The Restricted Boltzmann Machine Applied to Quantum Many Body Systems

Philip K. Sørli Niane
Department of Physics, University of Oslo, Norway
Github address: Link here

May 31, 2021

Abstract

In this article, a quantum mechanical system consisting of fermions trapped in a spherical harmonic oscillator were simulated using the restricted Boltzmann machine, with a neural network quantum state (NQS) representing the wave function. The optimal parameters in the NQS were found using the stochastic gradient decent method. The ground state energies were computed with interacting and non interacting particles. In the non interacting case, the energy found using both the Metropolis algorithm, Metropolis-Hastings algorithm and Gibbs sampling as sampling methods equaled the exact energy. The resulting energies where found to be 0.5 *a.u.* per particle in one dimension, 1.0 *a.u.* per particle in two dimensions and 1.5 *a.u.* per particle in three dimensions, everyone with 2 hidden nodes. The interacting case on the other hand resulted in the following ground state energies when having 2 particles in 2 dimensions: x.xx *a.u.* using Metropolis Algorithm, x.xx *a.u.* using Metropolis-Hastings and x.xx *a.u.* using gibbs sampling. Compared to the analytical solution presented in [1] dissimilarity was x.x%, x.x% and x.x% using the Metropolis Algorithm, Metropolis-Hastings and Gibbs sampling repsectively with 2 particles in 2 dimensions in the harmonic oscillator trap.

Contents

1 Introduction

Simulating quantum mechanical systems and computing ground state energies are quiet popular quests within numerical computing and many body quantum mechanics. In recent past machine learning have also been finding its way into the world of quantum mechanical system simulations. In this article there will be explored how one can compute ground state energies in quantum mechanical many body systems using deep learning.

The machine learning method used in this article will be the restricted Boltzmann machine (RBM), and the way the RBM is used is by representing the wave function as a neural network, as presented in a quiet recent scientific paper [2]. When the wave function is represented this way, the wave function is called a neural network quantum state(NQS).

The quantum mechanical system will consist of up to two particles which can represent e.g. electrons or bosons in a variety of dimensions. The particles will be confined by a harmonic oscillator trap. The particles will in first place not have the possibility to interact with each other, but will gain the skill to interact after the non interacting systems have been tested.

A number of different variables will be tested and explored regarding the NQS, the simulated system and optimizing process. The method used to optimize the NQS will be the stochastic gradient decent method to ensure stable optimization of the NQS.

The sampling methods that will be used is the Metropolis algorithm, the Metropolis-Hastings algorithm and Gibbs sampling which will be used in addition to the same methods with different step sizes(for the Metropolis- and Metropolis-Hastings algorithm), to see how different sampling methods can affect the ground state energies.

The article will start off with some theory about the important elements worth mentioning e.g. sampling methods and the quantum mechanical system, which is then followed by the details of the implementation. Then comes the analysis part unveiling the results and discussion of them, naturally followed by a conclusion summing up the article.

2 Theory

2.1 The Variational Method

Most many-body systems in quantum mechanics have the property of being highly advanced and complex. The more complex the system is, the less is the probability of having an analytical solution present. The road from a simple quantum mechanical system with an analytical solution present, to a system too complex for an analytical solution is very small. For example a hydrogen atom has one electron and can be computed analytically quiet easily, but when adding

an electron and trying to compute the energy of a helium atom it becomes impossible to calculate an analytical solution due to the complexness of the system. This is why the variational method often is used to find a highly accurate estimate.

The variational method is based on the fact that the expectation value of the Hamiltonian H is an upper bound for the ground state energy E_{gs}, for any wave function chosen. The proof can be seen in [3]

$$E_{gs} \leq \frac{\langle \Psi_T | H | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} \tag{1}$$

Where Ψ_T is the trial wave function chosen. The trial wave function can be any wave function but it is usually chosen according to the quantum mechanical system, due to similar systems usually having quiet similar wave functions. To ensure lowest possible energy, the trial wave function contains some parameter(s) that is determined by deciding which value of the parameter(s) is giving the lowest energy, knowing that this energy is just an upper bound for the ground state energy [4].

2.2 The Quantum Mechanical System

2.2.1 The Hamiltonian

The Hamiltonian operator of the system is given by the following:

$$\hat{H} = \hat{H}_0 + \hat{H}_1 \tag{2}$$

Where $\hat{H}_0$ is the unpertubed Hamiltonian, and $\hat{H}_1$ is the pertubed Hamiltonian.

The unpertubed Hamiltonian includes a standard harmonic oscillator part:

$$\hat{H}_0 = \sum_i^N \left(-\frac{1}{2} \bigtriangledown_i^2 + \frac{1}{2} \omega^2 r_i^2 \right) \tag{3}$$

Where N is the number of particles in the system, and ω is the trap frequency. Natural units are used. The pertubed repulsive part of the Hamiltonian is given as the following:

$$\hat{H}_1 = \sum_{i<j}^N \frac{1}{r_{ij}} \tag{4}$$

Where r_{ij} is the distance between the particles. r_{ij} is given by $r_{ij} = |r_i - r_j|$, with r_p given as $r_p = \sqrt{r_{px}^2 - r_{py}^2}$.

Which let the Hamiltonian of the whole system be written as follows:

$$\hat{H} = \sum_{i}^{N} \left(-\frac{1}{2} \bigtriangledown_i^2 + \frac{1}{2}\omega^2 r_i^2 \right) + \sum_{i<j}^{N} \frac{1}{r_{ij}} \tag{5}$$

2.2.2 Analytical Solution for the Non Interacting case

When computing the ground state energy of the unpertubed system, the computations is quiet straight forward. The wave function for the ground state of the two-electron system is given by [5]:

$$\Phi(r_1, r_2) = C \exp\left(-\omega(r_1^2 + r_2^2)/2\right) \tag{6}$$

With C being the normalization constant. When having the unpertubed Hamiltonian act on the wave function the resulting ground state energy comes out as:

$$E = \frac{ND\omega}{2} \tag{7}$$

Where N is the number of particles and D is the number of dimensions.

Switching the labels of the particles in equation (6), $\Phi(r_1, r_2) = \Phi(r_2, r_1)$ leaves the wave function unchanged, means that the spacial part of the wave function is symmetric. Then adding the fact that the electrons in the system are fermions, and both in the ground state spatial wise, means that the spins need to be opposite due to pauli exclusion principle stating that two fermions can not have the same spacial positions while having the same spin direction. This means that the electrons in the system is takes the singlet state. Which means that the total spin of the wave function in equation (6) is 0.

2.3 The Restricted Boltzmann Machine

The restricted Boltzmann machine is a machine learning methods that can be seen as a stochastic recurrent neural network, which is able to learn probability distributions over a set of inputs. Boltzmann machines consists of visible and hidden nodes, and often biases. The difference between a regular Boltzmann machine and a restricted Boltzmann machine is that no nodes in the same layer is connected to one another in the restricted one, while in a regular Boltzmann machine nodes usually are connected to all other nodes[6]. The architecture of a restricted Boltzmann machine can be seen in figure 1.

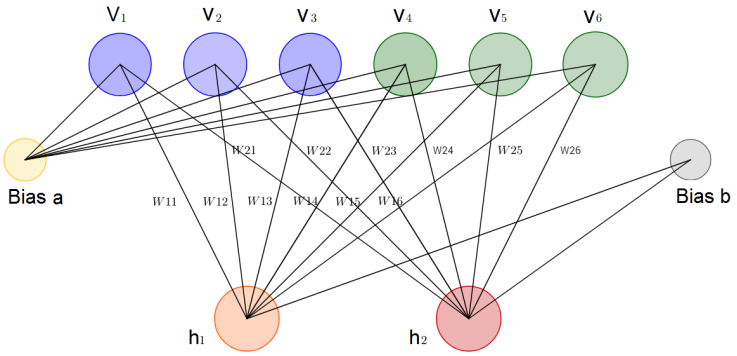


Figure 1: Architecture of a restricted Boltzmann machine, having the visible nodes connected to every hidden node and the visible bias, while the hidden nodes are connected to all the visible nodes and the hidden bias. The W's in the image is values of the weight matrix which decides how strong the connection between each visible and hidden node is. Image from [7]

2.4 Neural Network Quantum State as the Wave Function

Instead of working with training data as most machine learning methos do, in this article the RBM is rather using the fact that minimizing the energy by optimizing the weights and biases of the NQS is giving the best solution. This is called reinforcement learning.

The wave function represented by the probability distribution the RBM will be modelling is the following for the quantum mechanical system of the article:

$$\Psi = F_{rbm}(\mathbf{X}, \mathbf{H}) = \frac{1}{Z} e^{-\frac{1}{T_0} E(\mathbf{X}, \mathbf{H})} \tag{8}$$

Where X is a vector of visible nodes representing the positions of the particles and dimensions, H is a vector of the hidden nodes. In this article the RBM that will be used is a Gaussian-Binary type, meaning that the visible nodes has continous values, while the hidden nodes are restricted to be either 0 or 1. T_0 is set to 1, Z is the partition function/normalization constant which is given as follows:

$$Z = \int \int \frac{1}{Z} e^{-\frac{1}{T_0} E(\mathbf{x}, \mathbf{h})} d\mathbf{x} d\mathbf{h} \tag{9}$$

$E(\boldsymbol{X}, \boldsymbol{H})$ is the joint probability distribution defined:

$$E(\mathbf{X}, \mathbf{H}) = \frac{||\mathbf{X} - \mathbf{a}||^2}{2\sigma^2} - \mathbf{b}^T\mathbf{H} - \frac{\mathbf{X}^T\mathbf{W}\mathbf{H}}{\sigma^2} \tag{10}$$

Where $\boldsymbol{a}$ is the biases connected to the visible nodes with the same length as $\boldsymbol{X}$. $\boldsymbol{b}$ is the biases connected to the hidden nodes with the same length as $\boldsymbol{H}$. $\boldsymbol{W}$ is a matrix containing the weights deciding how strong the connections between the hidden and visible nodes are.

Writing the marginal probability as

$$F_{rbm}(\mathbf{X}) = \sum_{\mathbf{h}} F_{rbm}(\mathbf{X}, \mathbf{h}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{X},\mathbf{h})} \tag{11}$$

Then connecting the different elements gives the wave function that is going to be used:

$$\Psi(\mathbf{X}) = \frac{1}{Z} e^{-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2}} \prod_j^N (1 + e^{b_j + \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}) \tag{12}$$

2.4.1 The Systems Local Energy

The local energy of the system can be computed by combining equation (1), the Hamiltonian in equation (5) and the wave function from equation (12) which gives the following expression:

$$E_L = \frac{1}{\Psi} H\Psi = \sum_{i=1}^{N} \left(-\frac{1}{2\Psi}\nabla_i^2\Psi + \frac{1}{2}\omega^2 r_i^2 \right) + \sum_{i<j}^{N} \frac{1}{r_{\text{ik}}} \tag{13}$$

After inserting the wave function the final analytical expression for the local energy can be written as follows:

$$E_L = -\frac{1}{2}\sum_{i=1}^{M}\Bigg[\left(\frac{a_i - X_i}{\sigma^2} + \sum_{j=1}^{N} w_{ij}\delta(\delta_j^{input}) \right)^2 \tag{14}$$

$$- \frac{1}{\sigma^2} + \sum_{j=1}^{N} \frac{w_{ij}^2}{\sigma^4}\delta(\delta_j^{input})\delta(-\delta_j^{input}) - \omega^2 r_i^2 \Bigg] + \sum_{i<j} \frac{1}{r_{ij}} \tag{15}$$

Where M is the number of visible nodes, and N is the unmber of hidden nodes. $\delta(\cdot)$ is a sigmoid function, and δ_j^{input} is defines as follows:

$$\delta_j^{input} = b_j + \sum_{i=1}^{M} \frac{X_i w_{ij}}{\sigma^2} \tag{16}$$

Where the derivation can be seen in the appendix.

2.5 Stochastic Gradient Decent

The procedure of the simulations will have some aspects of similarities to the Variational Monte Carlo method, which has a more detailed description here [4]. In this article there will be a bigger focus on the optimization process, to ensure that the wave function results in the lowest energy possible. The method that will be used to minimize the energy is named the stochastic gradient decent method (SGD), which is a quite stable way of finding optimized parameters to a function. The way the SGD will optimize the parameters of the wave function is that for each RBM cycle the SGD will use the parameters used in the last cycle to optimize them and compute new parameters. These parameters is then used in the next RBM cycle which is then optimized again which leads to new parameters. This goes on and on resulting in better and better results.

The SGD is quiet straight forward. An initial parameter, also known as a guess, is needed to start of the method and compute the next parameter. The SGD is based on the fact that almost all functions that is going to be minimizes has the possibility to be written as a sum over the data points. the following way:

$$C(\beta) = \sum_{i=1}^{n} c_i(\boldsymbol{x_i}, \beta) \tag{17}$$

Which means that the gradient can be computed as a sum over i-th gradients the following way:

$$\nabla_\beta C(\beta) = \sum_{i}^{n} \nabla_\beta c_i(\boldsymbol{x_i}, \beta) \tag{18}$$

The SGD step can be computed by the following formula, for a more detailed explanation of the method, feel free to have a look here [8].

$$\beta_{j+1} = \beta_j - \gamma \sum_{i \in B_k}^{n} \nabla_\beta c_i(\boldsymbol{x_i}, \beta) \tag{19}$$

Where γ is the learning rate, k is picked at random (explaining the 'stochastic' part of the name).

To compute the derivative of the local energy the following formula is used:

$$\frac{\partial \langle E_L \rangle}{\partial \alpha_i} = 2(\langle E_L \frac{1}{\Psi} \frac{\partial \Psi}{\partial \alpha_i} \rangle - \langle E_L \rangle \langle \frac{1}{\Psi} \frac{\partial \Psi}{\partial \alpha_i} \rangle) \tag{20}$$

Where the parameter $\alpha_i = a_1, ..., a_M, b_1, ..., b_N, w_{11}, ..., w_{MN}$, being the free parameters. The derivatives of the wave function with respect to the different parameters can be seen in the appendix.

2.6 Sampling methods

The sampling methods that are going to be used are the Metropolis algorithm, also called brute force sampling, Metropolis-Hastings algorithm also called importance sampling, and Gibbs sampling.

2.6.1 Metropolis Algorithm

The Metropolis algorithm is used when proposing and carrying on a step in a Monte Carlo simulation. When choosing a step that a bit of randomness is introduced, that is because the steps are chosen randomly by the following equation:

$$r^{new} = r^{old} + \in [0, 1] \cdot dr \tag{21}$$

Where dr is a set step length. The step length is multiplied by a random number from the interval [0,1].

The algorithm then checks if the results are accepted or rejected by computing the ratio of the new and old wave function and comparing it to a random variable in the same interval as the previous step.

Having the steps accepted by the Metropolis algorithm is securing that the simulation and sampling is mainly focused around the largest part of the probability density function. For a more detailed description of the Metropois algorithm feel free to have a look at [4].

2.6.2 Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm, also known as importance sampling is an algorithm that can be used instead of the Metropolis algorithm for sampling and moving particles. The wave function is contribution to the movement of the particle by a so called drift force F, while the Fokker-Planck equation and the Langevin equation is used to produce the particle route.

The drift force F is expressed as the following:

$$F = \frac{2\nabla \Psi_T}{\Psi_T}. \tag{22}$$

Where the derivative of the wave function can be seen in the appendix.

Also in the Metropolis-Hastings algorithm a step is proposed and checked for acceptance, by computing the ratio and comparing to some random variable. in the interval of $[0, 1]$. The reason for using the Metropolis-Hastings algorithm instead of the Metropolis algorithm is that more steps would be accepted due to the drift force leading forward in the PDF. For a more descriptive explanation including the general formulas, feel free to have a look at [4].

2.6.3 Gibbs sampling

Gibbs sampling is a method that can be used as an alternative to the Metropolis- and Metropolis-Hastings algorithm. Gibbs sampling is a great choice when dealing with multivariate probability function. By using the following distribution as the wave function, it is possible to model the positional probability distribution:

$$\Psi_{Gibbs} = \sqrt{F_{rbm(\boldsymbol{X})}} \tag{23}$$

Which means that the following formula gives the positional probability distribution:

$$|\Psi_{Gibbs}|^2 = F_{rbm(\boldsymbol{X})} \tag{24}$$

The wave function in both formulas can be seen in equation (8).

In this project Gibbs sampling is sampling from the joint probability of the visible and hidden nodes. The samples are refreshed by setting the hidden nodes to binary values of 0 or 1 by the following formulas derived in [6]:

$$P(h_J = 1|\boldsymbol{X}) = \delta(\delta^{input}) \tag{25}$$

and

$$P(h_J = 0|\boldsymbol{X}) = \delta(-\delta^{input}) \tag{26}$$

Where $\delta(\cdot)$ is a sigmoid function and δ^{input} is defined in equation (16).

By using the sampled values of the visible nodes, to construct the probability density the positional distribution is received. Efficiency-wise Gibbs sampling is better compared to the Metropolis and the Metropolis Hastings algorith due to the fact that the acceptance rate is 100% because the nodes are updated based on the sampled values rather than accepted and rejected, which ensures that Gibbs sampling is using all its computational power on accepted moves.

2.7 The Blocking Method

Statistical errors almost always occur when using dealing with numerical simulations. Especially when combined with some sort of randomness. There are multiple ways of measuring errors and deviance. The one used to measure the statistical errors in this article will be a method called the blocking technique, which is an excellent choice when the variables are correlated.

The blocking technique is based on dividing multiple samples of correlated data into a number of blocks. The amount of samples need to be able to write as number on the form 2^K, where K is an integer number.

By calculating the average within each block, the correlation between the samples will fade away, and the standard deviation can be calculated as uncorrelated samples. More on the blocking method can be seen here [9].

3 Method

3.1 The Program Flow

The flow of the program on the restricted Boltzmann machine goes as follows. The implementation starts of by determining the desired parameters, and the preferred methods Metropolis algorithm, Metropolis-Hastings algorithm, Gibbs sampling. The restricted Boltzmann machine simulation then starts of by initializing the values of the nodes, biases and weight matrix, where the visible nodes represents the positions of the particles, the distribution of the spreading is chosen beforehand, e.g. uniform or normal.

After everything is initialized, the Monte Carlo simulation is performed as done in [1]. After the Monte Carlo simulation is done, the current data of the system is sent into a stochastic gradient decent function, which optimizes the weight matrix and the biases. This equals one cycle of the restricted Boltzmann machine. The cycles carries on, running the Monte Carlo simulations wiht the optimized parameters, which is optimised even further after each cycle, until the SGD tol is reached or the cycles runs out. Then one final simulation is done with the optimized values, computing the energy of the system.

3.2 Implementation of the Stochastic Gradient Decent algorithm

Implementation of the gradient decent is quiet straight forward. Using equation (19) gives the new values for the visible biases, hidden biases and the weights. The SGD algorithm computes optimized parameters every cycle, where each iteration gives a more optimized energy expression, with optimized parameters. The length of the simulation can be long or short depending on how hard it is for the gradient decent to converge towards optimal parameters. The optimization method runs the simulation until one of two options are ticked. Either an

amount of 100 cycles is reached, or the derivative of the parameters is less than a tol of 10^{-5}.

3.3 The Blocking Method

The blocking method is utilized by performing the algorithm with the sampled energies accumulated during the simulation. The statistical handling of the error will be computed by using the blocking method written by Marius Jonsson [9].

3.4 Parallelization of the System

Parallelization of the system is done by duplicating the current state of a core unto another core using the fork() command. The two cores then runs in parallel. This can be done for multiple cores resulting in multiple cores running in parallel. Assigning different parameters to each core which then will run in parallel ensures that the results can be accumulated even faster, when investigating the quantum mechanical system with different initial values and parameters.

4 Results

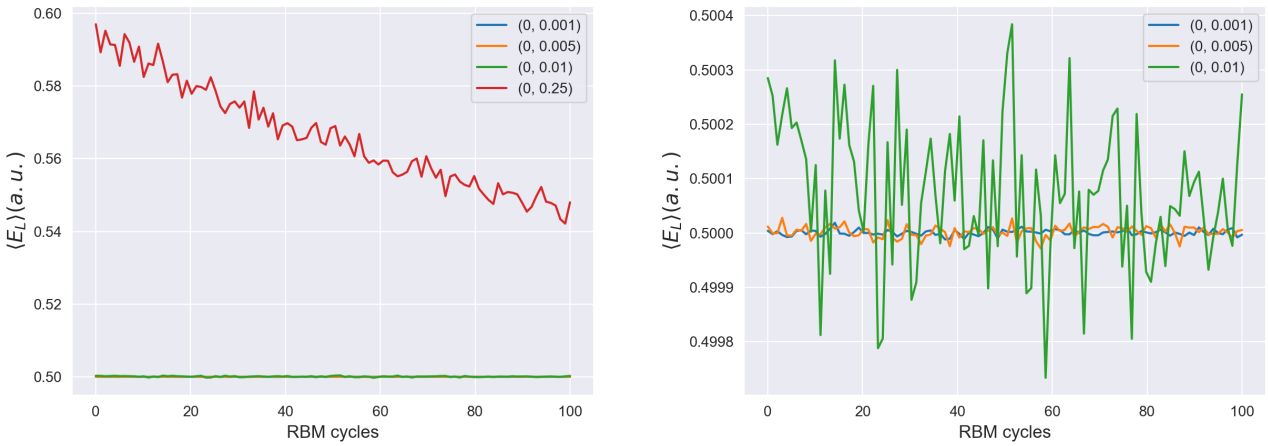
The number of metropolis steps used in all computations were chosen to be 2^20, due to the low number of particles and dimensions in the simulations.

4.1 Non Interacting System

In this subsection the results for non interacting systems will be unveiled.

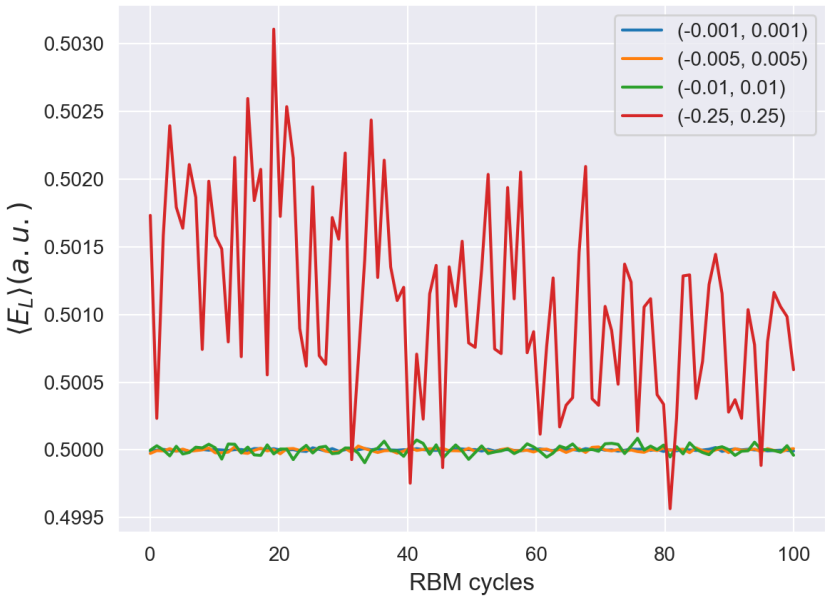
4.1.1 Choosing the distribution of particles

There are multiple possible ways of distributing the particles in the system. The most intuitive methods are either by a uniform distribution or a normal/-Gaussian distribution. To experiment with the distributions and initialisation of particles, the energy was plotted as a functions of multiple particle-initialisation for both a normal- and uniform distribution. The results for the normal distribution can be seen in figure 2, while the results for the uniform distribution can be seen in figure 3.



(a) (b)

Figure 2: Metropolis algorithm, Learning rate of 0.001, 1 particle, 1 dimension. a) Shows all, b) shows the most stable ones for a better view



(a)

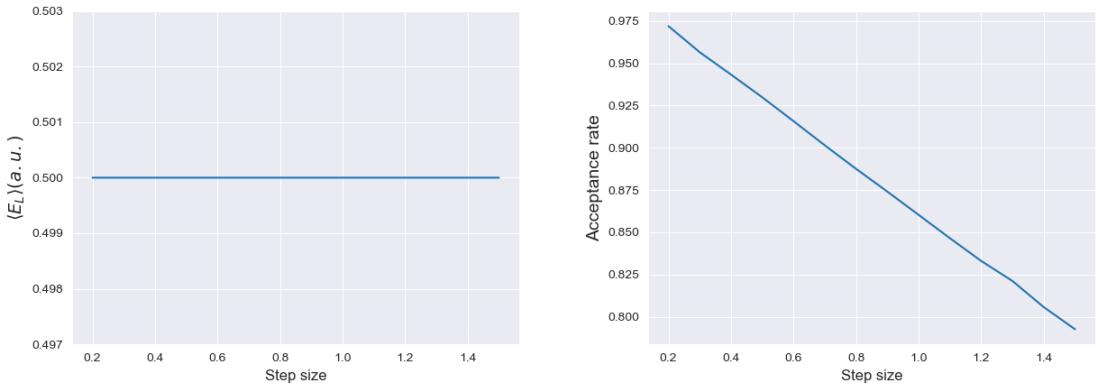
(b)

Figure 3: Metropolis algorithm, Learning rate of 0.001, 1 particle, 1 dimension. a) Shows all, b) shows the most stable ones for a better view

Normal distribution was chosen to be used for the rest of the simulations, due to the reasons given in section 5.1.

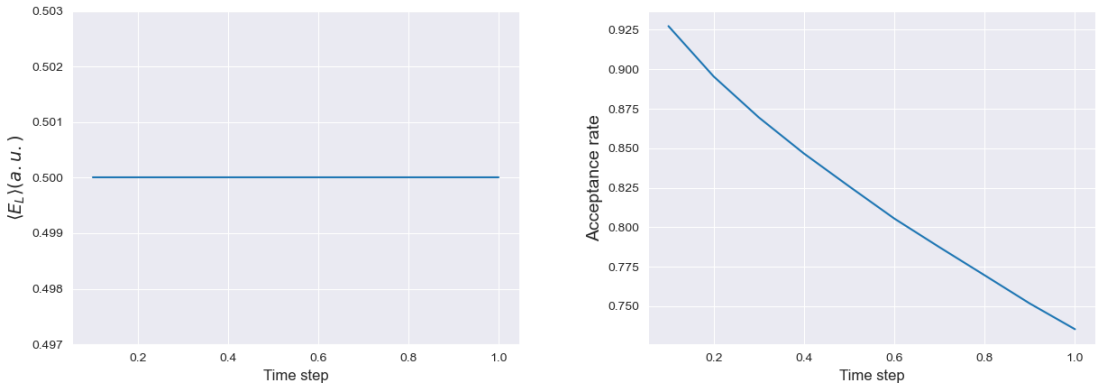
4.1.2 Choosing the Step Length and Time Step

Before setting of the more heavy calculations, the step length used in the Metropolis algorithm and the time step used in the Metropolis-Hastings algorithm was examined. Therefore the mean energies for different step lengths and time steps were plotted as a functions of the amount of Monte Carlo cycles/Metropolis steps, in addition the acceptance rate were also plotted as a function of step sizes and time steps. The results for the Metropolis algorithm can be seen in figure 4, while the results for the Metropolis-Hastings algorithm can be seen in 5.



(a) (b)

Figure 4: Learning rate of 0.35, 1 particle, 1 dimension, and sigma=1



(a) (b)

Figure 5: The mean energy plotted as a function of... Learning rate of 0.35, 1 particle, 1 dimension

The chosen step length for the Metropolis algorithm was decided to be set as 0.5 for the upcoming calculations, while the time step for the Metropolis-Hastings algorithm was decided to be set as 0.25 for the upcoming calculations. The reason for this can be seen in section **??**.

4.1.3 Choosing σ in the wave function

Figure 4 and figure 5 were computed with $\sigma = 1$ which gave results matching the analytically result quiet well. When using the Metropolis and Metropolis-Hastings algorithm σ was therefore kept as 1 throughout the article. Exploring σ when using gibbs sampling was done, and can be seen in figure 6.

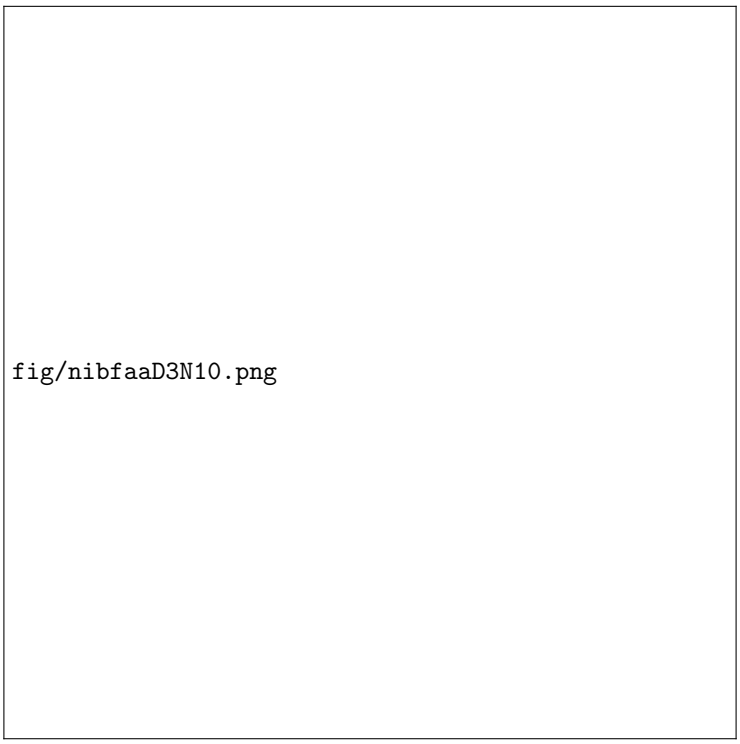


Figure 6: Gibbs sigma, plot for lr=0.001

The resulting σ was chosen to be 0.7 when using Gibbs sampling throughout the article.

CONTINUE FROM HERE!!

4.1.4 Investigating the learning rate and number of hidden nodes

By simulating the system for different amounts of hidden nodes in combination with different learning rates, the optimal learning rate and number of hidden nodes in the network can be found. In figure

By changing the variational parameter α by a set amount, it is easy to get an estimate of where the minimum is. Therefore the mean energy was calculated by varying α by 0.05 in the interval of $[0.3, 0.7]$. The results can be seen in figure **??** and figure **??**.

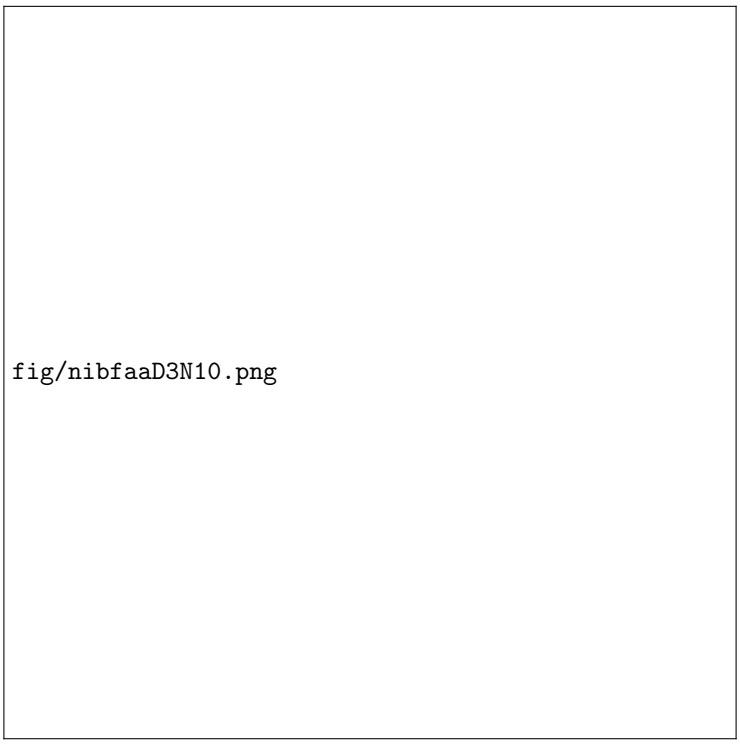


Figure 7: Heatmap of different learning rates and different amounts of hidden nodes using the Metropolis Algorithm with $2^{18} steps$

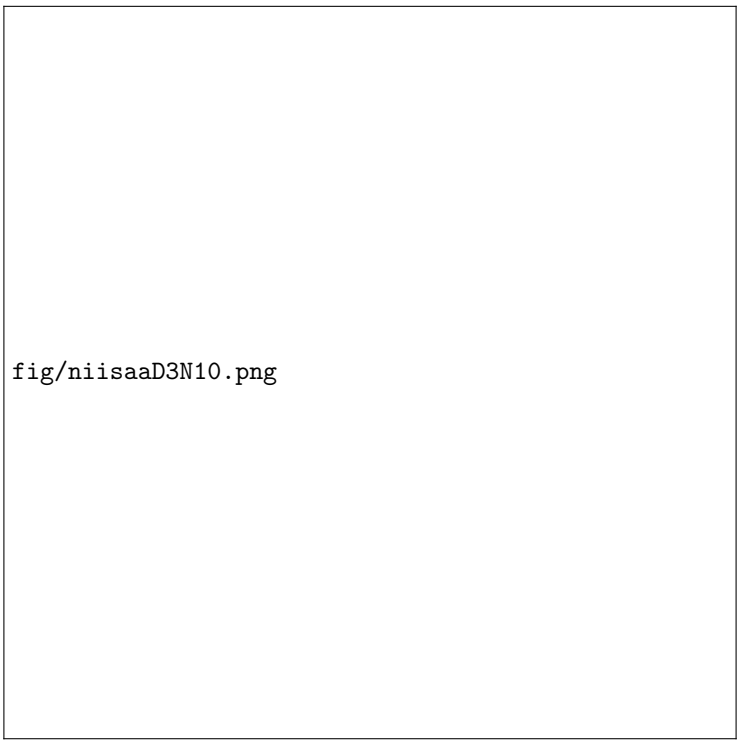


Figure 8: Heatmap of the error for simulations ran with different learning rates and number of hidden nodes using the Metropolis Algorithm with 2^{18} steps. The error is computed using the blocking method.

Both the Metropolis and the Metropolis-Hastings methods had an energy minimum at $\alpha = 0.5$.

4.1.5 Finding the Lowest Energy: Finding the Variational Parameter with Gradient Decent

Differing from varying the variational parameter α by a set amount, it is also possible to find the best variational parameter by using an optimization method. Therefore a simple gradient decent method was used and tested in search of the best variational parameter. The initial α was set to 0.3 in figure 9 while the initial α was set to 0.7 in figure 10.

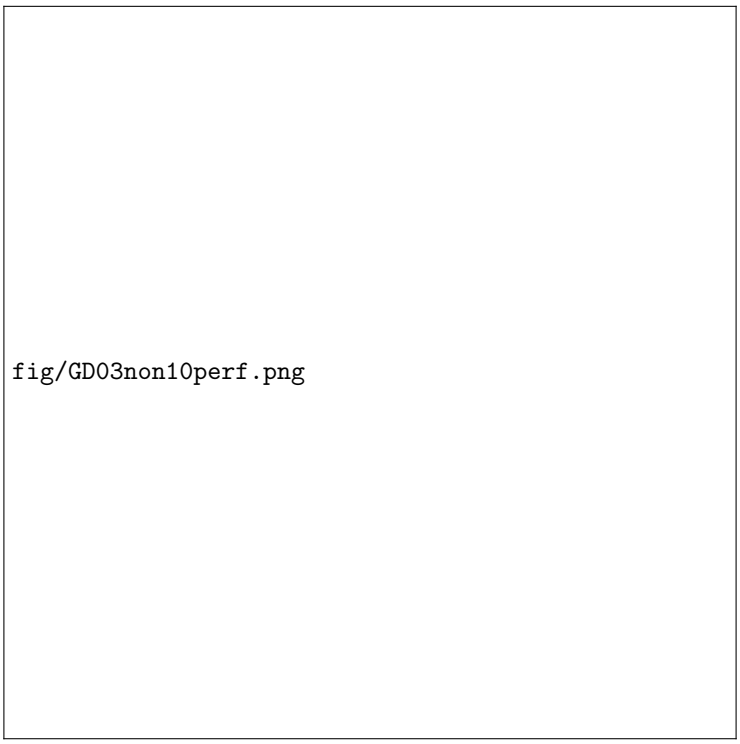


Figure 9: The mean energy plotted as a function of the variational parameter α, using the gradient decent method and the standard Metropolis algorithm with non interacting particles. Initial $\alpha = 0.3$, Metropolis steps per iteration $=10^5$, GD step size $\gamma = 0.005$, N=10, dimensions=3, and $\beta = 1$. Zoomed in plot of the last convergence part can be seen in the corner of the main plot, showing the method settling down at $\alpha = 0.5$.

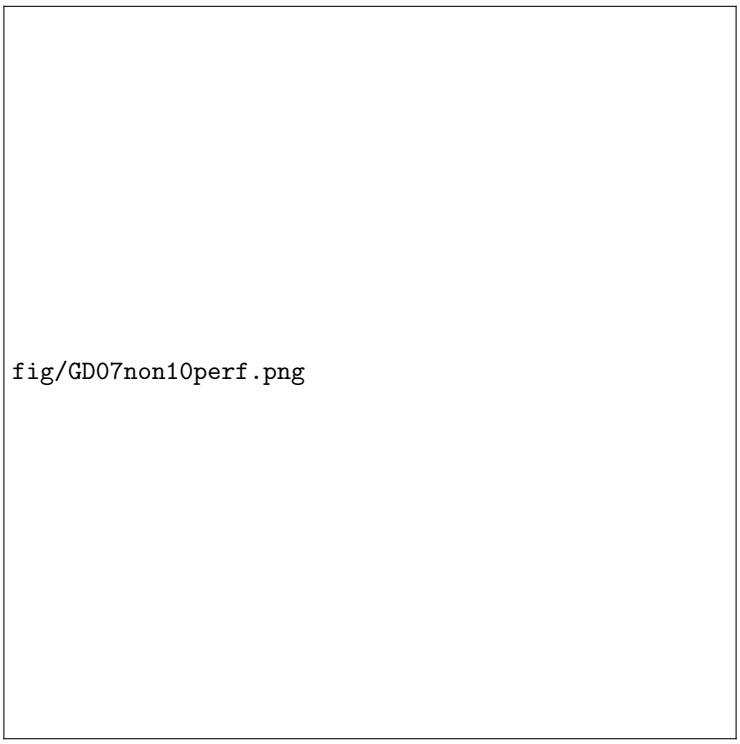


Figure 10: The mean energy plotted as a function of the variational parameter α, using the gradient decent method and the standard Metropolis algorithm with non interacting particles. Initial $\alpha = 0.7$, Metropolis steps per iteration=10^5, GD step size $\gamma = 0.005$, N=10, dimensions=3, and $\beta = 1$. Zoomed in plot of the last convergence part can be seen in the corner of the main plot, showing the method settling down at $\alpha = 0.5$.

In both cases, starting from each side of the optimal α, the values of alpha both converged to 0.5. The gradient decent was also tested for a larger amount of particles with N=100 to see how it performed. The results can be seen in figure 11.

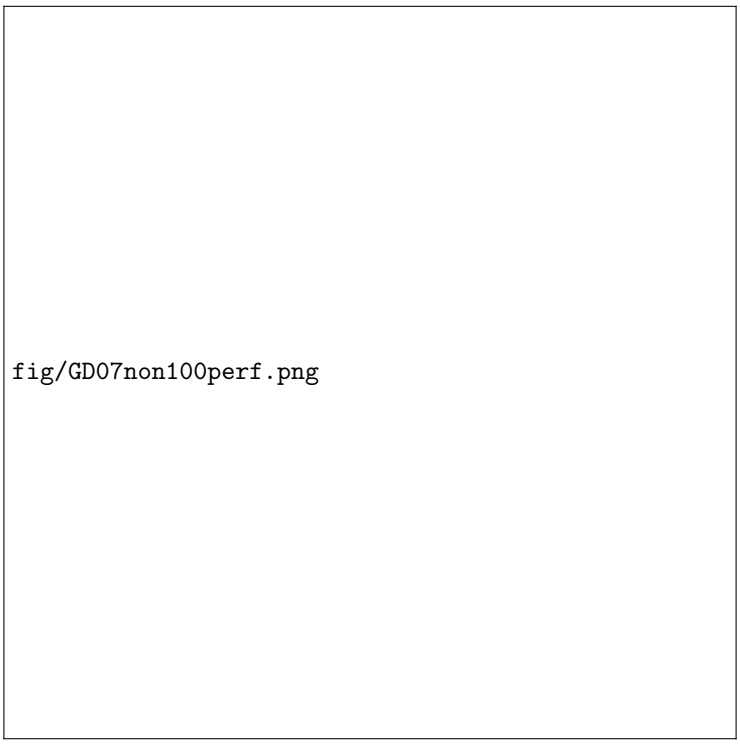


Figure 11: The mean energy plotted as a function of the variational parameter α, using the gradient decent method and the Metropolis algorithm with non interacting particles. Initial $\alpha = 0.7$, Metropolis steps per iteration=10^5, GD step size $\gamma = 0.005$, N=100, dimensions=3, and $\beta = 1$. Zoomed in plot of the last convergence part can be seen in the corner of the main plot, showing the method settling down at $\alpha = 0.5$.

It was a bit of a 'bumpy ride' on the road towards 0.5, due too some instability in the algorithm, but it ended up at 0.5 in all three cases so the results are favoured.

4.1.6 The Local Energies

The methods were tested for a variety of different amounts of particles and dimensions. The derivatives of the trial wavefunctions were also calculated by using analytical and numerical differentiation. The results by using the standard Metropolis algorithm can be seen in table 1, 2 and 3.

Particles	$E_E(\hbar\omega_{ho})$	$E_A(\hbar\omega_{ho})$	$E_N(\hbar\omega_{ho})$	μ_A	μ_N	CPU time$_A(s)$	CPU time$_N(s)$	$E_E/N(\hbar\omega_{ho})$
1	0.5	0.5	0.5	0	$1.3{\cdot}10^{-9}$	1.91	2.35	0.5
10	5	5	5	0	$1.3{\cdot}10^{-8}$	6.67	35.2	0.5
100	50	50	50	0	$1.7{\cdot}10^{-7}$	53.9	3397	0.5
500	250	250	250	0	$6.0{\cdot}10^{-6}$	257	60038	0.5

Table 1: Local energies in 1 dimension with N=1, 10, 100 and 500 particles, using the Metropolis algorithm, with non interacting particles. $\beta = 1$, Metropolis steps=2^{19}. The standard deviation μ was computed by using the blocking method. The undercored E, A and N stands for exact, analytical and numerical.

Particles	$E_E(\hbar\omega_{ho})$	$E_A(\hbar\omega_{ho})$	$E_N(\hbar\omega_{ho})$	μ_A	μ_N	CPU time$_A(s)$	CPU time$_N(s)$	$E_E/N(\hbar\omega_{ho})$
1	1	1	1	0	$1.8{\cdot}10^{-9}$	1.95	3.05	1
10	10	10	10	0	$1.7{\cdot}10^{-8}$	6.83	66.0	1
100	100	100	100	0	$1.2{\cdot}10^{-7}$	55.0	6525	1
500	500	500	500	0	$6.0{\cdot}10^{-6}$	263	257434	1

Table 2: Local energies in 2 dimension with N=1, 10, 100 and 500 particles, using the Metropolis algorithm, with non interacting particles. $\beta = 1$, Metropolis steps=2^{19}. The standard deviation μ was computed by using the blocking method. The undercored E, A and N stands for exact, analytical and numerical.

Particles	$E_E(\hbar\omega_{ho})$	$E_A(\hbar\omega_{ho})$	$E_N(\hbar\omega_{ho})$	μ_A	μ_N	CPU time$_A(s)$	CPU time$_N(s)$	$E_E/N(\hbar\omega_{ho})$
1	1.5	1.5	1.5	0	$1.9{\cdot}10^{-9}$	1.99	3.78	1.5
10	15	15	15	0	$1.5{\cdot}10^{-8}$	6.98	97.1	1.5
100	150	150	150	0	$1.3{\cdot}10^{-7}$	55.9	9607	1.5
500	750	750	750	0	$9.2{\cdot}10^{-7}$	269	357039	1.5

Table 3: Local energies in 3 dimension with N=1, 10, 100 and 500 particles, using the Metropolis algorithm, with non interacting particles. $\beta = 1$, Metropolis steps=2^{19}. The standard deviation μ was computed by using the blocking method. The undercored E, A and N stands for exact, analytical and numerical.

The methods were also tested and ran for different number of particles, and dimensions by using the Metropolis-Hastings algorithm. The results from the computations can be seen in table 4, 5 and 6.

Particles	$E_E(\hbar\omega_{ho})$	$E_A(\hbar\omega_{ho})$	$E_N(\hbar\omega_{ho})$	μ_A	μ_N	CPU time$_A(s)$	CPU time$_N(s)$	$E_E/N(\hbar\omega_{ho})$
1	0.5	0.5	0.5	0	$4.3{\cdot}10^{-10}$	1.90	2.75	0.5
10	5	5	5	0	$4.2{\cdot}10^{-9}$	6.59	40.0	0.5
100	50	50	50	0	$6.7{\cdot}10^{-8}$	54.8	3006	0.5
500	250	250	250	0	$9.2{\cdot}10^{-7}$	135	133992	0.5

Table 4: Local energies in 1 dimension with N=1, 10, 100 and 500 particles, using the Metropolis-Hasting algorithm, with non interacting particles. $\beta = 1$, Metropolis steps=2^{19}. The standard deviation μ was computed by using the blocking method. The undercored E, A and N stands for exact, analytical and numerical.

Particles	$E_E(\hbar\omega_{ho})$	$E_A(\hbar\omega_{ho})$	$E_N(\hbar\omega_{ho})$	μ_A	μ_N	CPU time$_A(s)$	CPU time$_N(s)$	$E_E/N(\hbar\omega_{ho})$
1	1	1	1	0	$1.7{\cdot}10^{-9}$	1.96	3.64	1
10	10	10	10	0	$1.2{\cdot}10^{-8}$	6.77	72.1	1
100	100	100	100	0	$9.4{\cdot}10^{-8}$	55.0	5093	1
500	500	500	500	0	$4.4{\cdot}10^{-7}$	137	257844	1

Table 5: Local energies in 2 dimension with N=1, 10, 100 and 500 particles, using the Metropolis-Hasting algorithm, with non interacting particles. $\beta = 1$, Metropolis steps=2^{19}. The standard deviation μ was computed by using the blocking method. The undercored E, A and N stands for exact, analytical and numerical.

Particles	$E_E(\hbar\omega_{ho})$	$E_A(\hbar\omega_{ho})$	$E_N(\hbar\omega_{ho})$	μ_A	μ_N	CPU time$_A(s)$	CPU time$_N(s)$	$E_E/N(\hbar\omega_{ho})$
1	1.5	1.5	1.5	0	$1.9{\cdot}10^{-9}$	2.00	4.56	1.5
10	15	15	15	0	$1.6{\cdot}10^{-8}$	6.97	104.4	1.5
100	150	150	150	0	$1.3{\cdot}10^{-7}$	28.8	6896	1.5
500	750	750	750	0	$5.6{\cdot}10^{-7}$	137	356536	1.5

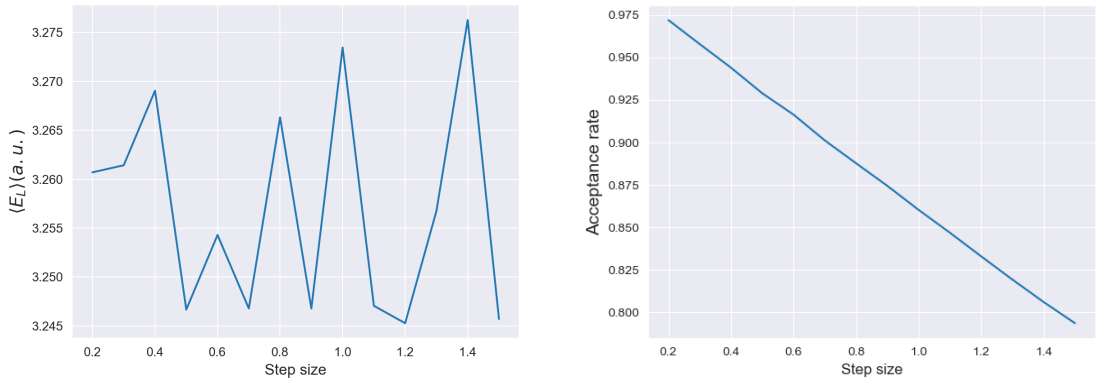
Table 6: Local energies in 3 dimension with N=1, 10, 100 and 500 particles, using the Metropolis-Hasting algorithm, with non interacting particles. $\beta = 1$, Metropolis steps=2^{19}. The standard deviation μ was computed by using the blocking method. The undercored E, A and N stands for exact, analytical and numerical.

4.2 Interacting Particles

Then it's time for the interacting cases. In this subsection the results for interacting particle systems will be unveiled.

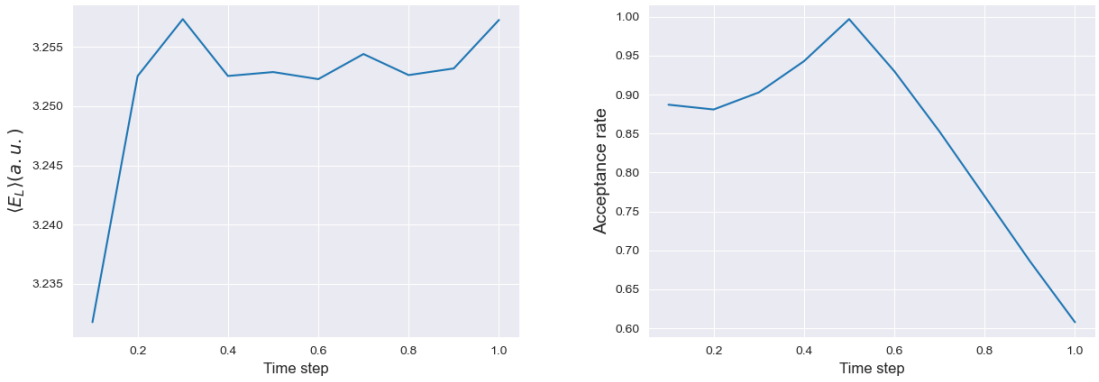
4.2.1 Choosing the Step Length and Time Step

Before setting of the more heavy calculations, the step length used in the Metropolis algorithm and the time step used in the Metropolis-Hastings algorithm was examined. Therefore the mean energies for different step lengths and time steps were plotted as a functions of the amount of Monte Carlo cycles/Metropolis steps, in addition the acceptance rate were also plotted as a function of step sizes and time steps. The results for the Metropolis algorithm can be seen in figure 12, while the results for the Metropolis-Hastings algorithm can be seen in 13.



(a) (b)

Figure 12: Learning rate of 0.35, 1 particle, 1 dimension



(a) (b)

Figure 13: The mean energy plotted as a function of... Learning rate of 0.001, 2 particle, 2 dimension

The chosen step length for the Metropolis algorithm was decided to be set as 0.5 for the upcoming calculations, while the time step for the Metropolis-Hastings algorithm was decided to be set as 0.4 for the upcoming calculations. The reason for this can be seen in section **??**.

CONTINUE FROM HERE! But step size and such are not right.

4.2.2 Finding the Lowest Energy: Varying the Variational Parameter by a Set Amount

To get an estimation of where the optimal alpha is laying, the mean energy was calculated by varying α by a set amount of 0.05 in the interval of $[0.3, 0.7]$. The results can be seen in figure 14.

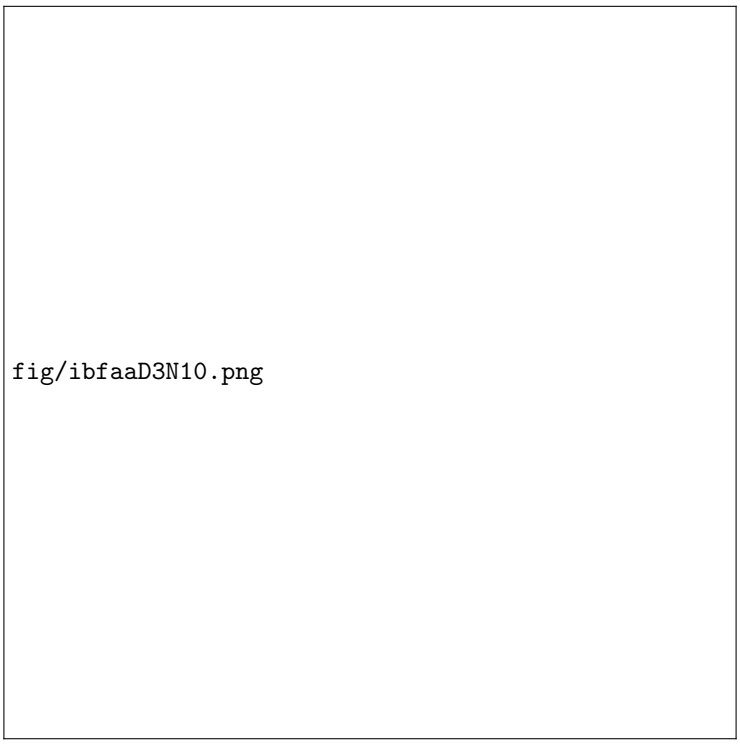


Figure 14: The mean energy plotted as a function of the variational parameter α, using the Metropolis algorithm with interacting particles and numerical differentiation. Metropolis steps=2^{18}, N=10, dimensions=3, and $\beta = 2.82843$

4.2.3 Finding the Lowest Energy: Finding the Variational Parameter with Gradient Decent

To find the best value of α in the interacting case, the gradient decent was performed with three different amounts of particles in three dimensions. From 14 it is quiet intuitive that the optimal α is laying between 0.45 and 0.55, which is why the initial α was set as 0.45 in all three cases. The results can be seen in figure 15, 16 and 17.

Figure 15: The mean energy plotted as a function of the variational parameter α, using the gradient decent method and the Metropolis algorithm with numerical differentiation and interacting particles. Initial $\alpha = 0.45$, Metropolis steps per GD cycle $=10^3$, GD step size $\gamma = 0.005$, N=10, dimensions=3, and $\beta = 2.82843$. Zoomed in plot of the last convergence part can be seen in the corner of the main plot. The method converges towards α=0.4951.

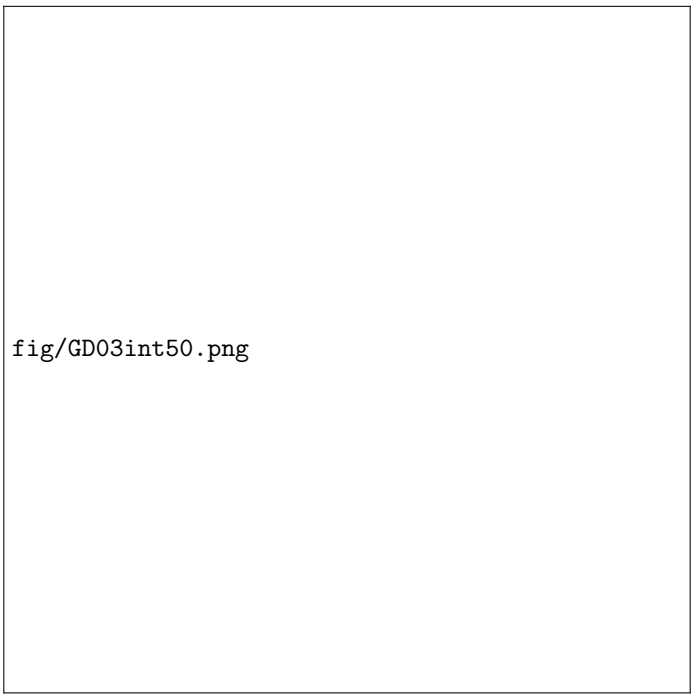


Figure 16: The mean energy plotted as a function of the variational parameter α, using the gradient decent method and the Metropolis algorithm with numerical differentiation with interacting particles. Initial $\alpha = 0.45$, Metropolis steps per GD cycle $=10^3$, GD step size $\gamma = 0.005$, N=10, dimensions=3, and $\beta = 2.82843$. The method settled at α=0.4847 after some unstable walking around.

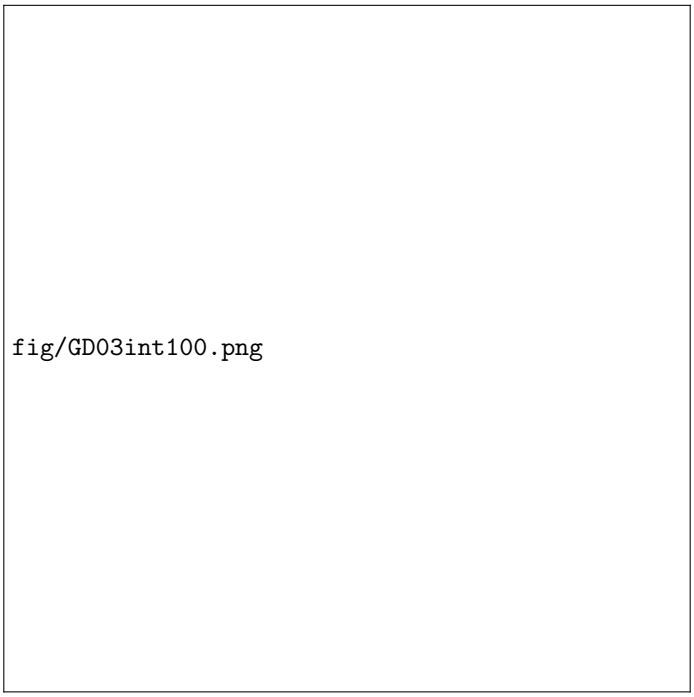


Figure 17: The mean energy plotted as a function of the variational parameter α, using the gradient decent method and the Metropolis algorithm with numerical differentiation with interacting particles. Initial $\alpha = 0.45$, Metropolis steps per GD cycle $=10^3$, GD step size $\gamma = 0.005$, N=10, dimensions=3, and $\beta = 2.82843$. The method settled at α=0.4696 after some unstable walking

4.2.4 The Local Energies

After running gradient decent for the three cases with interacting particles, the resulting α for each case was used to compute the local energy, and then compared to the values found in [1] and [2]. The results can be seen in table 7.

Particles	$E_{from[1][2]}(\hbar\omega)$	$E_C(\hbar\omega)$	$\mu_C(\hbar\omega)$	α	M. steps	Dissimilarity (%)	CPU time(s)	$E_C/N(\hbar\omega)$
10	24.2	24.396	0.004	0.4951	2^{16}	0.8	155.838	2.44
50	122	127.25	0.07	0.4847	2^{16}	4.3	17725.5	2.55
100	247	265.5	0.4	0.4696	2^{16}	7.0	259034	2.66
10	24.2	24.400	0.005	0.4951	2^{20}	0.8	4609.3	2.44

Table 7: Results from N=10, N=50 and N=100 particles with interacting particles, using the Metropolis algorithm with numerical differentiation and values of α found with gradient decent optimization. $\beta = 2.82843$, and 2^{16} metropolis steps were used in addition to one run with 2^{20} steps for N=10. The deviation was computed by using the blocking method. The undercored C stands for computed.

4.2.5 One Body Density

The one body density was plotted for two cases. One with 10 particles and one with 50 particles. Both the plots can be seen in figure 18.

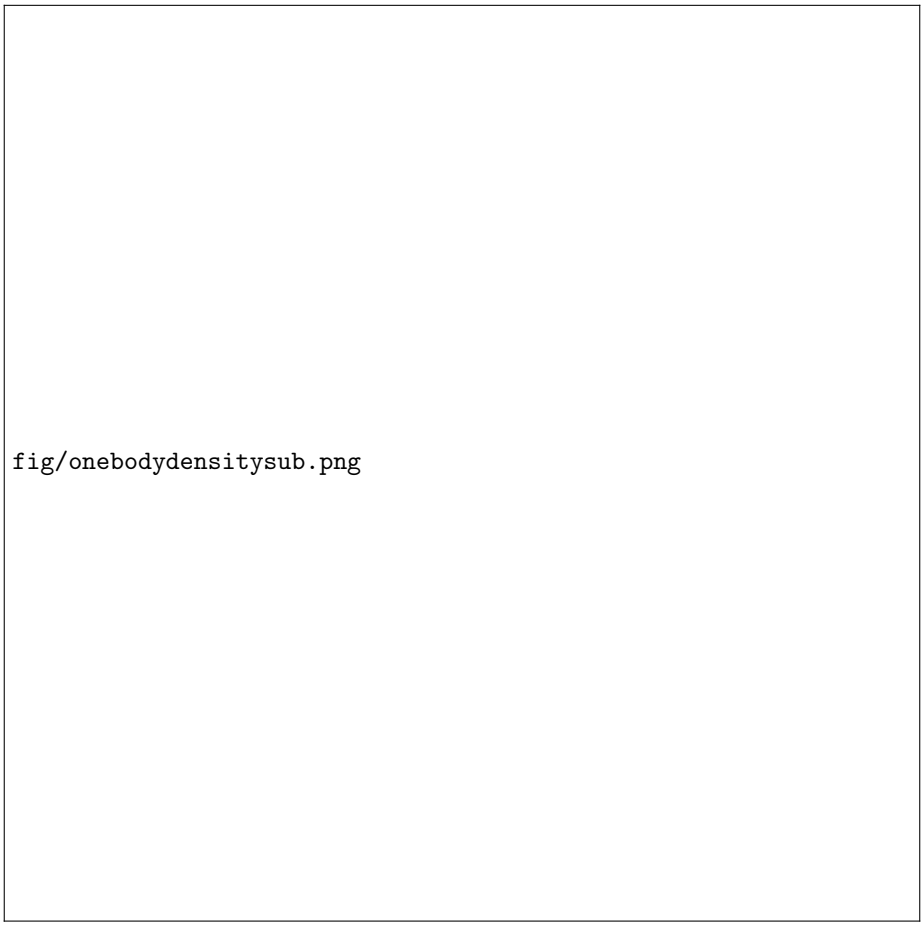


Figure 18: The one body density for N=50 in the upper subplot and N=10 in the lower subplot, with and without the Jastrow factor. The probability is plotted as a function of length r from the origin.

5 Discussion

5.1 Choosing the Distribution, Step Length, Time Step

It is easy to see in figure 4, that the Metropolis algorithm was the most stable with a step length of 1 or 0.5, compared to the functions with step length 0.1 and 0.25. The decision of choosing 0.5 over 1 was because of the amount of accepted steps. With a step length of 1, an acceptance rate of 74% followed, while with a step length of 0.5, an acceptance rate of 86% followed, making it the most preferable step length.

For the Metropolis-Hastings algorithm the most stable mean energy values was

obtained with a time step of 0.1 and 0.25. The reason 0.25 came out as the preferred time step, was because it followed with an acceptance rate of 65% while a time step of 0.1 only gave an acceptance rate of 38% which is quiet low using the Metropolis-Hastings algorithm.

5.2 Non Interacting Particles

5.2.1 Finding the Variational Parameter α

Finding the best value of α to be used in the non interacting case was a combination of plotting α as a function of the energy while changing α by a set amount. This gave an estimate of where the minimum was laying so it would be easier to choose an initial α when using gradient decent.

In figure **??** and **??** the only difference was that in the former figure the Metropolis algorithm was used, while in the later figure the Metropolis-Hastings algorithm was used. Both resulted in the optimal α of 0.5 which naturally shows that both methods uses the same optimal parameters.

The Metropolis algorithm continued to be used while finding α using gradient decent. To test the gradient decent method, figure 9 and 10 were created to see if choosing initial values on both sides of the minimum would result in the same α. It did, both initial values ended up with $\alpha = 0.5$, which shows that the gradient decent method works on both sides of the minimum.

Then the gradient decent method was tested with a higher amount of particles, N=100 to be exactly. With a higher amount of particles, α resulted in being 0.5 which was the expected. The unexpected was that the road towards $\alpha = 0.5$ was a bit more bumpy than expected. In figure 11 the gradient decent method had some jumps on the way towards 0.5, which could be due to some stability issues in the implementation of the algorithm, in addition to the amount of Metropolis steps used per iteration might have been a bit low for the energy to settle down. Due too all the gradient decent tests finishing with the favoured results of $\alpha = 0.5$, it was concluded that the gradient decent managed to give preferred results, and the affection of the instability not being on a large scale. The gradient decent method continued being used forward in the project.

5.2.2 Calculation of the Local Energies

Having a look at table 1, 2 and 3 showing the results from the Metropolis algorithm, shows that the calculated energies are really close to the exact energies. The analytically calculated energies equals the correct energies while the numerical calculated energies are almost exactly the same, but with a really small deviation on the order between 10^{-9} to 10^{-7}. Which shows that both analytically and numerical differentiation works just fine. The same conclusion stands for table 4, 5 and 6 showing results form the Metropolis-Hastings algorithm.

Timewise, running the simulations seem to increase the time needed to run quiet steady with increasing particles for both Metropolis algorithm and Metropolis-Hastings algorithm. The Metropolis-Hastings algorithm seems to use a bit less time than the Metropolis Algorithm, but this might be the cause of a bit ineffective implementation for the Metropolis-Hastings case. The time needed to simulate seems to stay quiet constant when adding and subtracting dimensions, except for when the particle amounts with numerical differentiation reaches 100, then the time needed to simulate increases almost 100x more than the time needed for simulating 10x less particles. This might be due to some 'traffic jam' when the code parallelizes such a high amount of particles. Even though the results are still correct.

5.3 Interacting Particles

5.3.1 Finding the Variational Parameter α

Finding the best value of α to be used in the interacting case was a combination of plotting α as a function of the energy in figure 14 This gave an estimate of where the minimum was laying, which opened up for choosing $\alpha = 0.45$ as the initial value for the gradient decent simulations in the interacting case.

The gradient decent method was used three times for the interacting case for N=10,50 and 100 particles which gave α= 0.4951, 0.4847 and 0.4696 respectively. Which seems like values within reason of the quantum mechanical system. In figure 15, 16 and 17, which shows the route from 0.45 to the mentioned values of α, exposes the instability of the gradient decent method with some of the energy values in the plots being lower than the minimum found. The reason causing the instability is most probably due to the low amount of Metropolis steps used when calculating the derivative in the gradient decent. Because of time consuming simulations the amount of steps were lowered from 10^5 to 10^3 when moving from the non interacting case, to the interacting case. The amount of steps being the cause is supported by the plots from gradient decent applied to the non interacting cases compared to the same plots for the interacting cases, showing that the non interacting cases is much more stable.

5.3.2 Calculation of the Local Energies

Having a look at table 7 showing the results from the fully interacting cases, shows that the calculated energies using the values of α are quiet close to the referenced energies in [1] and [2]. With N=10 the difference was 0.8%, for N=50, the difference was 4.3% and when N=100, the difference is 7.0%. Which is results that is within approval. Especially the case with 10 particles which had under 1% difference. The energy per particle seems to increase a small bit when increasing the particle amounts, but this might be due to the increase in numerical instability when increasing the amount of particles.

The three interacting simulations from table 7 were done with 2^{16} Metropolis steps. The simulation were done again for N=10 particles but this time with

2^{20} Metropolis steps. The second simulation resulted in approximately the same result which assists that the simulations in the same table were run with a high enough number of Metropolis steps.

5.3.3 The One Body Density

From figure 18 it is not quiet easy to see a difference between the plots with the Jastrow factor included and without the Jastrow factor which indicates that the interacting forces between particles are quiet weak in this case. In the lower plot with 50 particles in the system, especially in the range of [0.5,1] the plot without the Jastrow factor is pushing itself a bit in front of the plot with the Jastrow factor. This indicates that the Jastrow factor contributes with a repulsive force in the system. That supports the upper subplot in the same figure where there is only 10 particles present, because less particles leads to less contribution from the Jastrow factor since there is less particles to interact with each other.

6 Conclusion

After exploring the quantum mechanical system using the variational Monte Carlo method, it has been shown how it is possible to simulate trapped bosons in different types of harmonic oscillator traps, in one, two and three dimensions. Both the Metropolis algorithm and the Metropolis-Hastings algorithm as sample methods worked well and gave approximately similar results. The simulation in both cases where the bosons were interacting and not interacting, expressed satisfactory results.

In the non interacting case it was also shown that the ground state energies were calculated with such a small error that claiming the results to be exact is within reach. When the bosons were interacting, the results were closer to the referenced values of [1] and [2] when the amount of bosons in the trap were smaller. Even though the results were shown to be acceptable with a dissimilarity of 7% in the case of having 100 interacting bosons, while having 0.8% dissimilarity when the amount of bosons summed up to 10. The 50 boson case had 4.3% dissimilarity.

The one body density also shows that in this quantum mechanical system of interacting bosons, the forces contributed by interaction is quiet weak, and needs more particles embedded in the system for a higher contribution of in this case repulsion.

6.1 Prospects for the future

The main prospects for the future would quiet naturally be to optimize the gradient decent method or switch out gradient decent with another optimisation algorithm, for example a stochastic gradient decent method. As most codes, there is always room for optimization. Optimizing the code where it is possible, would be satisfying, making the code run more effectively saving computer

resources. There would also be naturally to scout for more ways to calculate the double derivative of the trial function. Another prospect for the future that sounds quiet fun, would be to inspect similarities and dissimilarities in the same quantum mechanical system if it were to contain fermions instead of bosons.

References

[1] M. Taut, "Two electrons in an external oscillator potential: Particular analytic solutions of a coulomb correlation problem," *Phys. Rev. A*, vol. 48, pp. 3561–3566, 5 Nov. 1993. DOI: 10.1103/PhysRevA.48.3561. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.48.3561.

[2] G. Carleo and M. Troyer, "Solving the quantum many-bodyproblem with artificialneural networks," *Science*, vol. 355, pp. 602–606, 6325 Feb. 2017. DOI: 10.1126/science.aag2302. [Online]. Available: https://science.sciencemag.org/content/355/6325/602/tab-pdf.

[3] D. J. Griffiths, *Introduction to Quantum Mechanics (3rd Edition)*, 3rd. 2018, pp. 327–331.

[4] P. Niane, *Variational monte carlo for bosonic systems*, [Online; accessed 18-Mai-2021], Mar. 2021. [Online]. Available: https://github.com/philipkarim/Variational-Monte-Carlo--Fys4411/blob/main/docs/report.pdf.

[5] M. Hjorth-Jensen, *Project 2, the restricted boltzmannn machine applied to the quantum many body problem*, http://compphysics.github.io/ComputationalPhysics2/doc/Projects/2021/Project2/Project2ML/pdf/Project2ML.pdf, [Online; accessed 18-Mai-2021], 2021.

[6] ——, *Advanced topics in computational physics: Computational quantum mechanics*, https://compphysics.github.io/ComputationalPhysics2/doc/LectureNotes/_build/html/intro.html, [Online; accessed 18-Mai-2021], 2021.

[7] A. Oppermann, *Deep learning meets physics: Restricted boltzmann machines part i*, https://towardsdatascience.com/deep-learning-meets-physics-restricted-boltzmann-machines-part-i-6df5c4918c15, [Online; accessed 18-Mai-2021], 2018.

[8] M. Hjorth-Jensen, *Gradient methods*, http://compphysics.github.io/ComputationalPhysics2/doc/pub/week6/pdf/week6.pdf, [Online; accessed 20-May-2021], 2021.

[9] M. Jonsson, "Standard error estimation by an automated blocking method," *Phys. Rev. E*, vol. 98, p. 043 304, 4 Oct. 2018. DOI: 10.1103/PhysRevE.98.043304. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.98.043304.

A Appendix

A.1 Source code

All the source code is located in this GitHub repository.

A.2 Analytical Expression of the Local Energy

A.2.1 Local Energy-Metropolis and Metropolis-Hastings algorithms

To derive the analytical expression for the local energy, an intuitive place to start is by the expression for the local energy and inserting the Hamiltonian expression:

The local energy is given by the following:

$$E_L = \frac{1}{\Psi}\hat{H}\Psi = \sum_{i=1}^{M}\left(-\frac{1}{2\Psi}\nabla_i^2\Psi + \frac{1}{2}\omega^2 r_i^2\right) + \sum_{i<j}\frac{1}{r_{ij}} \tag{27}$$

Where the main part of the computations will be the double derivative of the wave function. Which looks like this:

$$\frac{1}{\Psi}\nabla_i^2\Psi \tag{28}$$

Which can be written as:

$$\left(\frac{1}{\Psi}\nabla\Psi\right)^2 + \nabla\left(\frac{1}{\Psi}\nabla\Psi\right) = \left[\nabla\log\Psi\right]^2 + \nabla^2\log\Psi \tag{29}$$

Which is easier to compute. The differentiated logarithm of Ψ can then be written as the following:

$$\log\Psi = -\log Z - \sum_{i=1}^{M}\left(\frac{(X_i - a_i)^2}{2\sigma^2}\right) + \sum_{j=1}^{N}\log\left(1 + \exp\left(b_j + \sum_{i=1}^{M}\frac{X_i w_{ij}}{\sigma^2}\right)\right) \tag{30}$$

Then by defining the sigmoid function as:

$$\delta(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x} \tag{31}$$

and defining the following which is used as input in the sigmoid function:

$$\delta_j^{input} = b_j + \sum_{i=1}^{M}\frac{X_i w_{ij}}{\sigma^2} \tag{32}$$

When differentiating the logarithm of Ψ with respect to the visible nodes the result is as follows:

$$\begin{aligned}\frac{\partial \log \Psi}{\partial X_i} &= \frac{(a_i - X_i)}{\sigma^2} + \sum_{j=1}^{N} \frac{w_{ij} \exp\left(b_j + \sum_{i=1}^{M} \frac{X_i w_{ij}}{\sigma^2}\right)}{\sigma^2 \left(1 + \exp\left(b_j + \sum_{i=1}^{M} \frac{X_i w_{ij}}{\sigma^2}\right)\right)} \\ &= \frac{a_i - X_i}{\sigma^2} + \sum_{j=1}^{N} w_{ij} \delta(\delta_j^{input})\end{aligned} \tag{33}$$

Now that the first derivative is defined, the next derivative can be written as follows by differentiate one more time:

$$\frac{\partial^2 \log \Psi}{\partial X_i^2} = -\frac{1}{\sigma^2} + \sum_{j=1}^{N} \frac{w_{ij}^2}{\sigma^4} \delta(\delta_j^{input}) \delta(-\delta_j^{input}) \tag{34}$$

Then by inserting the derivatives into equation (27) the final expression for the local energy of the system is given as:

$$E_L = -\frac{1}{2} \sum_{i=1}^{M} \left[\left(\frac{a_i - X_i}{\sigma^2} + \sum_{j=1}^{N} w_{ij} \delta(\delta_j^{input}) \right)^2 \right. \tag{35}$$

$$\left. - \frac{1}{\sigma^2} + \sum_{j=1}^{N} \frac{w_{ij}^2}{\sigma^4} \delta(\delta_j^{input}) \delta(-\delta_j^{input}) - \omega^2 r_i^2 \right] + \sum_{i<j} \frac{1}{r_{ij}} \tag{36}$$

Where M is the number of visible nodes, and N is the unmber of hidden nodes.

A.2.2 Local Energy-Gibbs sampling

When using Gibbs sampling the wave function is set to $\Psi(x) = \sqrt{F_{rbm}}$ which means that the local energy will be a bit different than for the Metropolis and Metropolis-Hastings algorithm. To derive the local energy when using Gibbs sampling the energy contribution from the interaction between particles and the interaction from the trap will still be the same due to the wave function being removed after applying the Hamiltonian.

When differentiating the logarithm of Ψ the same way as for the Metropolis and Metropolis-Hastings algorithm, but using the following formuala first:

$$log\sqrt{\Psi} = \frac{1}{2} log\Psi \tag{37}$$

gives the local energy by adding a factor of $\frac{1}{2}$ to the logarithmic terms in equation (29). Which gives the following local energy after inserting the expressions for the derivatives and the rest of the energy terms:

$$E_{L,Gibbs} = -\frac{1}{2}\sum_{i=1}^{M}\Bigg[\frac{1}{4}\left(\frac{a_i - X_i}{\sigma^2} + \sum_{j=1}^{N} w_{ij}\delta(\delta_j^{input})\right)^2 \tag{38}$$

$$- \frac{1}{2\sigma^2} + \sum_{j=1}^{N}\frac{w_{ij}^2}{2\sigma^4}\delta(\delta_j^{input})\delta(-\delta_j^{input}) - \omega^2 r_i^2\Bigg] + \sum_{i<j}\frac{1}{r_{ij}} \tag{39}$$

A.3 SGD-Derivatives of the free parameters

The derivatives used in the stochastic gradient decent method are the derivatives of the wave function with respect to the parameters. Since the Metropolis- and Metropolis-Hastings algorithm has a slightly different wave function than used in Gibbs sampling the derivatives will also be slightly different.

A.3.1 Metropolis and Metropolis-Hasting case

The derivatives of the wave function is used in equation (20), which gives the following expressions:

$$\frac{1}{\Psi_T}\frac{\partial \Psi_T}{\partial \alpha_k} = \frac{\partial}{\partial \alpha_k}\log \Psi_T \tag{40}$$

Then by inserting equation (30) into the equation and differentiate with respect to the different free parameters leaves the following expression:

$$\frac{\partial \log \Psi_T}{\partial a_k} = \frac{X_k - a_k}{\sigma^2} \tag{41}$$

$$\frac{\partial \log \Psi_T}{\partial b_k} = \frac{\exp\left(b_k + \sum_{i=1}^{M}\frac{X_i w_{ik}}{\sigma^2}\right)}{1 + \exp\left(b_k + \sum_{i=1}^{M}\frac{X_i w_{ik}}{\sigma^2}\right)} = \delta(\delta^{input}) \tag{42}$$

$$\frac{\partial \log \Psi_T}{\partial w_{kl}} = \frac{X_k \exp\left(b_l + \sum_{i=1}^{M}\frac{X_i w_{il}}{\sigma^2}\right)}{\sigma^2\left(1 + \exp\left(b_l + \sum_{i=1}^{M}\frac{X_i w_{il}}{\sigma^2}\right)\right)} = \frac{X_k}{\sigma^2}\delta(\delta^{input}) \tag{43}$$

A.3.2 Gibbs sampling case

Because of the following relation

$$log\sqrt{\Psi} = \frac{1}{2} log\Psi \tag{44}$$

The derivatives with respect to the free parameters will follow the same result as for the Metropolis and the Metropolis-Hasting algorithm, but with a factor $\frac{1}{2}$, which gives the following results:

$$\frac{\partial \log \Psi_T}{\partial a_k} = \frac{X_k - a_k}{2\sigma^2} \tag{45}$$

$$\frac{\partial \log \Psi_T}{\partial b_k} = \frac{1}{2}\delta(\delta^{input}) \tag{46}$$

$$\frac{\partial \log \Psi_T}{\partial w_{kl}} = \frac{X_k}{2\sigma^2}\delta(\delta^{input}) \tag{47}$$