \Psi$ analytically first. The derivation of the kinetic energy is done in Appendix A. The derivation in this appendix gives the full local energy expression with the potential and interaction energies included last:

$$\begin{aligned} E_L = &-\frac{M}{\sigma^2} + \sum_i^M \sum_j^N \Sigma(u_j) \cdot \Sigma(-u_j) \cdot \frac{w_{ij}^2}{\sigma^4} + \sum_i^M \frac{X_i - a_i}{\sigma^2} \\ &- 2 \sum_i^M \frac{X_i - a_i}{\sigma^2} \sum_j^N \Sigma(u_j) \cdot \frac{w_{ij}}{\sigma^2} + \sum_i^M \sum_j^N (\Sigma(u_j))^2 \cdot \frac{w_{ij}^2}{\sigma^4} \\ &+ \frac{1}{2} \sum_{k=1}^{M} \omega^2 X_k^2 + \sum_{i<j} \frac{1}{X_{ij}} \end{aligned} \tag{14}$$

This energy derivation works for the brute force Metropolis and Importance sampling.

2.4.2 Local Energy with Gibbs Sampling

For the system we are studying, we know that the wave function is positive definite. That is why we can use Gibbs sampling as well. To do this, we will represent the wave function as $\Psi_G(\mathbf{X}) = \sqrt{F_{rbm}(\mathbf{X})}$ instead of $\Psi(\mathbf{X}) = F_{rbm}(\mathbf{X})$. This will only change the kinetic energy part of the local energy. The derivation of the new kinetic energy is done in Appendix B.

This gives the new full local energy expression:

$$
\begin{aligned}
E_L = & -\frac{M}{2\sigma^2} + \frac{1}{2}\sum_i^M \sum_j^N \Sigma(u_j)\cdot\Sigma(-u_j)\cdot\frac{w_{ij}^2}{\sigma^4} + \frac{1}{4}\sum_i^M \frac{X_i - a_i}{\sigma^2} \\
& -\frac{1}{2}\sum_i^M \frac{X_i - a_i}{\sigma^2}\sum_j^N \Sigma(u_j)\cdot\frac{w_{ij}}{\sigma^2} + \frac{1}{4}\sum_i^M\sum_j^N (\Sigma(u_j))^2\cdot\frac{w_{ij}^2}{\sigma^4} \\
& + \frac{1}{2}\sum_{k=1}^M \omega^2 X_k^2 + \sum_{i<j}\frac{1}{X_{ij}}
\end{aligned}
\tag{15}
$$

This new change in the local energy can easily be coded. We clearly see that the only difference in equation 14 and 15 is that the Gibbs local energy has some $\frac{1}{2}$ factors where the other would have a factor of 1. So we only have take into account the half-factors when using the Gibbs sampling in comparison to the other sampling methods numerically.

2.4.3 Benchmark Ground State Energies

For the non-interacting case we can easily find an analytical expression for the local energy. Here we only look at the harmonic oscillator part of the Hamiltonian $\hat{H}_0$ int equation 1. The wave function for one electron in an oscillator potential in 2D can be written in terms of Hermite polynomials, $H_{n_x}(\sqrt{\omega}x)$, and a normalization constant A as:

$$
\phi_{n_x,n_y}(x,y) = AH_{n_x}(\sqrt{\omega}x)H_{n_y}(\sqrt{\omega}y)\exp\left(-\frac{\omega(x^2+y^2)}{2}\right) = \phi_{n_x}(x)\cdot\phi_{n_y}(y) \tag{16}
$$

From the derivation of the ground state energies for non-interacting electrons in Appendix C, we can see that the analytical expression for the local energy (in a.u.) can be written in terms of the frequency ω, number of particles P and number of dimensions D as:

$$
E_L = \frac{\omega}{2}P\cdot D \tag{17}
$$

For analytical ground state energy values for interacting electrons, we use already calculated analytical ground state energies by Taut [5]. He found that the ground state energy for two interacting electrons in 2D with $\omega = 1$ is 3 a.u.

2.5 Statistical Error Analysis

An important part of the results we get are the errors in these results. In most cases the final results will have some errors that have to be considered to actually give the more correct final results. These errors can be categorized into statistical and systematical errors. Systematical errors are method dependent, mostly due to some changing error in often the method used or the machinery producing the data. This means that they have to be considered differently from case to case. The statistical errors are the differences between the value resulting from the experiment and the known exact solution, which in most cases actually is unknown. To calculate the statistical errors we use the central limit theorem which says that the more data points we use, the closer our resulting distribution will be to the normal distribution. To use this theorem, we have to make an assumption that we have independent and identically distributed (iid) events. With the probability distribution $p(x)$ for n measured data points, we can then calculate the mean value

$$\langle x \rangle = \mu = \frac{1}{n}\sum_{i=1}^{n} p(x_i)x_i. \tag{18}$$

With this mean value, we can calculate the sample variance

$$\sigma^2 = \langle x^2 \rangle - \mu^2 = \frac{1}{n}\sum_{i=1}^{n} p(x_i)(x_i-\mu)^2, \tag{19}$$

and the standard deviation

$$\text{STD} = \frac{\sigma}{\sqrt{n}}. \tag{20}$$

For iids there should not be any correlations between the data points. In this project we use a random generator to produce the data points. These random generators do not give 100% uncorrelated data points, which means that the sample variance gets a covariance term as well (also called the sample error)

$$\sigma_m^2 = \frac{\sigma^2}{n} + \text{cov}. \tag{21}$$

Computationally we will use Monte-Carlo calculations, which runs the experiment M number of samples/MC cycles. To account for the possible numerical precision loss we introduce a correlation function

$$f_d = \frac{1}{n-d}\sum_{k=1}^{n-d}(x_k - \langle x_n \rangle)(x_{k+d} - \langle x_n \rangle), \tag{22}$$

where d is the distance between the measurements in the sample samples. This now yields an autocorrelation function

$$\kappa_d = \frac{f_d}{\text{Var}(x)} \tag{23}$$

giving value 1 if there are no correlation between the data. The sample error yields

$$\sigma_m^2 = \frac{\sigma^2}{n} + \frac{2}{n}\sigma^2\sum_{d=1}^{n-1}\frac{f_d}{\sigma^2}.$$

Now we introduce a correction factor τ which we will call the autocorrelation time

$$\tau = 1 + 2\sum_{d=1}^{n-1}\kappa_d. \tag{24}$$

To find this τ factor, we will use what is called a blocking method, which is explained more later. The blocking method will be used on the produced ground state energy data from the VMC simulations to give more correct errors in the produced energies.

3 Numerical Algorithms

3.1 Variational Monte Carlo

To do the numerics in this project we will use a well known numerical method called variational Monte Carl (VMC). The good thing about this method is that we do not need to deal with differential equations which usually are difficult to handle. Instead we use statistical simulation to describe the system. This method uses random sampling of probability distributions through usually a large number of steps to achieve a solution. This type of Monte Carlo simulation needs some sampling technique combined with an algorithm for accepting or declining a proposed move towards the most probable result. In this project we will apply three sampling variants; the Metropolis algorithm, the Importance sampling algorithm and the Gibbs sampling algorithm.

Most quantum systems are many-body problems with a large amount of particles. For such systems, it is often too difficult to obtain exact solutions. That is why the quantum Monte Carlo method is very good for simulating such systems, since we can compute different expectation values like the ground state energy. It is this quantity we will focus on in this project for a system of N_p particles up till two dimensions. We will utilize the variational principle (like in section 2.4) that the expectation value of the Hamiltonian will be an upper bound to the ground state energy E_0 of the system as:

$$E_0 \leq \langle H \rangle = \frac{\int \Psi_T^* \hat{H} \Psi_T d\mathbf{X}}{\int \Psi_T^* \Psi_T d\mathbf{X}} \tag{25}$$

The MC integration is used to avoid the multi-dimensional integral of a expectation value by approximating it to a sum over the number of MC cycles

$$E_{exact} \leq \langle E_L \rangle = \int P(\mathbf{X}, \alpha) E_L(\mathbf{X}, \alpha) d\mathbf{X} \approx \frac{1}{MC} \sum_{i=1}^{MC} E_L(X_i, \alpha), \tag{26}$$

with the same PDF as earlier. From the law of large numbers we would then expect as $MC \to \infty$, the above approximation should become exact.

This VMC scheme will first initialize the trial wave function $\Psi_T(\mathbf{X}, \alpha)$, which is dependent on the variational (RBM) parameters $\alpha = [\mathbf{a}, \mathbf{b}, \mathbf{W}]$ that we want as close as possible to the optimal parameters. Then we initialize the position of the particles for given α, the energy and variance. For a given number of MC cycles we start the MC simulation:

1. Here we will propose a new step $R' = R + r \cdot r_{\text{step}}$, where $r \in [-0.5, 0.5]$ (Metropolis) or $r \in [0, 1]$ (Importance) is drawn at random and r_{step} is a tuned MC step length.

2. Then we calculate the trial wave function and the probability distribution function of the new position.

3. Use a sampling method to test if we should accept or decline the new position.

4. For accepted position we update the initial position to the proposed step, $R = R'$.

5. Then we update the energy and variance.

After the MC simulation we compute the average of the interesting quantities. Then vary the range of α selected and repetition of steps to find the minimum energy.

3.2 Metropolis Algorithm and Sampling Methods

The Metropolis algorithm uses the fundamentals of Markov Chains for a random walk to evolve the probability distribution function from a state j to a state i. This transition is described by the transition probability matrix $W(j \rightarrow i) = W_{j\rightarrow i}$ and is not usually known. This transition matrix can be modeled into a multiplication of two parts; $A(j \rightarrow i)$ which is the probability of accepting a proposed move from j to i, and $T(j \rightarrow i)$ which is the likelihood for the transition from j to i to happen:

$$W(j \rightarrow i) = A(j \rightarrow i) \cdot T(j \rightarrow i) \tag{27}$$

This leads to the probability of the system to occur in the state i after n steps to be:

$$P_i^{(n)} = \sum_j \left[P_j^{(n-1)} T_{j\rightarrow i} A_{j\rightarrow i} + P_i^{(n-1)} T_{i\rightarrow j} (1 - A_{i\rightarrow j}) \right]$$

The probability of making some transition must be one such that $\sum_j T_{j\rightarrow i} = 1$, which yields

$$P_i^{(n)} = P_i^{(n-1)} + \sum_j \left[P_j^{(n-1)} T_{j\rightarrow i} A_{j\rightarrow i} - P_i^{(n-1)} T_{i\rightarrow j} A_{i\rightarrow j} \right]$$

For large n we require that $P_j^{(n\rightarrow\infty)} = p_i$. This gives what is called the balance requirement

$$\sum_j \left[p_j T_{j\rightarrow i} A_{j\rightarrow i} + p_i T_{i\rightarrow j} (1 - A_{i\rightarrow j}) \right],$$

which can be rewritten to a more sturdy requirement called the detailed balance:

$$\frac{A_{j\rightarrow i}}{A_{i\rightarrow j}} = \frac{p_i T_{i\rightarrow j}}{p_j T_{j\rightarrow i}} \tag{28}$$

This requirement is what decides if the new step is accepted or not. In other words, it defines the condition for moving towards the most probable state. This is the foundation of the two Metropolis methods we are using in this project, the brute Metropolis and the Importance samplings.

3.2.1 Metropolis Sampling

The first thing in the standard (brute force) Metropolis algorithm is that we assume the transition likelihood to be symmetric, $T_{j\rightarrow i} = T_{i\rightarrow j}$. Since this is assumed to be symmetric and the acceptance probabilities are unknown, the acceptance ratio is defined as the ratio of the different probabilities. We want the highest possible number of accepted moves in number of proposed steps, which means we want to maximize

$$A_{j\rightarrow i} = \min(1, \frac{p_i}{p_j}).$$

The acceptance ratio in equation 28 will have the following effect: If the ratio is equal to 1 the proposed move is standing still, if the ratio is larger than 1 the proposed move is moving towards the most probable state and if the ratio is smaller than 1 the proposed move is moving away from the most probable state. So for a random number $r \in [-0.5, 0.5]$ (uniformly distributed) we will accept the move if

$$r \leq \frac{|\Psi_T(\mathbf{r}_{k+1})|^2}{|\Psi_T(\mathbf{r}_k)|^2}, \tag{29}$$

and for each MC cycle the new position is proposed

$$R' = R + r \cdot r_{\text{step}}$$

where r_{step} is usually tuned to give acceptance rate of $\approx 50\%$ of the proposed moves. Then the energy and variance is updated before going to the next MC cycle number. Since the step length r_{step} has to be manually tuned, it makes the brute force Metropolis not the most efficient. We will normalize the acceptance ratio by the total number of MC steps such that $A \in [0, 1]$.

3.2.2 Importance Sampling

Instead of having to manually choose the step size like in the brute force method, we want the step length to be determined by the trial wave function of the system. This introduces the Importance sampling where the main goal is to increase the acceptance rate such that we don't use unnecessary time on unwanted moves. This algorithm is based on the Fokker-Planck equation

$$\frac{\partial P(\mathbf{r},t)}{\partial t} = \sum_k D\left(\nabla_k^2 \cdot P(\mathbf{r},t) - D\nabla_k \cdot F(\mathbf{r}_k)P(\mathbf{r},t)\right) = \sum_k D\nabla_k \cdot (\nabla_k - F(\mathbf{r}_k))P(\mathbf{r},t) \tag{30}$$

and the Langevin equation

$$\frac{\partial \mathbf{r}(t)}{\partial t} = DF(\mathbf{r}(t)) + \eta, \tag{31}$$

where D is the diffusion coefficient, F is the drift/quantum force, P is the PDF and η is a uniformly distributed stochastic variable for each dimension. The diffusion coefficient comes from the half-factor in the kinetic energy, and is thus $D = \frac{1}{2}$. The solution to the Fokker-Planck equation with the assumption of stationary densities gives us the quantum force

$$\mathbf{F} = \frac{2\nabla\Psi_T}{\Psi_T}. \tag{32}$$

This force will push the proposed moves towards the regions of configuration space where the trial wave function is large, which should increase the acceptance rate of the moves made. The quantum force is derived in Appendix D.

The Langevin equation can be solved by using Euler's method to find the new position of the particles:

$$\mathbf{r}_{k+1} = \mathbf{r}_k + DF(\mathbf{r}_k)\Delta t + \xi\sqrt{\Delta t} \tag{33}$$

ξ is a Gaussian random variable and Δt is the time step parameter to be chosen.

The Importance sampling algorithm implies that the transition probability can't be omitted any more. The Langevin equation provides with the new trial position, while the

Fokker-Planck equation provides with the new transition probability on the form of Green's function:

$$G(\mathbf{r}_k, \mathbf{r}_{k+1}, \Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp\left(-\frac{(\mathbf{r}_{k+1} - \mathbf{r}_k - DF(\mathbf{r}_k)\Delta t)^2}{4D\Delta t}\right) \tag{34}$$

Equation 29 for the brute force Metropolis is now exchanged with the following Importance algorithm condition for accepting the proposed move:

$$r \leq \frac{G(\mathbf{r}_k, \mathbf{r}_{k+1}, \Delta t)|\Psi_T(\mathbf{r}_{k+1})|^2}{G(\mathbf{r}_{k+1}, \mathbf{r}_k, \Delta t)|\Psi_T(\mathbf{r}_k)|^2} \tag{35}$$

Here r is a Gaussian distributed random number. Also here we normalize the acceptance ratio by the total number of MC steps.

3.2.3 Gibbs Sampling

For the system we are studying, we know that the wave function is positive definite. So with the Gibbs sampling we represent the wave function as $\Psi_G(\mathbf{X}) = \sqrt{F_{rbm}(\mathbf{X})}$ instead of $\Psi(\mathbf{X}) = F_{rbm}(\mathbf{X})$. In this method we will sample from the joint probability of $\mathbf{X}$ and $\mathbf{h}$ in a two step sampling process. For the samples $\mathbf{X}$ by themselves the probability density $|\Psi_G(\mathbf{X})|^2$ is modeled as we wish. The updated samples are generated according to conditional probabilities, which accept with probability 1. We use the Gaussian RBM in the Gibbs sampling such that the two conditional probabilities are

$$P(X_i|\mathbf{h}) = \mathcal{N}(X_i; a_i + \sum_j h_j w_{ij}, \sigma^2) \tag{36}$$

for the positions/visible units which follows a normal distribution, and

$$P(h_j = 1|\mathbf{X}) = \frac{1}{1 + e^{-b_j - \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}} \tag{37}$$

for the hidden nodes which follows a logistic sigmoid function. These two probabilities can be seen as activation functions in the neural network, since they only accept or not.

3.3 Gradient Descent Method

To optimize the Boltzmann machine and minimize the energy computed by the Monte Carlo method, we will use a stochastic gradient descent method (SGD). This method is supposed to find the variational (RBM) parameters $\alpha_i = (a_1, ..., a_M, b_1, ..., b_N, w_{11}, ..., w_{MN})$ that minimize the local energy.

We start by choosing an initial α and find the next value as

$$\alpha_{i+1} = \alpha_i - \eta G_i = \alpha_i - \eta\frac{\partial\langle E_L\rangle}{\partial\alpha_i} = \alpha_i - 2\eta\left[\langle E_L\frac{1}{\Psi}\frac{\partial\Psi}{\partial\alpha_i}\rangle - \langle E_L\rangle\langle\frac{1}{\Psi}\frac{\partial\Psi}{\partial\alpha_i}\rangle\right], \tag{38}$$

where η is the learning rate of the RBM and G_i is the gradient of the local energy with respect to the RBM parameters. This gradient is also the cost function in our neural network. This means we need to calculate the derivative of the NQS wave function (eq. 7) with respect to the RBM parameters. Like we did in the derivation of the kinetic energy in Appendix A, we will use the sigmoid function $\Sigma(u_j)$ (eq. 43) in the following gradients for the Metropolis methods:

$$\frac{1}{\Psi}\frac{\partial \Psi}{\partial a_k} = \frac{X_k - a_k}{\sigma^2}$$
$$\frac{1}{\Psi}\frac{\partial \Psi}{\partial b_k} = \Sigma(u_k)$$
$$\frac{1}{\Psi}\frac{\partial \Psi}{\partial w_{kl}} = \frac{X_k}{\sigma^2} \cdot \Sigma(u_l)$$

These derivatives needs to be sampled in the VMC program during the MC steps to get the gradient of the local energy.

In case of the Gibbs sampling, we use Ψ_G wave function which gives us a different gradient of the local energy:

$$\frac{1}{\Psi_G}\frac{\partial \Psi_G}{\partial a_k} = \frac{X_k - a_k}{2\sigma^2}$$
$$\frac{1}{\Psi_G}\frac{\partial \Psi_G}{\partial b_k} = \frac{1}{2}\Sigma(u_k)$$
$$\frac{1}{\Psi_G}\frac{\partial \Psi_G}{\partial w_{kl}} = \frac{X_k}{2\sigma^2} \cdot \Sigma(u_l)$$

The only difference is also here a $\frac{1}{2}$-factor in each of the derivations, which is also easily implemented numerically.

3.4 Blocking

To perform a statistical analysis (sect. 2.5) of the Monte Carlo calculation we will use the blocking resampling method which is the better resampling method for bigger data sets. The blocking method is well explained by Flyvbjerg and Petersen [3], and we will use a more improve blocking method developed by Jonsson [4] in this project.

The basic idea of the blocking method is that we start with a correlated data set which we will transform into an uncorrelated data set which we can use statistical analysis on. If we start with a data set $\mathbf{X} = (X_1, X_2, \cdots, X_n)$. The data points X_i in $\mathbf{X}$ will be turned into new vectors with the mean of subsequent pairs for each k blocking transformation. We define then $\mathbf{X}_i$ recursively for all $1 \leq i \leq d-1$:

$$(\mathbf{X}_0)_k = (\mathbf{X})_k$$
$$(\mathbf{X}_{i+1})_k = \frac{1}{2}\left((\mathbf{X}_i)_{2k-1} + (\mathbf{X}_i)_{2k}\right)$$

These new vectors are less correlated since the elements are further away from each other. By following the derivation by Jonsson [4], the variance of the sample mean of $\mathbf{X}_k$ after k blocking transformations is

$$\sigma_b^2(\overline{\mathbf{X}}_k) = \frac{\sigma_k^2}{n_k} + \frac{2}{n_k}\sum_{h=1}^{n_k-1}\left(1 - \frac{h}{n_k}\right)\gamma_k(h) = \frac{\sigma_k^2}{n_k} + e_k \quad \text{if} \quad \gamma_k(0) = \sigma_k^2, \tag{39}$$

where γ_k is the auto-covariance of $\mathbf{X}_k$ and e_k is called the truncation error. Flyvbjerg and Petersen [3] showed that the truncation error goes to zero as k increases. After e_k is small enough we should get a better estimation of the error in our results. This error should also be larger than we first calculated since it accounts for the correlations.

4 Methods

The main computation of the ground state energies in this project is done in C++, while the blocking and plotting are done in Python 3.7. All the program-files are found at the GitHub repository [1] among with various data files and figures produced. The programming of the VMC scheme is written in a object oriented way with a class structure. The code structure is more explained in the program itself.

The SGD method is applied many times iteratively in the VMC method to ensure that the set of randomly initiated α gives a lower energy. The final result should then be a good estimate to the ground state energy. After N_{SGD} number of SGD cycles of smaller MC runs (for saving time), we do a final set of the final parameters for a larger number of MC cycles. The local energy for each SGD cycle, together with the standard deviation and the values of the RBM parameters, are exported to data files and used for plotting. The final run exports the local energy for each MC cycles to a separate data file to be used for statistical analysis (blocking).

This works well with non-interacting electrons which gives converges to an energy with a small fluctuation. For interacting electrons, we do not separate the final run for the final local energy. Here we use all the SGD cycle data to perform a linear fit to the local energies after equilibrium. This mean energy is then used as the final result with a statistical analysis.

4.1 Non-Interacting Electrons

4.1.1 Metropolis Sampling

Start by implementing the standard Metropolis algorithm (sect 3.2.1) in the VMC program. Now we will study the ground state energy of one electron in 1D with two hidden nodes. First we must find the optimal step length L which gives the best compromise between acceptance rate and how fast we reach equilibrium. Test $L = [0.01, 0.1, 0.5, 1.0, 2.0]$ with learning rate $\eta = 0.1$, $\omega = 1$, $N_{SGD} = 100$, $N_{MC} = 1e5$, $N_{MC,final} = 1.5e6$ and $\sigma = 1.0$. To find the best step length, we plot the local energy for each step length as a function of the number of SGD cycles to see equilibrium time, and we look at the acceptance rate after the final run. With the optimal L we test different learning rates $\eta = [0.01, 0.05, 0.1, 0.2, 0.5, 0.6]$. Then we plot the local energy for each learning rate as a function of the number of SGD cycles to find the optimal η value. With the optimal L and η, we use $N_{SGD} = 150$ and vary the number of particles $N_P = [1, 2]$, dimensions $D = [1, 2]$ and hidden nodes $N_H = [1, 2, 3, 4]$ to see how the final local energy changes. With the final energy for each case, we do a statistical analysis with blocking which we will use to compare with the benchmarked energy (eq. 17) and between the sampling methods later.

4.1.2 Importance Sampling

Now we implement the Importance sampling algorithm (sect. 3.2.2) into the VMC method. With this sampling method we will do the same analysis on the same system as for the standard Metropolis, except that we first test various time steps Δt instead of step lengths. We use the same other parameters when testing $\Delta t = [0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 2.0]$ as for testing L. We plot the local energy as we did earlier, and use the optimal Δt to test the same learning rates. This is also plotted as before to find the best η parameter for this sampling method. With the two optimal parameters, we vary the number of particles, dimensions and hidden nodes as earlier to get the ground state energy with a statistical analysis.

4.1.3 Gibbs sampling

Then we implement the Gibbs sampling algorithm (sect. 3.2.3) into the VMC method. We do the exact same as for the two earlier sampling methods, only that we change the neural network $\sigma = [0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$ instead of L or Δt. Once again we plot the local energy for changing the σ and η before doing a statistical analysis on the final local energies for the various number of particles, dimensions and hidden nodes.

After doing the statistical analysis of the local energies, we compare these energies produced by the three sampling methods with the analytical benchmark energies to see which method achieves the closest local energies to the analytical energies.

4.2 Interacting Electrons

Since we do not have accounted for Pauli's principle, we will only look at a system of two electrons in 2D. With the optimal values for the parameters L, Δt and σ for the three sampling methods we now add interaction to the system. For each of the three sampling methods we once again vary the learning rate $\eta = [0.1, 0.2, 0.5, 0.6]$ and plot the local energies as function of the number of SGD cycles, which is now $N_{SGD} = 600$. For Metropolis and Importance sampling we use $\sigma = 1.0$, while in Gibbs sampling we use the optimal σ we found for non-interacting Gibbs sampling.

With the respective optimal η-values for the sampling methods, we experiment with the number of hidden nodes and plot the local energies to see how the hidden nodes affect the computed local energies. On these computed energies we do a linear fit after equilibrium to get a mean energy with a statistical analysis. These mean energies, for each hidden node tested, are compared with the analytical result (sect. 2.4.3) of $E_L = 3$ a.u., and compared between the three sampling methods to see which is the better method.

5 Results

All the figures produced in this project can be found in the Figures folder at the GitHub-repository [1]. The produced data files can be found in the Data folder at the same repository.

5.1 Non-Interacting Electrons

5.1.1 Metropolis Sampling

The first thing we have to test is how the local energy as a function of the SGD cycles behaves as we vary the step length L of the Metropolis sampling in the VMC. The parameters we use is described in the Methods section 4.1.1. This is seen in Figure 2. There we see that when $L \geq 0.5$, we get very stable curves for the local energy and they all converge towards the analytically calculated ground state energy of $E_L = 0.5$ (in this case). From this we can conclude that the step length should be $L \geq 0.5$. At the GitHub Figures folder there is also a plot for $L = 0.01$, but this gave such a bad solution and did not converge very well that we did not include it here. To find the best step length, we also look at the acceptance rate we get in Table 1. There we see that the smaller the step length is, the better is the acceptance rate. What we want with the step length is that the solution reaches equilibrium as fast as possible, with as high as possible acceptance rate. So we want a compromise between a high acceptance rate and a fast equilibrium of the solution. This seems to be good when we use $L = 0.5$, since the acceptance rate is very good, $A = 0.935$ and the equilibrium is reached quite fast around 20 SGD cycles.

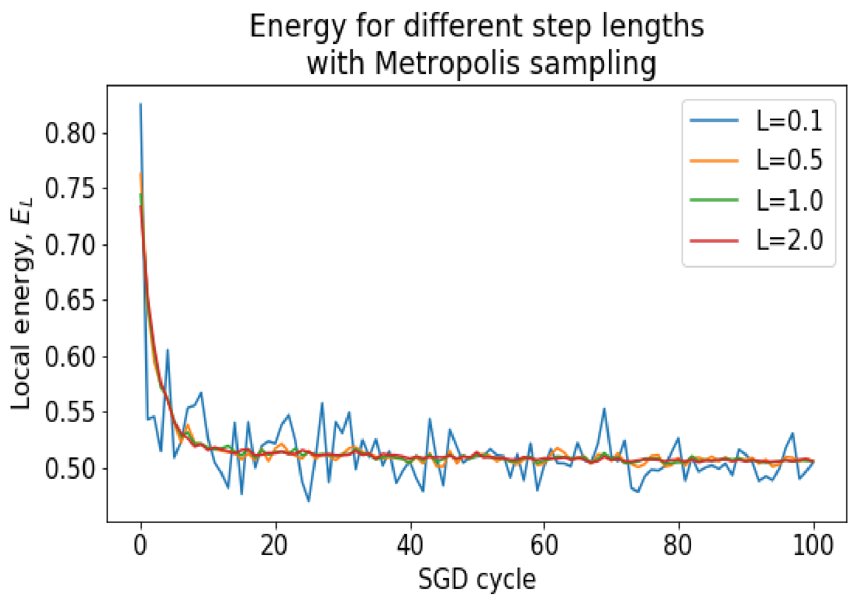

Figure 2: Local energy as function of the SGD cycle for various Metropolis step lengths. Here we see that for $L \geq 0.5$ we get more or less stable curves, and they all reach an equilibrium around 20 SGD cycles.

L	0.01	0.1	0.5	1.0	2.0
A	0.999	0.987	0.935	0.870	0.748

Table 1: Table for the acceptance rate (A) of the Metropolis sampling for various step lengths (L). See here that the smaller the step length is, the bigger and better is the acceptance rate.

With the chosen step length $L = 0.5$, we experiment with the learning rate to see how it affects the system. In Figure 3 we see how the local energy behaves as a function of the SGD cycles for various learning rates. For higher learning rates we see a faster convergence time. The change in learning rate does not affect the acceptance rate of the system. When $\eta = 0.01$, the solution does not converge to the analytical solution within 100 SGD cycles like the others. When $\eta > 0.6$, the solution simply diverges and is not included here. So we choose $\eta = 0.2$ as the learning rate parameter, since the solution converges under 20 SGD cycles and does not converge too fast.

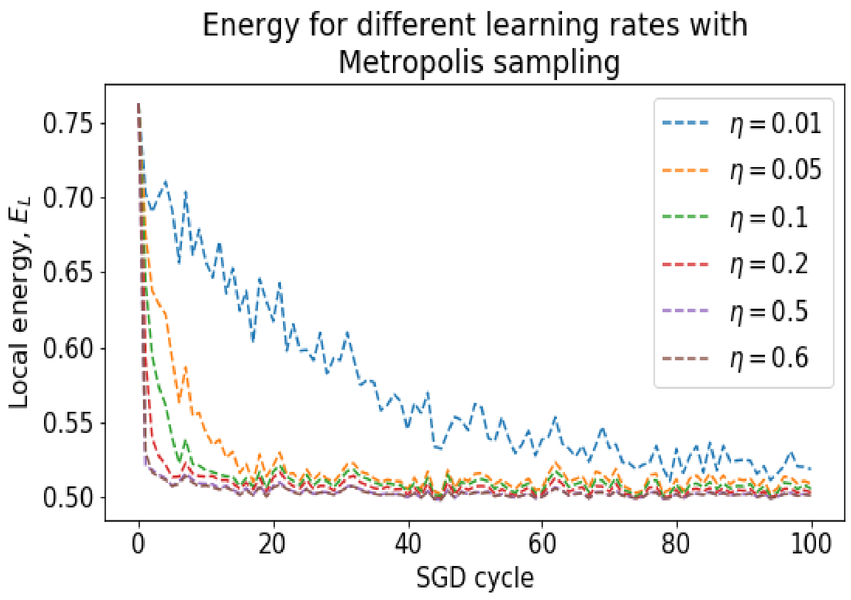

Figure 3: Local energy as function of the SGD cycle for various learning rates with the Metropolis sampling and $L = 0.5$. See here that the higher the learning rate is, the faster the solution reaches the equilibrium for these selected values of learning rates.

5.1.2 Importance Sampling

Then we do the same analysis with the Importance sampling instead of Metropolis. The parameters we use are described in the Methods section 4.1.2. Now we have to find the best time step parameter Δt, which satisfies the same compromise as for L in Metropolis. In Figure 4 we see how the local energy as a function of the SGD cycles behaves for various time steps. The higher Δt is, the better is the convergence towards the exact solution. For $\Delta t \geq 0.05$, we see that the convergence of the solution becomes quite good. So we want a $\Delta t \geq 0.05$. Once again we look at the acceptance rate in Table 2 for the different time steps to find the best time step parameter for the sampling method. When $\Delta t > 0.5$ the acceptance rate is not that good. When $\Delta t < 0.1$ the acceptance rate is too close to zero, meaning that we accept more or less every move such that we do not move enough towards the minimum energy of the system. By looking at the acceptance rate, $\Delta t = 0.5$ seems to be the best parameter, which fits well with our conclusion from Figure 4.

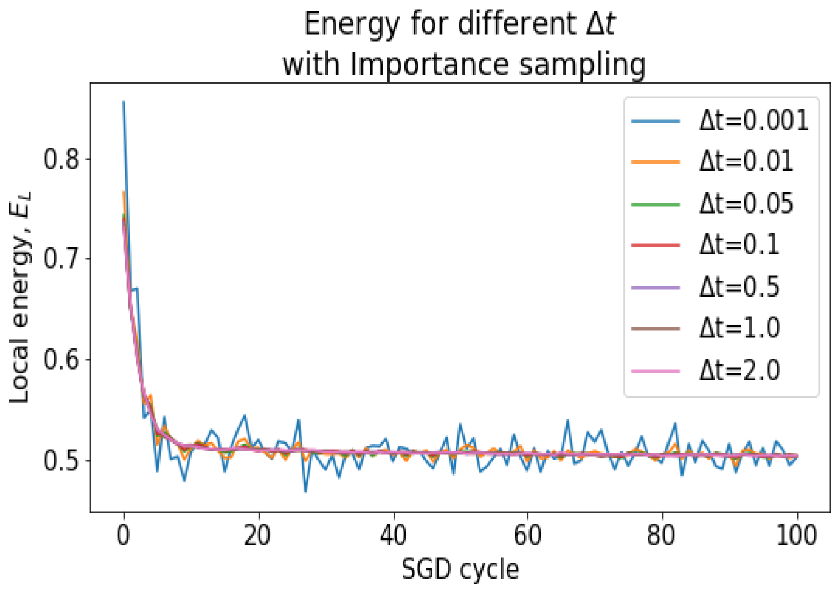

Figure 4: Local energy as function of the SGD cycle for various Importance sampling time steps. For $\Delta t \geq 0.05$ we get smooth curves and fast convergence for all parameters used.

Δt	0.001	0.01	0.05	0.1	0.5	1.0	2.0
A	0.999996	0.9998	0.998	0.994	0.93	0.81	0.56

Table 2: Table for the acceptance rate (A) of the Importance sampling for various time step (Δt). See here that the smaller the time step is, the bigger and better is the acceptance rate.

With the chosen time step $\Delta t = 0.5$, we experiment with the learning rate once again. In Figure 5 we see how the local energy behaves for various learning rates. We see a similar behavior of the solution as we got with Metropolis in Figure 3. The only difference seems to be that the curves with Importance sampling are much smoother than with Metropolis. Since they are so similar, we make the same conclusion for the best learning rate $\eta = 0.2$.

5.1.3 Gibbs Sampling

The last sampling method we use is the Gibbs sampling. The parameters we use are described in the Methods section 4.1.3. For this sampling method we have to tune the σ parameter which we used as $\sigma = 1.0$ for the other two sampling methods. In Figure 6 we see how the local energy as a function of the SGD cycles behaves for various sigmas. From the figure we can clearly see that the Gibbs sampling converges towards different energies for different sigmas. So the Gibbs sampling is very dependent on which value we use for σ. Here we see that when we use $\sigma = 0.6$ or 0.7, the solution converges towards the analytical ground state energy. $\sigma = 0.6$ reach equilibrium in fewer SGD cycles than $\sigma = 0.7$ does, which is why we might assume $\sigma = 0.6$ as the best Gibbs parameter in this case. With this sampling method, the acceptance rate is always 1. So we get no new information by looking at the acceptance rate when choosing the best sigma parameter.

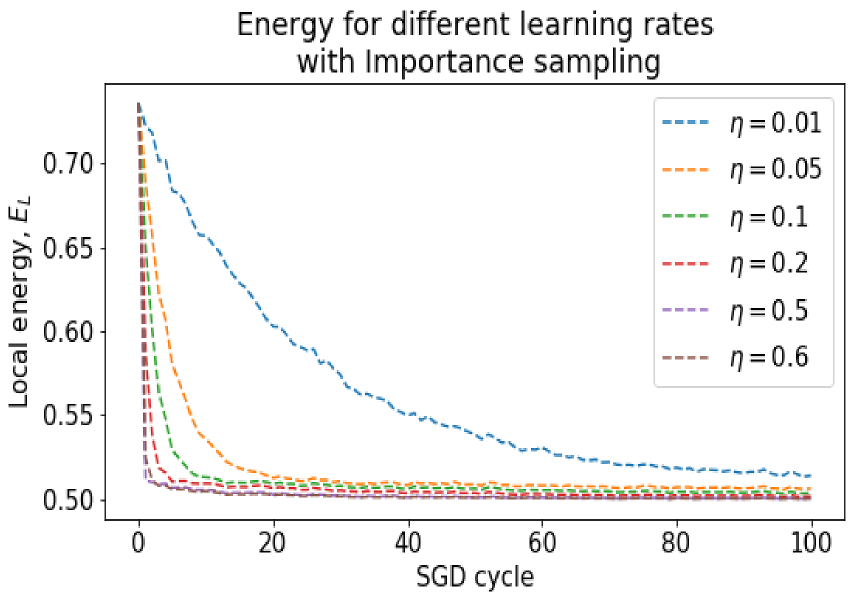

Figure 5: Local energy as function of the SGD cycle for various learning rates with the Importance sampling and $\Delta t = 0.5$. See here that the higher the learning rate is, the faster the solution reaches the equilibrium. Get very similar curves as for Metropolis, but in this case the curves are much smoother.

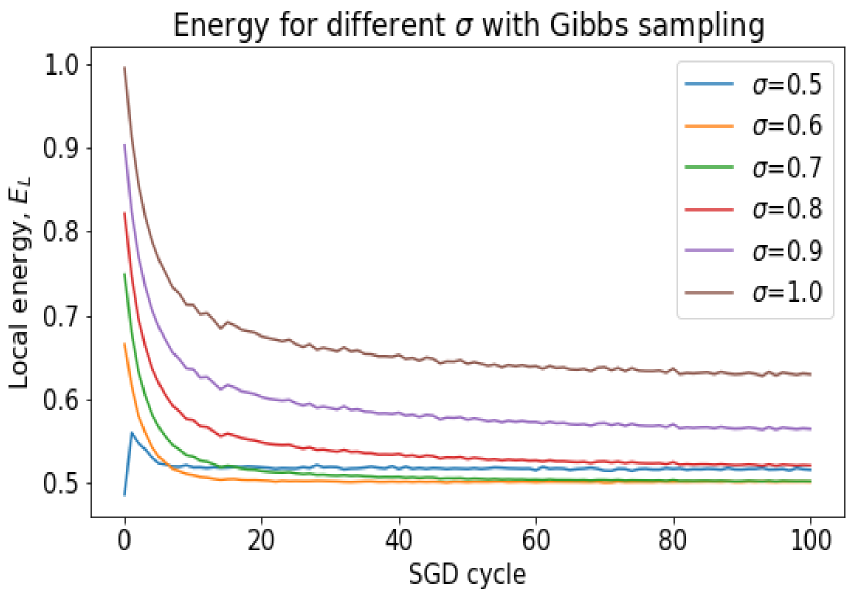

Figure 6: Local energy as function of the SGD cycle for various Gibbs sampling sigmas. For $\sigma = [0.6, 0.7]$ we see convergence towards the analytical ground state energy.

With $\sigma = 0.6$, we experiment with the learning rate. In Figure 7 we see how the local energy behaves for various learning rates. The plots are similar to Importance sampling, except that for $\sigma \geq 0.5$ the curves jump at the beginning before a fast convergence towards the exact solution. The best learning rate parameter seems to be $\eta = 0.2$, as for the other sampling methods.

In Figure 8 we see a comparison of the local energy with the two best learning rates for both $\sigma = 0.6$ and 0.7. There we see that $\eta = 0.2$ is the best learning rate for both sigmas. For the smaller sigma, we reach equilibrium faster than for the bigger. From these figures we could conclude with that $\sigma = 0.6$ would be the best parameter to use with Gibbs sampling. By further testing of the local energy with these sigmas, the final local energy seems to be best when we use $\sigma = 0.7$. So even though $\sigma = 0.6$ reaches equilibrium the fastest, it's $\sigma = 0.7$ that gives the best final energy. That is why we use $\sigma = 0.7$ as the best Gibbs sampling parameter.

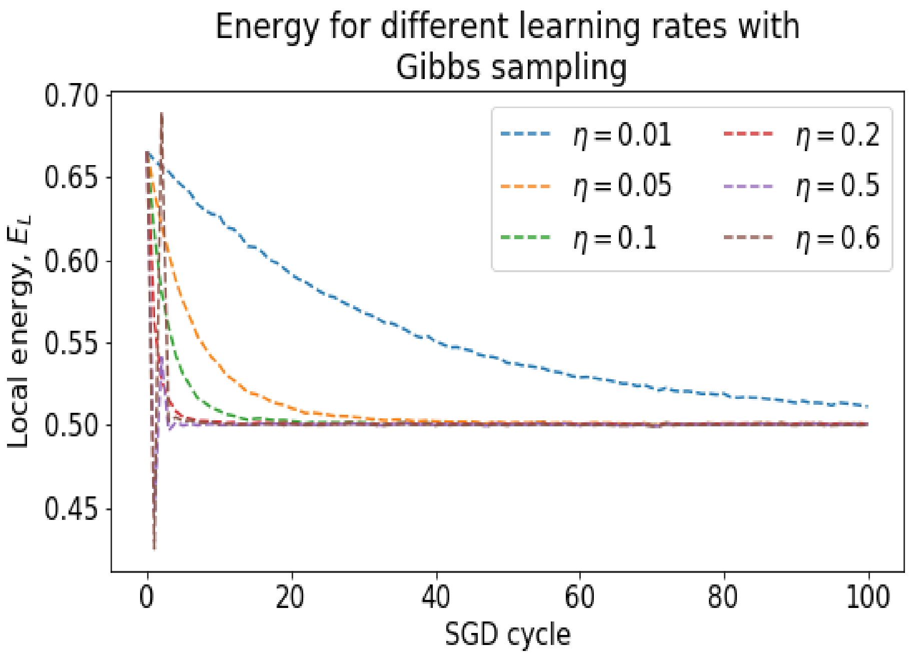

Figure 7: Local energy as function of the SGD cycle for various learning rates with the Gibbs sampling and $\sigma = 0.6$. See here that the higher the learning rate is, the faster the solution reaches the equilibrium. Get very similar curves as for Importance sampling, except for the two biggest parameters which deviates some in the beginning.

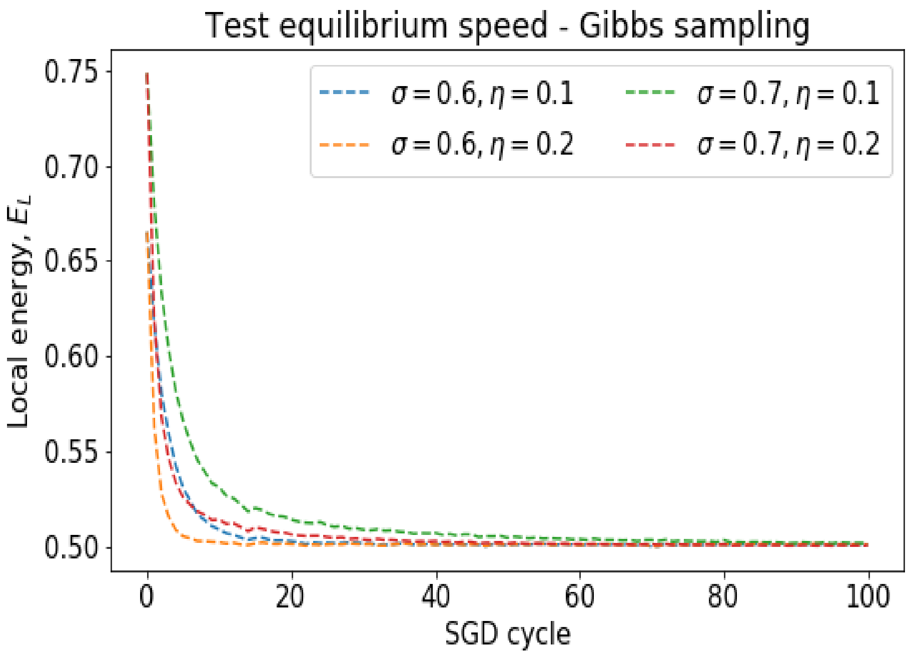

Figure 8: Local energy as function of the SGD cycle for various learning rates with the Gibbs sampling and $\sigma = 0.6$ and $\sigma = 0.7$. Further check for the best sigma and learning rate parameters.

5.1.4 Local Energy Analysis of Sampling Methods

With all the best parameters for the three sampling methods, we will look at the final energies produced from each sampling method when we vary the number of particles (N_P), dimensions (D) and hidden nodes (N_H). In Table 3 we see the local energies for non-interacting electrons produced by each sampling method as described in the caption of the table. To produce these local energies we use $N_{SGD} = 150$ and $\eta = 0.2$ for all three sampling methods. The Metropolis step length is $L = 0.5$, and the Importance time step is $\Delta t = 0.5$. With these two sampling methods we also use $\sigma = 1.0$, while we use $\sigma = 0.7$ with Gibbs sampling.

The most accurate sampling method seems to be Gibbs, since the produced local energies are on average more close to the analytical ground state energies in terms of both accuracy and the errors (from blocking). In the case of CPU time for doing the computation, the Metropolis is the fastest sampling method while Importance sampling gave the slowest CPU time. For all three sampling methods we see that the local energies for $N_P = 1, D = 2$ and $N_P = 2, D = 1$ are the same for each sampling method respectively. From this it seems that the computation of the energy is the same when the sum of the number of particles and dimensions are the same with that sampling method.

Another thing we notice when we increase the number of hidden nodes but keep N_P and D the same, is that the average E_L with blocking increases by a small amount each time N_H is increased. On average, the errors also increase as the number of hidden nodes as increased. This also goes for the time to do the full MC calculation for each sampling method.

N_P	D	N_H	E_A	E_L^M	E_L^I	E_L^G
1	1	1	0.5	$0.500 \pm 1.6 \cdot 10^{-8}$	$0.500 \pm 3.6 \cdot 10^{-9}$	$0.500 \pm 1.4 \cdot 10^{-5}$
1	1	2	0.5	$0.501 \pm 5.0 \cdot 10^{-4}$	$0.501 \pm 6.0 \cdot 10^{-5}$	$0.500 \pm 2.3 \cdot 10^{-5}$
1	2	1	1.0	$1.050 \pm 3.5 \cdot 10^{-3}$	$1.001 \pm 8.1 \cdot 10^{-5}$	$1.000 \pm 2.6 \cdot 10^{-5}$
1	2	2	1.0	$1.002 \pm 5.6 \cdot 10^{-4}$	$1.001 \pm 1.1 \cdot 10^{-4}$	$1.000 \pm 2.9 \cdot 10^{-5}$
2	1	1	1.0	$1.050 \pm 3.5 \cdot 10^{-3}$	$1.001 \pm 8.1 \cdot 10^{-5}$	$1.000 \pm 2.6 \cdot 10^{-5}$
2	1	2	1.0	$1.002 \pm 5.6 \cdot 10^{-4}$	$1.001 \pm 1.1 \cdot 10^{-4}$	$1.000 \pm 2.9 \cdot 10^{-5}$
2	2	1	2.0	$1.999 \pm 3.0 \cdot 10^{-4}$	$2.000 \pm 6.4 \cdot 10^{-5}$	$2.000 \pm 3.0 \cdot 10^{-5}$
2	2	2	2.0	$2.000 \pm 5.1 \cdot 10^{-4}$	$2.001 \pm 1.1 \cdot 10^{-4}$	$2.000 \pm 2.7 \cdot 10^{-5}$
2	2	3	2.0	$2.002 \pm 8.5 \cdot 10^{-4}$	$2.002 \pm 1.7 \cdot 10^{-4}$	$2.000 \pm 3.4 \cdot 10^{-5}$
2	2	4	2.0	$2.001 \pm 9.3 \cdot 10^{-4}$	$2.002 \pm 2.0 \cdot 10^{-4}$	$2.000 \pm 3.5 \cdot 10^{-5}$

Table 3: Table for the final local energy (in a.u.) of non-interacting electrons with the three sampling methods Metropolis (E_L^M), Importance (E_L^I) and Gibbs (E_L^G) with blocking errors, and the analytical ground state energy (E_A).

5.2 Interacting Electrons

With interaction between the electrons, we will do the same analysis of the system as we did for non-interacting electrons. The difference is that we now only look at two electrons in 2D since Pauli's principle is not accounted for. In this case we expect the analytical ground state energy to be 3 a.u (sect. 2.4.3). The numerical parameters we use are described in the Methods section 4.2.

5.2.1 Metropolis Sampling

We use the same step length $L = 0.5$ that we found in the non-interacting case for Metropolis sampling first. Then we once again will experiment with the learning rate of this interacting system. In Figure 9 we see how the local energy behaves for various learning rates. Here we see that as the learning rate increase, the faster the solution converges to equilibrium. Since there is such a big spread of data points for the local energy, it is not easy to see what the solution converges towards exactly. This can be fixed by doing a linear regression fit of the data points. By doing a linear regression on the data points after equilibrium for each learning rate respectively, we get that they do not converge exact to the exact ground state energy. The computed energies after linear regression gives the converged energies around 3.08 a.u., which is a little higher than the exact energy of 3 a.u. From this figure we can conclude that $\eta = 0.2$ seems to give the best solution like for non-interacting Metropolis.

The interacting solution uses more SGD cycles to reach equilibrium than the non-interacting system. So the system needs more cycles to adapt to the interacting wave function than for a non-interacting wave function. The non-interacting Metropolis only needed around 20 SGD cycles to reach equilibrium, while the best case here needs around 250-300 SGD cycles to reach equilibrium.

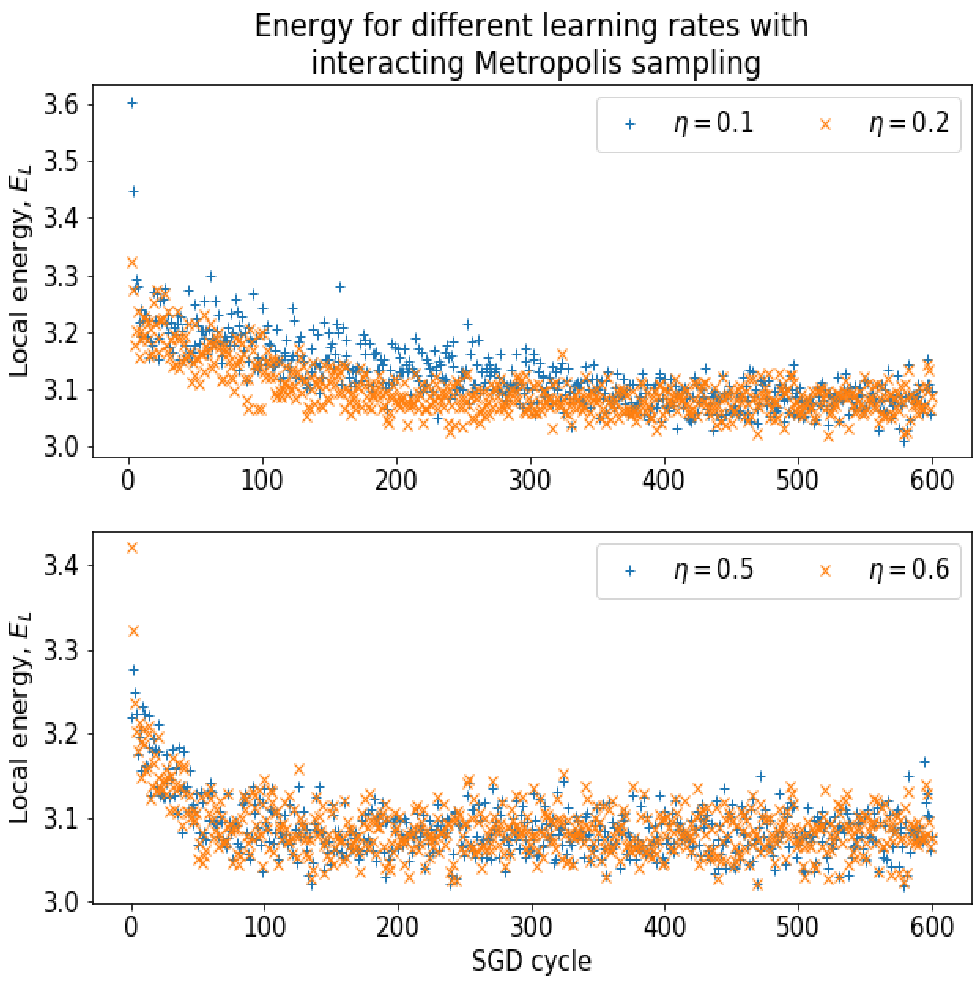

Figure 9: Local energy as function of the SGD cycle for various learning rates with the Metropolis sampling and $L = 0.5$ for two interacting electrons in 2D. The higher the learning rate is, the faster the solution converges. The energy curves are much thicker compared to non-interacting electrons in Figure 3.

Changing the number of hidden nodes did not change much for the non-interacting system. We test now the interacting system in terms of the number of hidden nodes to see if we get any differences. In Figure 10 we see the local energy when we use up till four hidden nodes in our neural network. Using one hidden node gives a much higher energy after equilibrium than the other three cases. It looks like when using a higher number of hidden nodes, the system manages to reach equilibrium in fewer SGD cycles. With three and four hidden nodes we get very similar behavior of the energy solutions, but we can barely see that four nodes drops the solution a little bit faster.

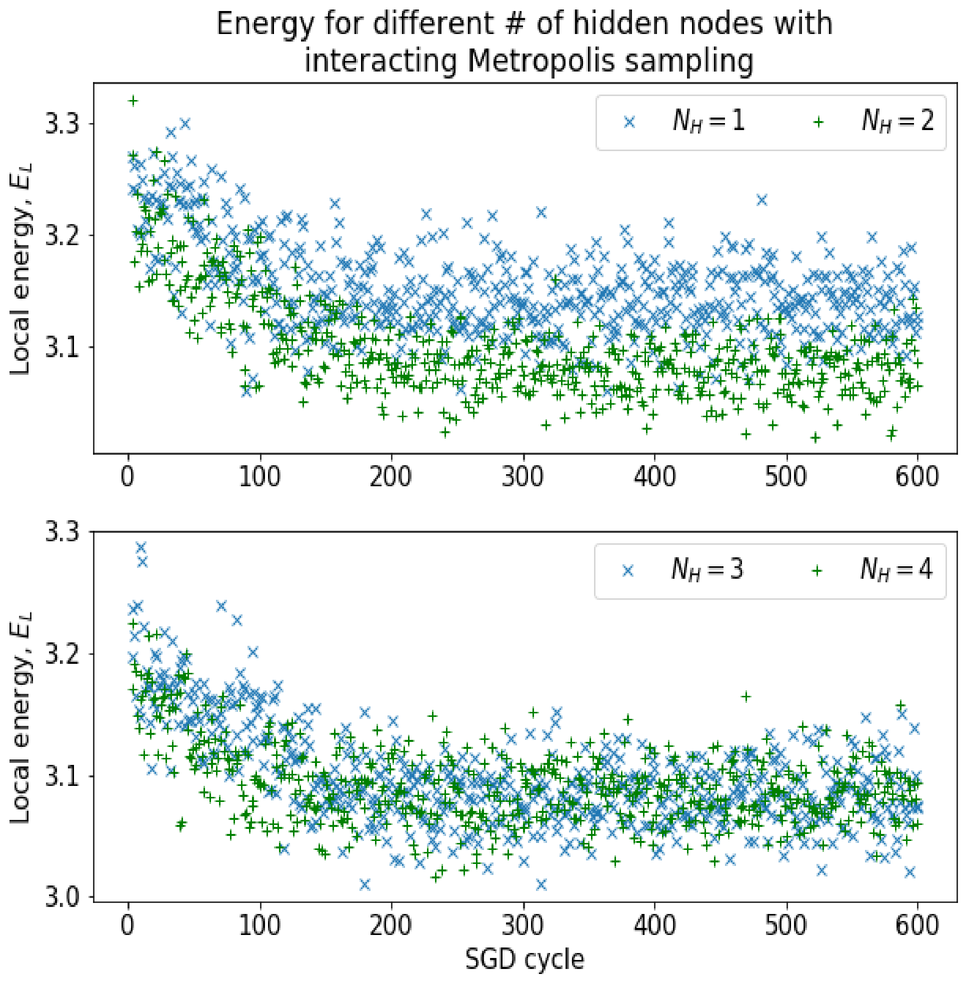

Figure 10: Local energy as function of the SGD cycle for various number of hidden nodes with the Metropolis sampling, $L = 0.5$ and $\eta = 0.2$ for two interacting electrons in 2D. Using one hidden node does not give as good solution as the two, three and four hidden nodes.

5.2.2 Importance Sampling

In the non-interacting case with Importance sampling when varying the learning rate, we got a much smaller spread of data points/smoother curve compared to Metropolis sampling. This is also the case in the interacting system with Importance sampling. By looking at Figure 11 we see that the spread of data points/thickness of the curve is smaller/thinner than for interacting Metropolis. The form of the curves are the same for both cases, where a higher learning rate gives faster equilibrium convergence. For $\eta = 0.1$, the solution takes a lot of SGD cycles before reaching equilibrium compared to the other learning rates. We use the same conclusion like before for why we choose the same learning rate $\eta = 0.2$ here as well.

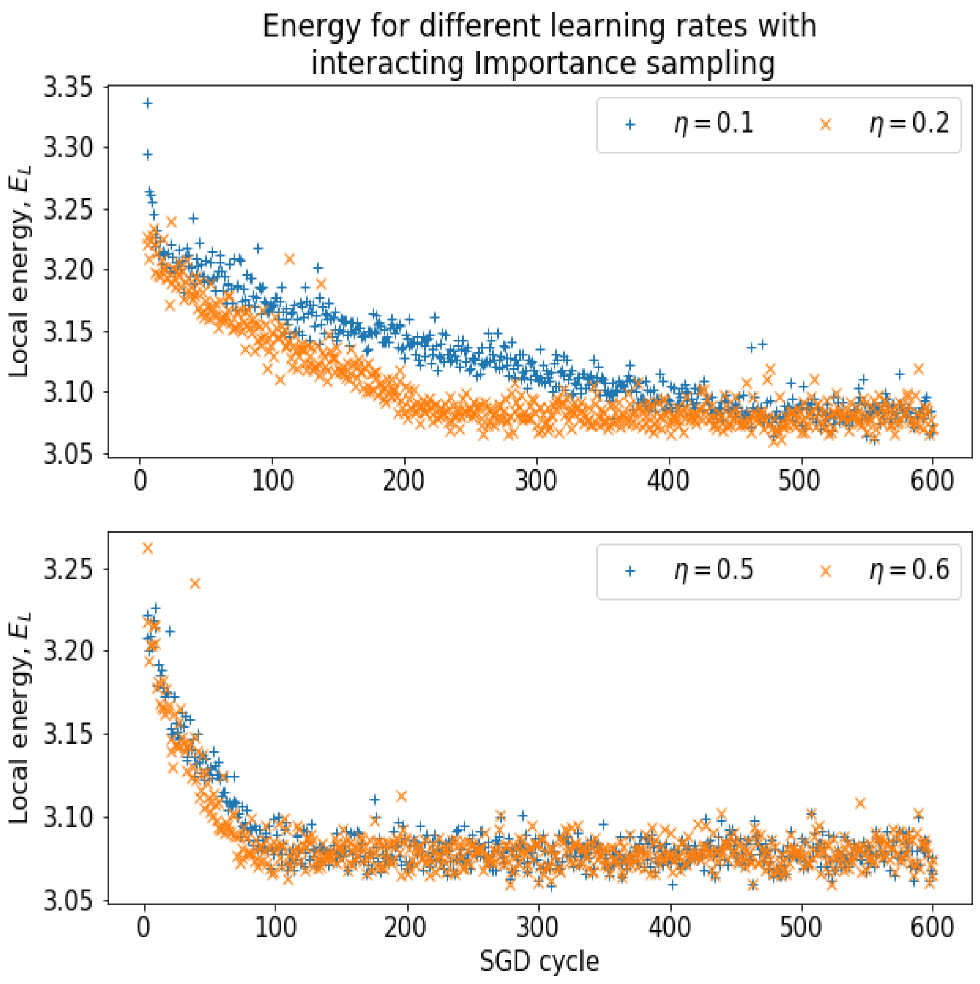

Figure 11: Local energy as function of the SGD cycle for various learning rates with the Importance sampling and $\Delta t = 0.5$ for two interacting electrons in 2D. The higher the learning rate is, the faster the solution converges. The energy curves are much thinner compared to interacting Metropolis in Figure 9.

In Figure 12 we vary the number of hidden nodes with Importance sampling. As with the learning rate, the spread of data points are much smaller for Importance than for Metropolis. Now we more clearly see that as the number of hidden nodes increases, the quicker in terms of SGD cycles the solution reaches its equilibrium. Still they converge to more or less the same local energy. Again, only using one hidden node converges to a higher energy than for two, three and four hidden nodes which seem to converge to around energy of 3.08 a.u.

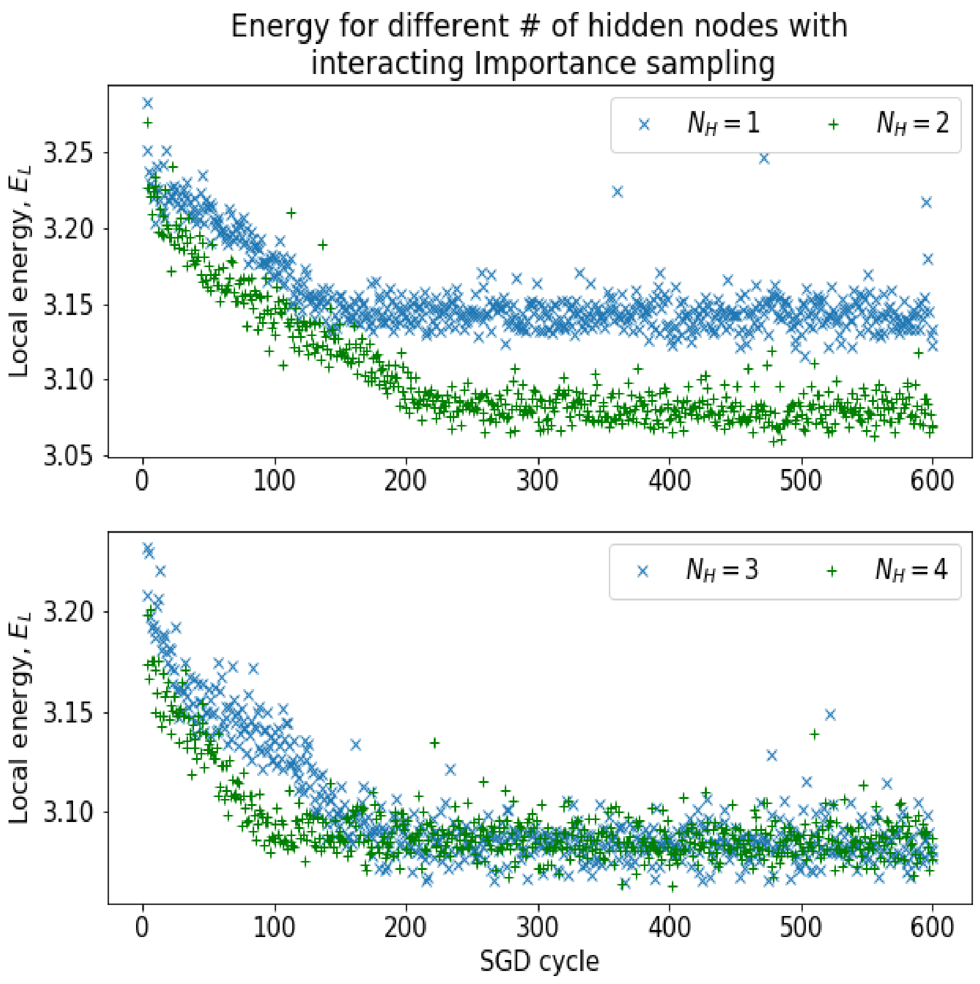

Figure 12: Local energy as function of the SGD cycle for various number of hidden nodes with the Importance sampling, $\Delta t = 0.5$ and $\eta = 0.2$ for two interacting electrons in 2D. Using one hidden node does not give as good solution as using two, three and four hidden nodes.

5.2.3 Gibbs Sampling

As with the non-interacting case, we use the same Gibbs parameter $\sigma = 0.7$. With this parameter we test the learning rate with Gibbs sampling. In Figure 13 we see that the local energy drops quicker towards equilibrium than the other sampling methods, before it oscillates a little around the equilibrium energy. The local energy which the solutions converge towards wit Gibbs sampling, are clearly a little closer to the analytical ground state energy than the other two sampling methods when we vary the learning rate. Like the other cases, a higher learning rate gives faster equilibrium where $\eta = 0.1$ uses a lot more SGC cycles to reach equilibrium than the others. The two highest learning rates are close to each other like earlier. The best learning rate seems to be $\eta = 0.2$, like before.

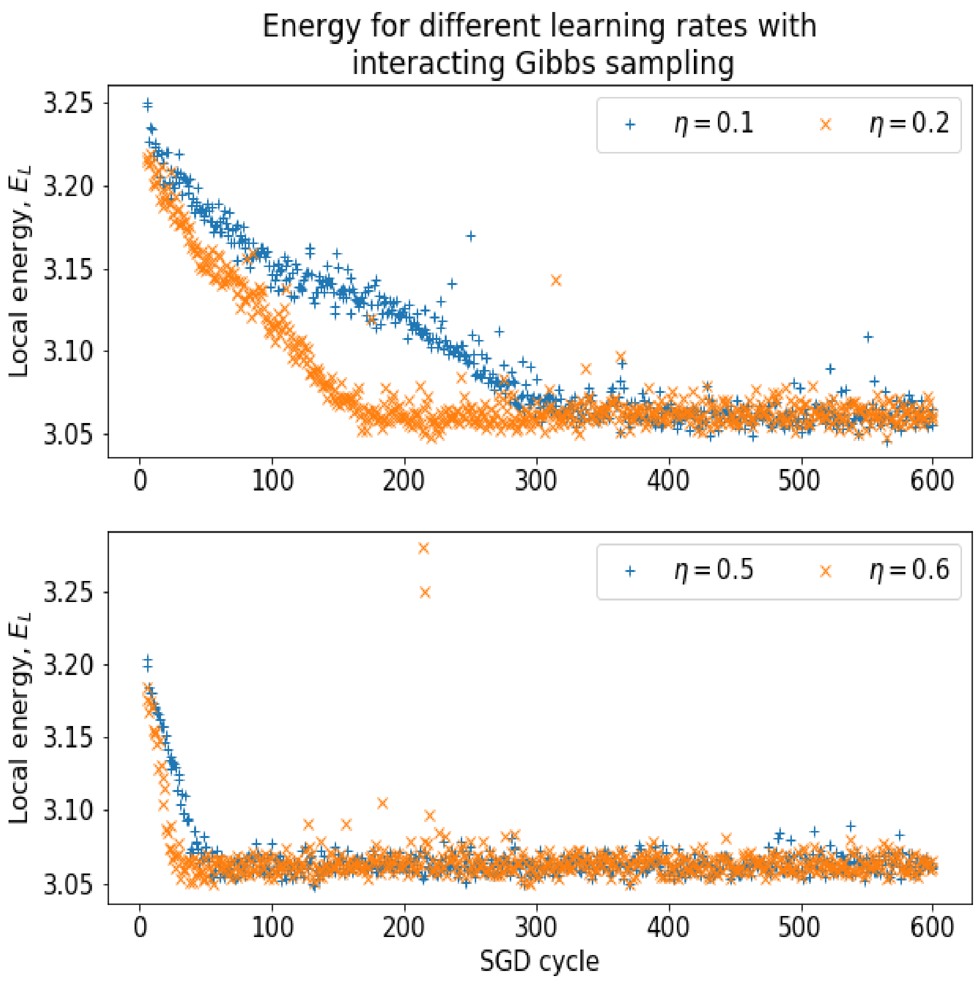

Figure 13: Local energy as function of the SGD cycle for various learning rates with the Gibbs sampling and $\sigma = 0.7$ for two interacting electrons in 2D. The higher the learning rate is, the faster the solution converges towards equilibrium energy.

In Figure 14 we vary the number of hidden nodes using the best learning rate with Gibbs sampling. Like for Metropolis and Importance, using one hidden node gives a much higher local energy. With two, three and four hidden nodes we get convergence towards a local energy of around 3.05-3.06 a.u. Once again, using more hidden nodes in our NN leads to a faster convergence towards equilibrium. After around 250 SGD cycles we see that all solutions seem to have reached equilibrium energy.

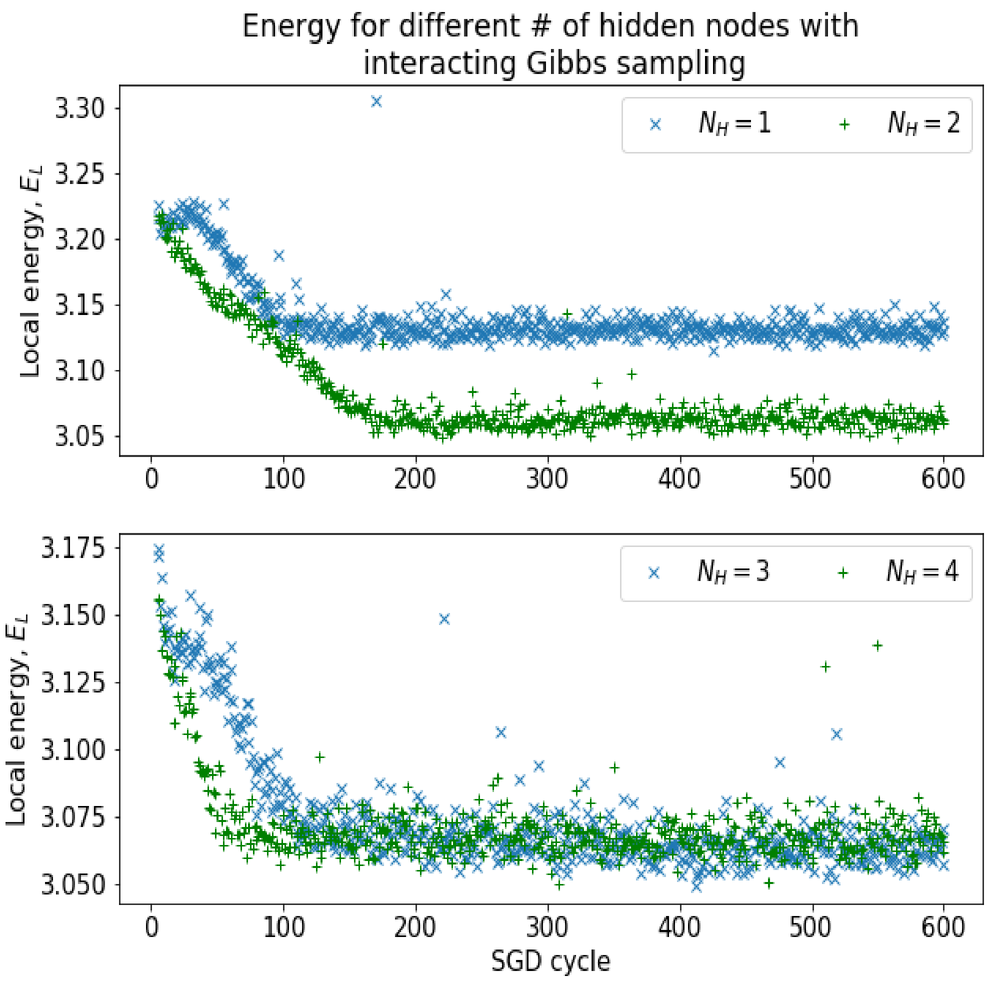

Figure 14: Local energy as function of the SGD cycle for various number of hidden nodes with the Gibbs sampling, $\sigma = 0.7$ and $\eta = 0.2$ for two interacting electrons in 2D. Using more than one hidden node seem to give good convergence towards equilibrium.

5.2.4 Local Energy Analysis of Sampling Methods

With the best parameters for the three sampling methods, we look more at how the computed local energies are as we vary the number of hidden nodes. The final local energies for two interacting electrons in 2D can be seen in Table 4. The blocking method is applied on the produced energy data with a linear regression fit after equilibrium. Here we have used $N_{SGD} = 600$ and $\eta = 0.2$ for all the sampling methods, and used the same best sampling parameters as in the non-interacting cases.

In the non-interacting case we got that the Gibbs sampling gave the best solutions. We would also expect this in the interacting case since the sampling method should not change the solution of the model that significantly, this is also the case with interaction in our system. In this case the standard Metropolis gives slightly better average energies than Importance. The Gibbs method is clearly the best sampling method, which we see by looking at the computed local average energies in the table for various hidden nodes. The best average energies for each sampling method is when we use two hidden nodes.

Like with the non-interacting system, the Metropolis is the fastest method and Importance is the slowest method of the three regarding CPU time. Also here, the CPU time for doing the energy computations increase when we use a higher number of hidden nodes. Using one hidden node gives the fastest CPU time, but gives a higher final energy for all sampling methods.

N_P	D	N_H	E_A	E_L^M	E_L^I	E_L^G
2	2	1	3.0	$3.140 \pm 1.3 \cdot 10^{-5}$	$3.144 \pm 4.0 \cdot 10^{-5}$	$3.131 \pm 5.3 \cdot 10^{-5}$
2	2	2	3.0	$3.081 \pm 7.7 \cdot 10^{-5}$	$3.081 \pm 1.2 \cdot 10^{-4}$	$3.062 \pm 5.2 \cdot 10^{-5}$
2	2	3	3.0	$3.082 \pm 1.9 \cdot 10^{-4}$	$3.084 \pm 3.3 \cdot 10^{-5}$	$3.066 \pm 2.8 \cdot 10^{-4}$
2	2	4	3.0	$3.084 \pm 8.7 \cdot 10^{-5}$	$3.086 \pm 3.5 \cdot 10^{-5}$	$3.067 \pm 4.2 \cdot 10^{-5}$

Table 4: Table for the final local energy of interacting electrons with the three sampling methods Metropolis (E_L^M), Importance (E_L^I) and Gibbs (E_L^G) with blocking errors, and the analytical ground state energy (E_A). The Gibbs sampling method provides the best average energies.

6 Conclusion

In this project we have used a restricted Boltzmann machine and neural network with a variational Monte Carlo method to compute the ground state energy of interacting and non-interacting systems consisting of electrons in a 2D isotropic harmonic oscillator potential. These ground state energies have been computed both analytically and numerically to check how well the RBM neural network can reproduce the energy of the system. To do this we have used the standard brute-force Metropolis sampling, Importance sampling and Gibbs sampling methods. To optimize the system we also used a stochastic gradient descent method with a constant learning rate, and used blocking for a statistical analysis of the energy errors. With these methods we looked at different choices of parameters for each sampling method to see how they affected our computation of the ground state energies.

In the non-interacting case we got energies very close to the analytically calculated ground state energies for all the three sampling methods. Here the Gibbs sampling gave the best solutions with the smallest errors. The Metropolis method was the fastest method in CPU time, but it also gave slightly worse solutions than the other two sampling methods.

For the interacting case we got energies a little higher than the analytically calculated energies. For all three sampling methods, we got reasonably good energies close to the analytical energies. The Metropolis was also the fastest method in CPU time here, and gave energies a little better than the Importance method, which where the slowest method again. As for non-interacting electrons, the Gibbs sampling method gave the best local energies closest to the analytical ground state energies.

From our analysis in this project, we can conclude that the Gibbs sampling method seems to be the best sampling method for this RBM NN system with electrons.

For future work, we could try to use other optimization methods like ADAM. We could also do a steepest descent method on some of the parameters we have used to make them adapt to make a better fit to the evolving behavior of the system. To get faster CPU times we could implement parallelization by using MPI. The system we have looked at in this project did not include Pauli's principle. So this is also something we could implement in future work, and eventually extend our system with more particles with higher dimensions.

Appendix:

A Kinetic Energy Derivation

Here we will derive the kinetic energy part of the local energy expression in equation 13. We start by rewriting the kinetic part:

$$\frac{1}{\Psi}\nabla^2\Psi = \nabla^2 \ln\Psi + (\nabla \ln\Psi)^2 \tag{40}$$

Then we find the logarithm of the wave function:

$$\ln\Psi = -\sum_{i=1}^{M}\frac{(X_i-a_i)^2}{2\sigma^2} + \sum_{j}^{N}\ln\left(1+e^{b_j+\Sigma_i^M \frac{X_i w_{ij}}{\sigma^2}}\right) \tag{41}$$

Than we take the gradient with respect to X_k:

$$\begin{aligned}\frac{\partial}{\partial X_k}\ln\Psi &= -\frac{X_k-a_k}{\sigma^2} + \sum_{j}^{N}\frac{e^{b_j+\Sigma_i^M \frac{X_i w_{ij}}{\sigma^2}}}{1+e^{b_j+\Sigma_i^M \frac{X_i w_{ij}}{\sigma^2}}}\cdot\frac{w_{kj}}{\sigma^2}\\ &= -\frac{X_k-a_k}{\sigma^2} + \sum_{j}^{N}\frac{1}{1+e^{-b_j-\Sigma_i^M \frac{X_i w_{ij}}{\sigma^2}}}\cdot\frac{w_{kj}}{\sigma^2}\\ &= -\frac{X_k-a_k}{\sigma^2} + \sum_{j}^{N}\Sigma(u_j)\cdot\frac{w_{kj}}{\sigma^2}\end{aligned} \tag{42}$$

The denominator in the fraction above has a famous name called the logistic sigmoid function. For simplicity we will use Σ for the sigmoid function as

$$\Sigma(u_j) = \frac{1}{1+e^{u_j}} \quad \text{with} \quad u_j = -b_j - \sum_{i}^{M}\frac{X_i w_{ij}}{\sigma^2}. \tag{43}$$

The second derivative is then:

$$\frac{\partial^2}{\partial X_k^2}\ln\Psi = -\frac{1}{\sigma^2} + \sum_{j}^{N}\Sigma(u_j)\cdot\Sigma(-u_j)\cdot\frac{w_{kj}^2}{\sigma^4} \tag{44}$$

The full kinetic energy part in equation 40 is then:

$$\begin{aligned}\frac{1}{\Psi}\nabla^2\Psi = &-\frac{M}{\sigma^2} + \sum_{i}^{M}\sum_{j}^{N}\Sigma(u_j)\cdot\Sigma(-u_j)\cdot\frac{w_{ij}^2}{\sigma^4} + \sum_{i}^{M}\frac{X_i-a_i}{\sigma^2}\\ &-2\sum_{i}^{M}\frac{X_i-a_i}{\sigma^2}\sum_{j}^{N}\Sigma(u_j)\cdot\frac{w_{ij}}{\sigma^2} + \sum_{i}^{M}\sum_{j}^{N}(\Sigma(u_j))^2\cdot\frac{w_{ij}^2}{\sigma^4}\end{aligned}$$

B Gibbs Kinetic Energy Derivation

With the new representation of the wave function as $\Psi_G(\mathbf{X}) = \sqrt{F_{rbm}(\mathbf{X})}$, we get that $\ln(\Psi_G) = \frac{1}{2}\ln(\Psi)$. This means that the kinetic energy part of the local energy becomes:

$$\begin{aligned}
\frac{1}{\Psi_G}\nabla^2\Psi_G &= \nabla^2 \ln \Psi_G + (\nabla \ln \Psi_G)^2 = \frac{1}{2}\nabla^2 \ln \Psi + \frac{1}{4}(\nabla \ln \Psi)^2 \\
&= -\frac{M}{2\sigma^2} + \frac{1}{2}\sum_i^M \sum_j^N \Sigma(u_j)\cdot\Sigma(-u_j)\cdot\frac{w_{ij}^2}{\sigma^4} + \frac{1}{4}\sum_i^M \frac{X_i - a_i}{\sigma^2} \\
&\quad -\frac{1}{2}\sum_i^M \frac{X_i - a_i}{\sigma^2}\sum_j^N \Sigma(u_j)\cdot\frac{w_{ij}}{\sigma^2} + \frac{1}{4}\sum_i^M\sum_j^N (\Sigma(u_j))^2\cdot\frac{w_{ij}^2}{\sigma^4}
\end{aligned}$$

C Analytic Derivation of the Local Energy

* 1 particle in 1D:
To derive the analytic expressions for the local energies we use the harmonic oscillator wave function in equation 16. For the simplest case with one particle in 1D, the eigenvalues (ϵ) of the Hamiltonian are dependent on the principal quantum number n_x in natural units as

$$\epsilon_{n_x} = \omega(n_x + \frac{1}{2}).$$

The local energy then becomes;

$$E_{n_x} = \frac{1}{\phi_{n_x}} H\phi_{n_x} = \epsilon_{n_x} = \omega(n_x + \frac{1}{2})$$

In the lowest-lying state $n_x = 0 \quad \Rightarrow \quad E_0 = \frac{\omega}{2}$.

* 1 particle in 2D:
For one particle in 2D we get

$$H\phi_{n_x,n_y} = H\phi_{n_x}\cdot\phi_{n_y} + H\phi_{n_y}\cdot\phi_{n_x} = \epsilon_{n_x}\phi_{n_x,n_y} + \epsilon_{n_y}\phi_{n_x,n_y}.$$

The local energy simply becomes:

$$E_{n_x,n_y} = \frac{1}{\phi_{n_x,n_y}} H\phi_{n_x,n_y} = \epsilon_{n_x} + \epsilon_{n_y} = \omega(n_x + n_y + 1)$$

In the lowest-lying state $n_x = n_y = 0 \quad \Rightarrow \quad E_{00} = \omega$.

* 2 particles in 2D:
For two electrons in 2D, the wave function can be split into two single particle wave function parts:

$$\begin{aligned}
\phi(\mathbf{r}_1,\mathbf{r}_2) &= \phi_1(\mathbf{r}_1)\cdot\phi_2(\mathbf{r}_2) = \phi_{1n_{x_1},n_{y_1}}\cdot\phi_{2n_{x_2},n_{y_2}} \\
\rightarrow H\phi(\mathbf{r}_1,\mathbf{r}_2) &= H\phi_1\cdot\phi_2 + H\phi_2\cdot\phi_1 = \epsilon_{n_{x_1},n_{y_1}}\phi_{1,2} + \epsilon_{n_{x_2},n_{y_2}}\phi_{1,2}
\end{aligned}$$

The local energy becomes:

$$E_{n_x,n_y} = \frac{1}{\phi(\mathbf{r}_1,\mathbf{r}_2)} H\phi(\mathbf{r}_1,\mathbf{r}_2) = \epsilon_{n_{x_1},n_{y_1}} + \epsilon_{n_{x_2},n_{y_2}} = \omega(n_{x_1} + n_{x_2} + n_{y_1} + n_{y_2} + 2)$$

In the lowest-lying state $n_{x_1} = n_{x_2} = n_{y_1} = n_{y_2} = 0 \quad \Rightarrow \quad E_{00} = 2\omega$.

D Quantum Force Derivation

For the Importance sampling we need to derive the quantum force in equation 32 analytically. Start be rewriting:

$$\frac{\nabla\Psi_T}{\Psi_T} = \nabla_k \ln \Psi_T$$

This gradient we have already calculated in Appendix A. This gives the quantum force as:

$$F_k = -2\frac{X_k - a_k}{\sigma^2} + 2\sum_j^N \Sigma(u_j) \cdot \frac{w_{kj}}{\sigma^2} \tag{45}$$

The quantum force is only used in Importance sampling, but can be altered for other wave functions (like the one in Gibbs sampling) as well.

References

[1] GitHub repository:. https://github.com/krilangs/FYS4411/tree/master/Project2.

[2] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[3] Henrik Flyvbjerg and Henrik Gordon Petersen. Error estimates on averages of correlated data. *The Journal of Chemical Physics*, 91(1):461–466, 1989.

[4] Marius Jonsson. Standard error estimation by an automated blocking method. *Physical Review E*, 98(4):043304, 2018.

[5] M Taut. Two electrons in an external oscillator potential: Particular analytic solutions of a Coulomb correlation problem. *Physical Review A*, 48(5):3561, 1993.