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

June 5, 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 energies found using the Metropolis algorithm, Metropolis-Hastings algorithm and Gibbs sampling as sampling methods with one particle in multiple dimensions agrees at worst 98% with the analytical solution.

The interacting case on the other hand resulted in the following ground state energies when having 2 particles in 2 dimensions: 3.26 a.u. using Metropolis Algorithm, 3.25 a.u. using Metropolis-Hastings and 3.26 a.u. using Gibbs sampling. Compared to the analytical solution presented in [1], the dissimilarity was 8.7%, 8.3% and 8.7% using the Metropolis Algorithm, Metropolis-Hastings and Gibbs sampling respectively, all simulated with fine tuned parameters.

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 and fine tuned 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 later 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:

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

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:

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

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 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 method 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.

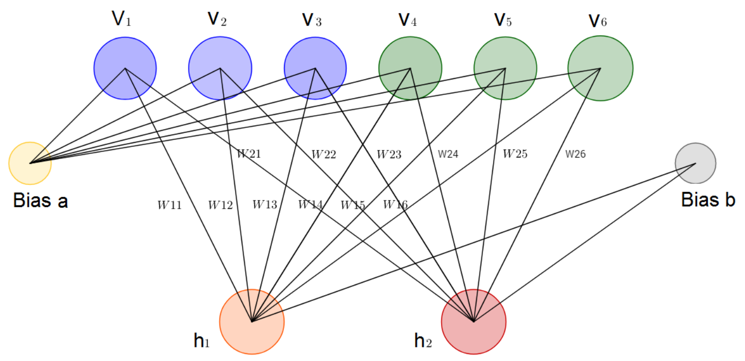


Figure 1: Architecture of a restricted Boltzmann machine, having the visible nodes connected to every hidden node and the visible bias, while the hidden nodes are connected to all the visible nodes and the hidden bias. The W's in the image is values of the weight matrix which decides how strong the connection 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 completed Monte Carlo simulation which will be called a RBM cycle in this article, 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 more and more optimized 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 minimized 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.

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

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. For a more detailed explanation of the method, feel free to have a look here [8].

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:

$$r^{new} = r^{old} + \in [-0.5, 0.5] \cdot dr \tag{21}$$

Where dr is a set step length. The step length is multiplied by a random number in the uniform interval [-0.5,0.5], to allow moves along both the positive and negative axis.

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 uniform 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 article 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 RBM cycles carries on, running the Monte Carlo simulations with the optimized parameters, which is optimised even further after each cycle, until the SGD tol is reached or the RBM cycles run 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 RBM cycles are 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 times 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

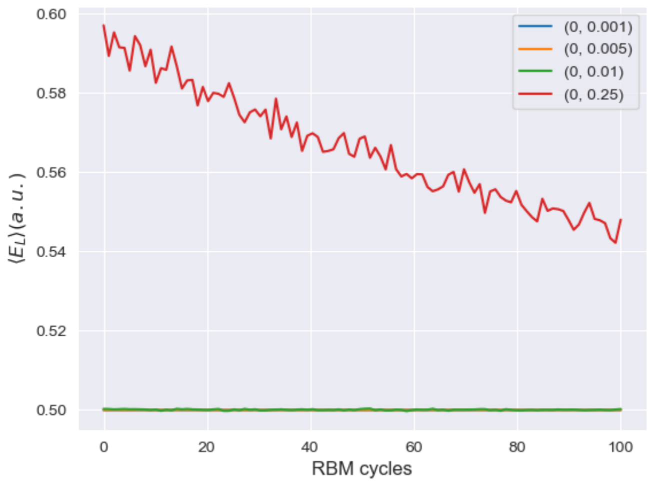
The number of metropolis steps used in all computations was 2^{20} unless stated otherwise.

4.1 Non Interacting System

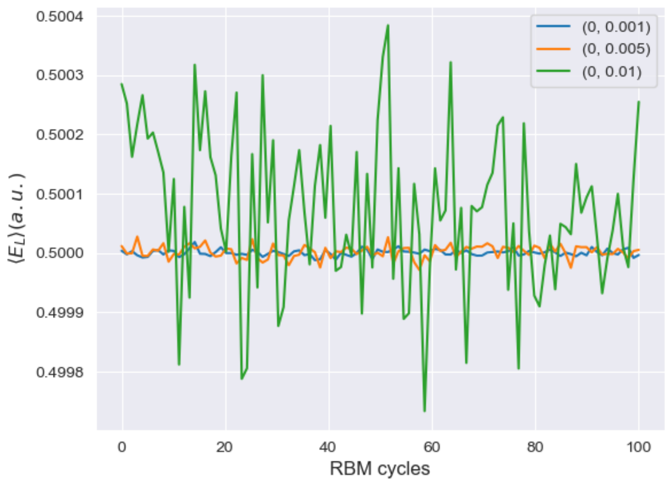
In this subsection the results for non interacting systems will be unveiled. All simulations in this subsesction are ran for 1 particle in 1 dimension, 2 hidden nodes and σ=1 unless else is stated.

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 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.

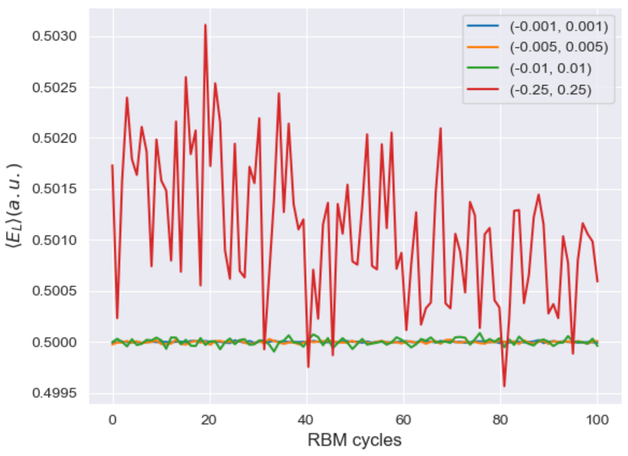


(a)

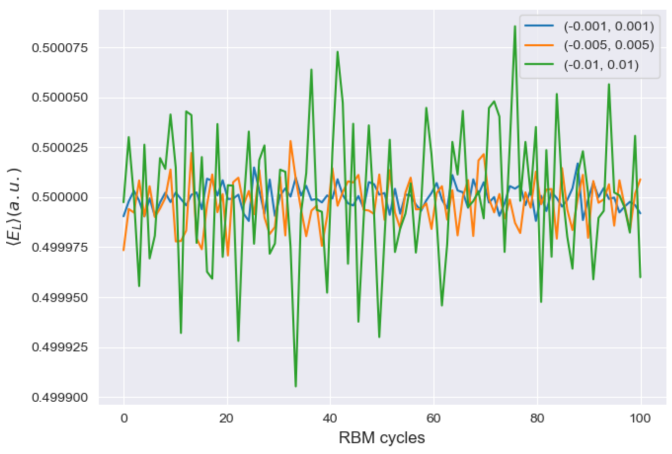


(b)

Figure 2: Energy plotted as a function of RBM cycles with normal distribution and different initialisation values. The sampling method used was the Metropolis algorithm. The simulation was ran with a learning rate of 0.001. a) Shows all initialisation values, b) shows the most stable ones for a better view.



(a)



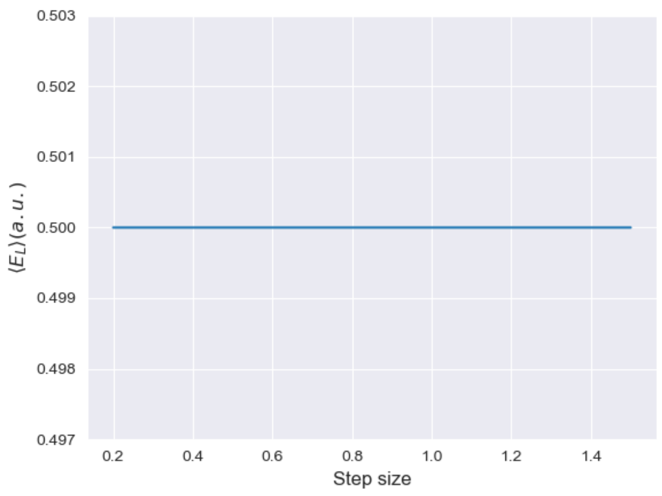
(b)

Figure 3: Energy plotted as a function of RBM cycles with uniform distribution and different initialisation values. The sampling method used was the Metropolis algorithm. The simulation was ran with a learning rate of 0.001. a) Shows all initialisation values, b) shows the most stable ones for a better view.

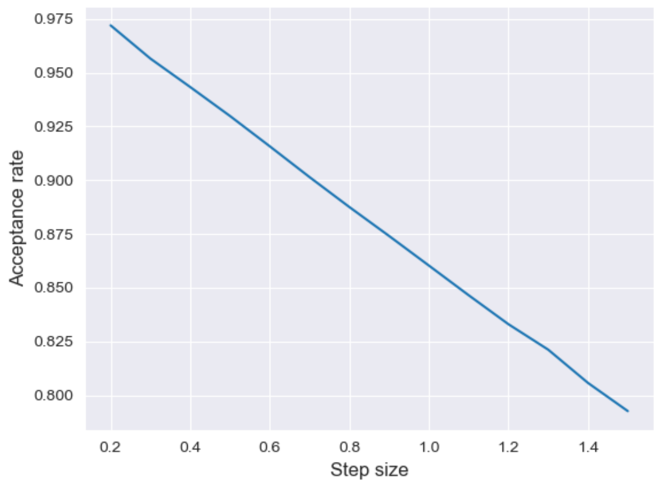
Normal distribution was chosen to be used with an initialisation value of 0.001 for the rest of the simulations, due to the reasons given in section 5.1.1.

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 energies were plotted as a function of step lengths and time 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.

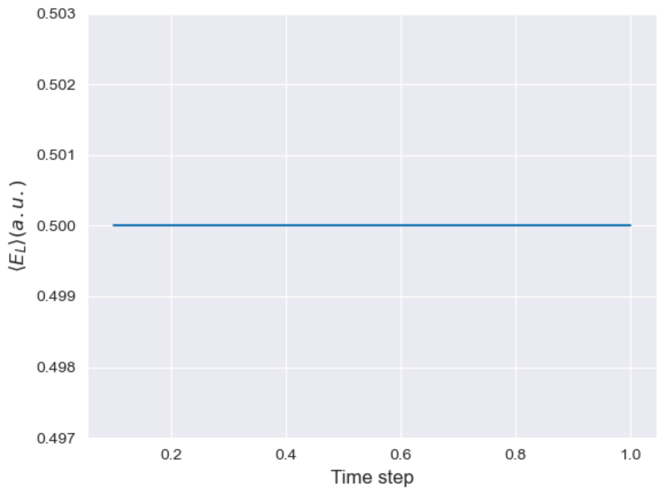


(a)

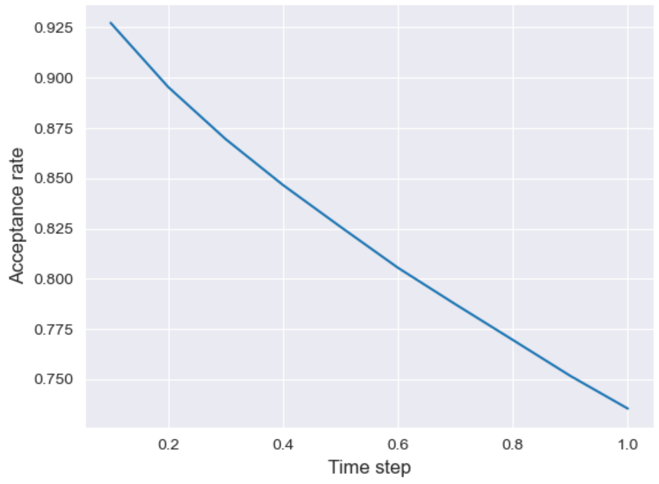


(b)

Figure 4: Metropolis algorithm, learning rate=0.35. a) Energy vs. step size. b) Acceptance rate vs. step size



(a)



(b)

Figure 5: Metropolis-Hastings algorithm, learning rate=0.35. a) Energy vs. time step. b) Acceptance rate vs. time step.

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.

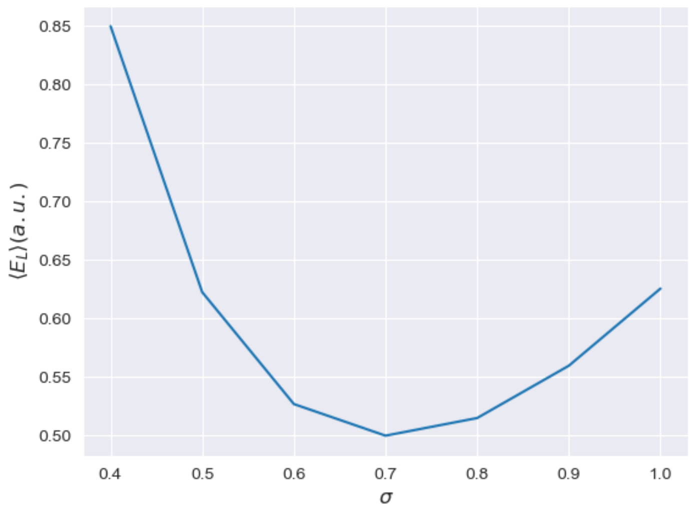
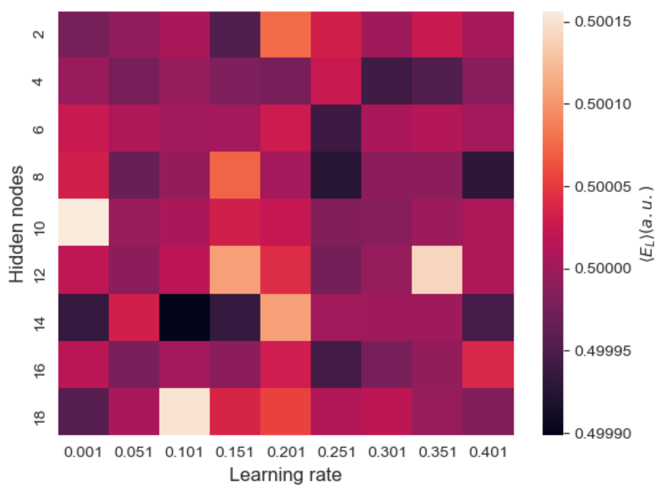


Figure 6: Energy plotted as a function of σ using Gibbs sampling.

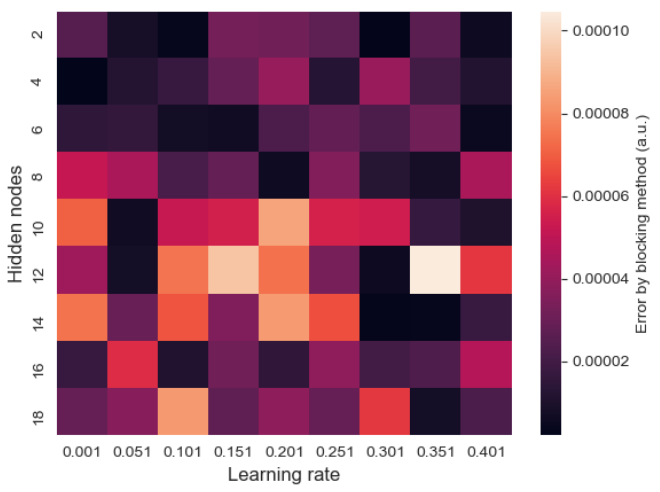
The resulting σ was chosen to be 0.7 when using Gibbs sampling throughout the article.

4.1.4 Investigating the learning rate and number of hidden nodes

The system was simulated for different amounts of hidden nodes in combination with different learning rates. The heatmap of the results using Metropolis-Algorithm, Metropolis-Hastings and Gibbs sampling can be seen in figure 7, 8 and 9 respectively.

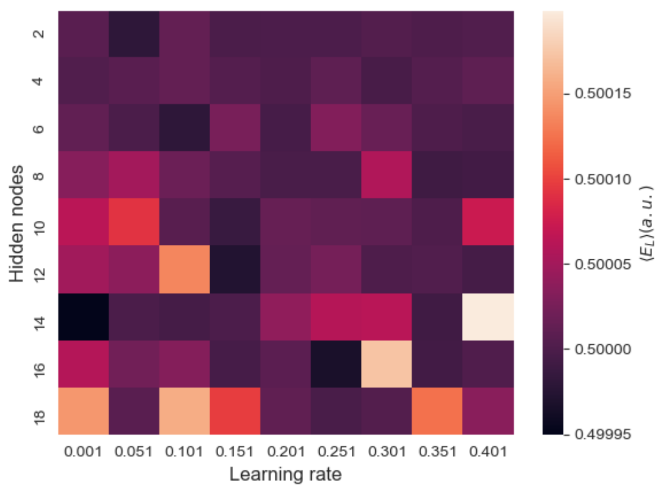


(a)

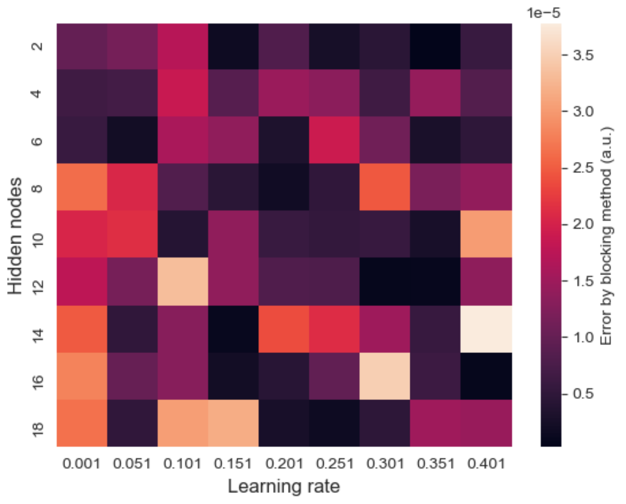


(b)

Figure 7: Heatmap of learning rates and hidden nodes using the Metropolis algorithm with 2^{18} steps and 40 RBM cycles. a) Energy b) The corresponding error

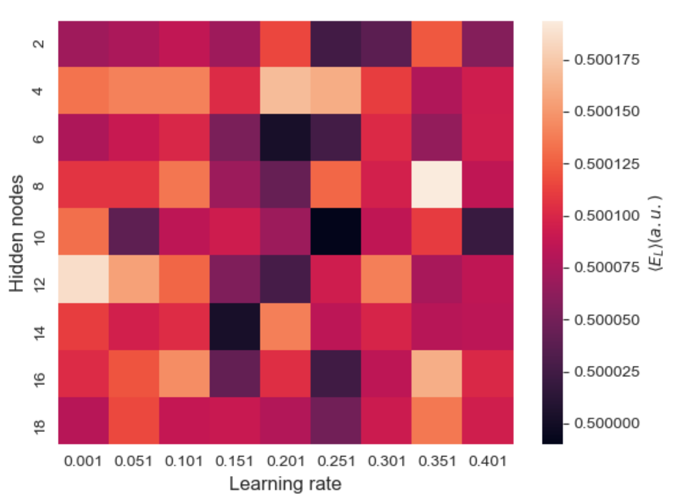


(a)

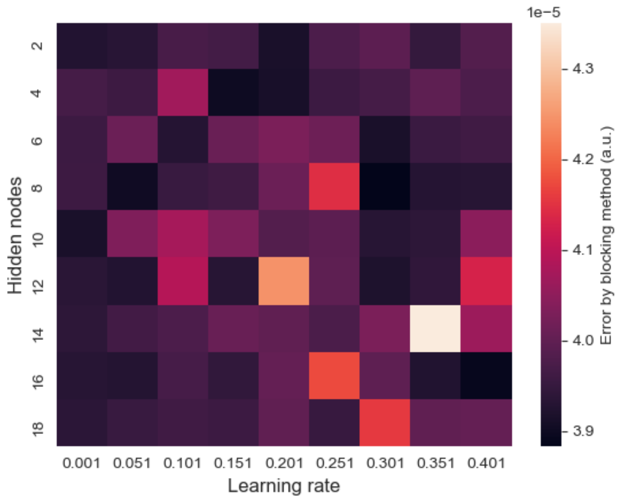


(b)

Figure 8: Heatmap of learning rates and hidden nodes using Metropolis-Hastings with 2^{18} steps and 40 RBM cycles. a) Energy b) The corresponding error



(a)



(b)

Figure 9: Heatmap of learning rates and hidden nodes using Gibbs sampling with 2^{18} steps and 40 RBM cycles. a) Energy b) The corresponding error

From figure 7, 8 and 9, the optimal learning rates and amount of hidden nodes in the non interacting case can be seen in table 1.

	Hidden nodes	Learning rate
Metropolis algorithm	14	0.101
Metropois-Hastings	14	0.001
Gibbs sampling	10	0.251

Table 1: Ideal amount of hidden nodes and learning rates found, in a system of non interacting particles

4.1.5 The Local Energies

The methods were tested for a variety of different dimensions with one particle. The results can be seen in table 2.

Dimension	Solver	$E_C(a.u)$	μ_C	$E_A(a.u)$	CPU time(s)
1	Metropolis algorithm	0.50	$7.7\cdot 10^{-5}$	0.5	9.15
	Metropolis-Hastings	0.50	$1.5\cdot 10^{-6}$	0.5	7.68
	Gibbs sampling	0.50	$2.0\cdot 10^{-5}$	0.5	7.61
2	Metropolis algorithm	0.99	$7.5\cdot 10^{-3}$	1.0	13.0
	Metropolis-Hastings	1.00	$1.7\cdot 10^{-3}$	1.0	11.3
	Gibbs sampling	1.00	$7.0\cdot 10^{-4}$	1.0	11.4
3	Metropolis algorithm	1.48	$1.4\cdot 10^{-2}$	1.5	14.9
	Metropolis-Hastings	1.49	$3.7\cdot 10^{-3}$	1.5	12.0
	Gibbs sampling	1.50	$1.2\cdot 10^{-3}$	1.5	15.5

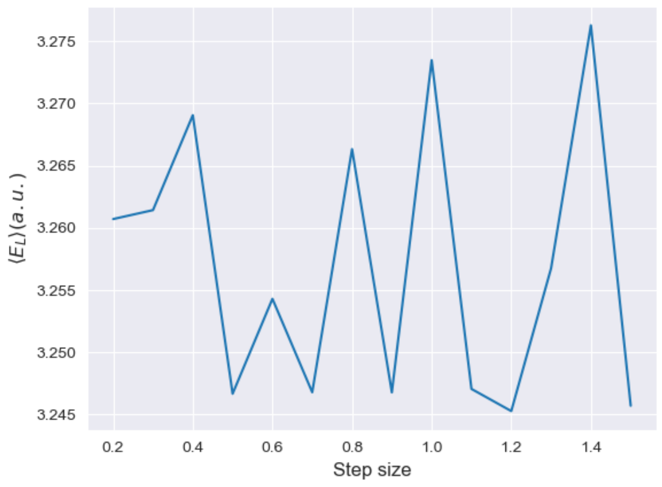
Table 2: Local energies in multiple dimensions with 1 particle in the system, using the optimal parameters found in section 4.1. RBM cycles=100. The standard deviation μ is the error computed by the blocking method. The undercored C and A stands for computed and analytical.

4.2 Interacting Particles

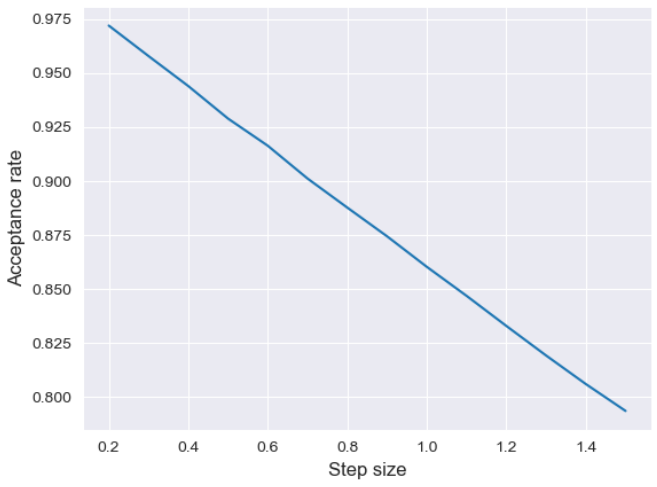
Then it's time for the interacting cases. In this subsection the results for interacting particle systems will be unveiled. All simulations in this subsection are ran for 2 particles in 2 dimensions unless else is stated.

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 10, while the results for the Metropolis-Hastings algorithm can be seen in 11.

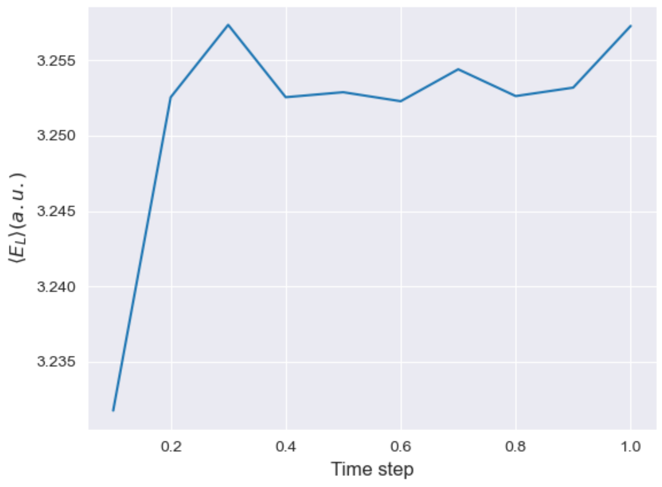


(a)

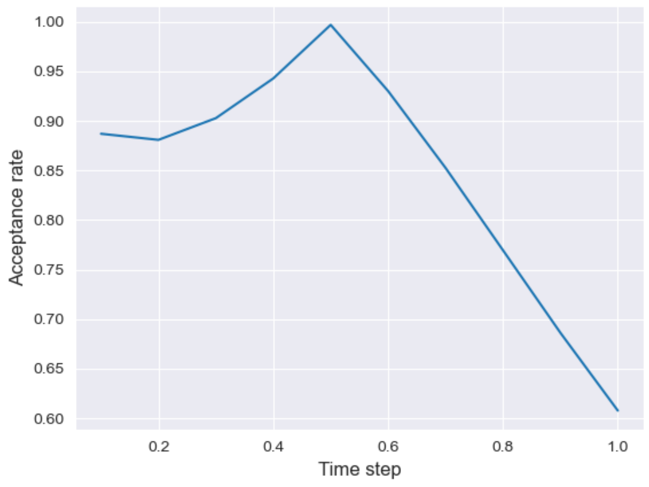


(b)

Figure 10: Metropolis algorithm, learning rate=0.001, 2 hidden nodes. a) Energy vs. step size. b) Acceptance rate vs. step size



(a)



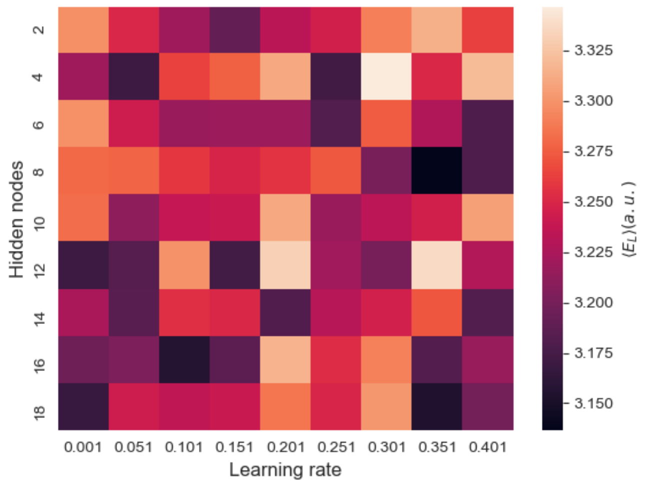
(b)

Figure 11: Metropolis-Hastings algorithm, learning rate=0.001, 2 hidden nodes. a) Energy vs. time step. b) Acceptance rate vs. time step

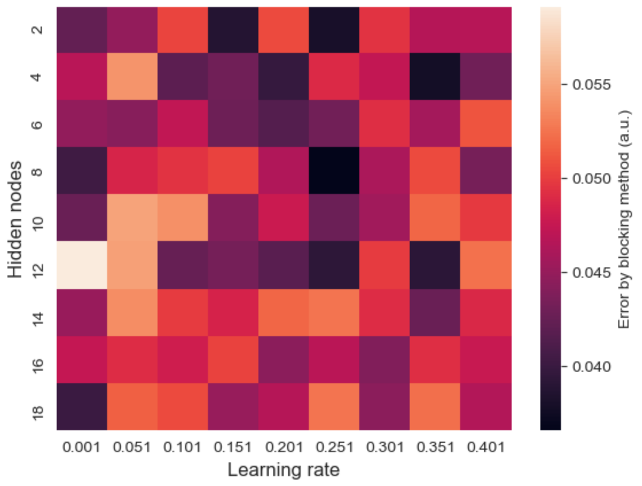
The chosen step length for the Metropolis algorithm was decided to be set as 0.5 for the upcoming calculations, while the time step for the Metropolis-Hastings algorithm was decided to be set as 0.4 for the upcoming calculations. The reason for this can be seen in section **??**.

4.2.2 Investigating the learning rate and number of hidden nodes

The system was simulated for different amounts of hidden nodes in combination with different learning rates. The heatmap of the results using Metropolis-Algorithm, Metropolis-Hastings and Gibbs sampling can be seen in figure 12, 13 and 14 respectively.

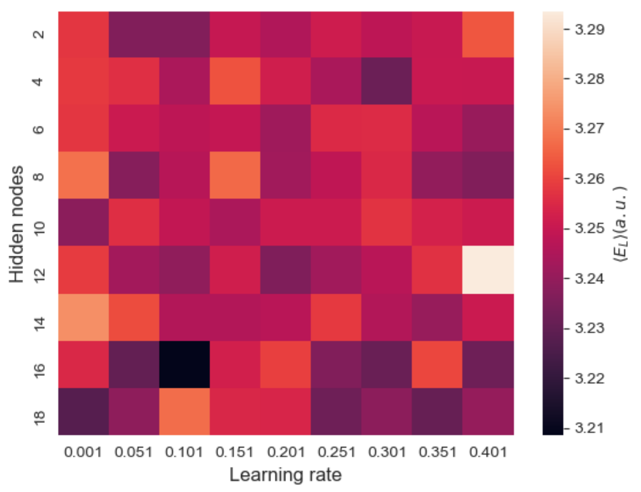


(a)

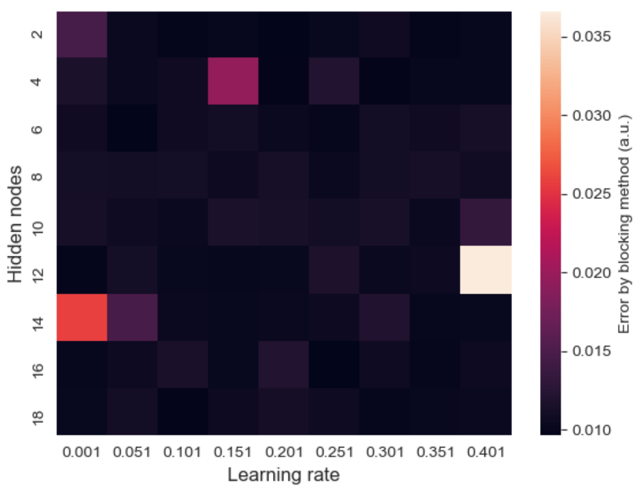


(b)

Figure 12: Heatmap of learning rate and number of hidden nodes using the Metropolis Algorithm with 2^{18} steps. a) Energy. b) The corresponding error

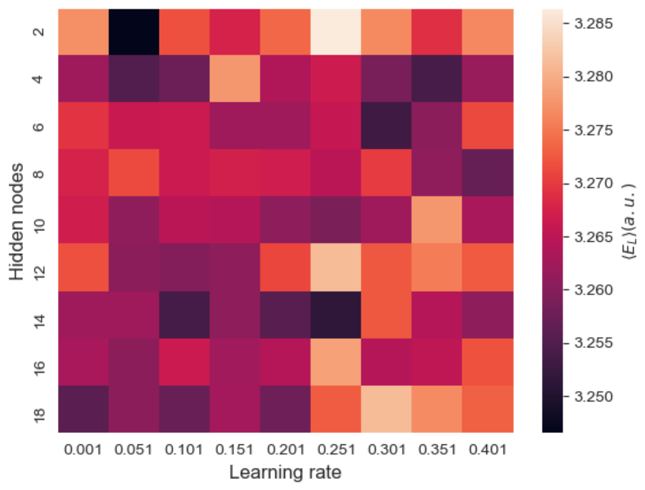


(a)

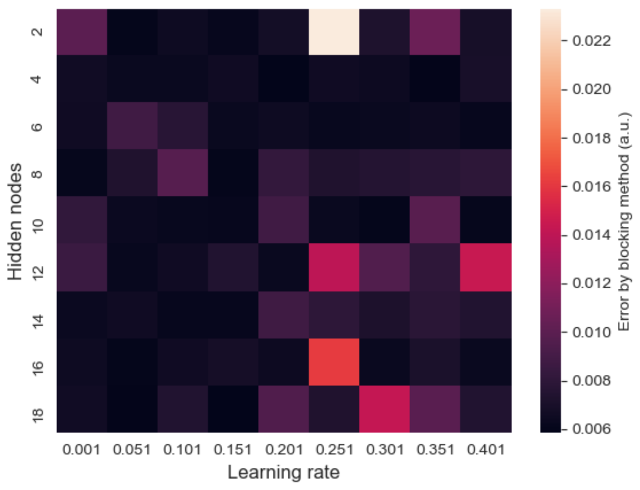


(b)

Figure 13: Heatmap of learning rate and number of hidden nodes using Metropolis-Hastings with 2^{18} steps. a) Energy. b) The corresponding error



(a)



(b)

Figure 14: Heatmap of learning rate and number of hidden nodes using Gibbs sampling with 2^{18} steps. a) Energy. b) The corresponding error

From figure 12, 13 and 14, the optimal learning rate and amount of hidden nodes in the non interacting case can be seen in table 3.

	Hidden nodes	Learning rate
Metropolis Algorithm	8	0.351
Metropois-Hastings	16	0.101
Gibbs sampling	2	0.051

Table 3: Optimal amount of hidden nodes and learning rate with interacting particles

4.2.3 The Local Energies

The methods were tested for a variety of different dimensions with one particle. The results can be seen in table 4.

Dimension	Particles	Solver	$E_C(a.u)$	μ_C	$E_A(a.u)$	CPU time(s)
2	2	Metropolis algorithm	3.26	$2.6 \cdot 10^{-2}$	3.0	15.1
2	2	Metropolis-Hastings	3.25	$5.2 \cdot 10^{-3}$	3.0	12.2
2	2	Gibbs sampling	3.26	$2.9 \cdot 10^{-3}$	3.0	18.4

Table 4: Local energies in 2 dimensions with 2 particles in the system, using the optimal parameters found in section 4.2. RBM cycles= 100. The standard deviation μ was computed using the blocking method. The undercored C and A stands for computed and analytical

5 Discussion

5.1 Non interacting particles

5.1.1 Experimenting with different parameters

Distribution: There is quiet a few parameters within the system that was explored in search of finding the optimal parameters. To start of the experimentation, the distribution was investigated by exploring different initialization values for both a uniform- and a normal distribution of the particles. From figure 3 and 2 the two distributions doesn't seem to have too much difference in stability which might be due to the low number of particles in the systems. The low initialisation value of the distributions shows that in both cases, the lowest initialisation values are most stable, that is why the normal distribution with initialisation value of 0.001 was chosen throughout the article.

Step length: The energy seen in figure 4 shows to be quiet stable for a variety of step length, by using the chosen parameters. The acceptance rate on the other hand increased with decreasing step length, which motivates the choice of a step length of 0.5 with an acceptance rate of approximately 93% in the non interacting case to ensure that the acceptance rate is quiet high to ensure that a lot of the steps are being accepted, without having such a small step length that the system would have a hard time getting out of local minimums.

The same analysis was also performed in the Metropolis-Hasting case when choosing the time step. Figure 5 shows quiet the same trend as with the Metropolis algorithm. Due to this, we choose a time step of 0.25. A time step of 0.25 ensures that the acceptance rate of is quiet high with about 86%, but still not too high to keep the system out of local minimums due too a too small time step.

σ of the wave function: The σ of the wave function was set equal to 1 in the computations done in figure 4 and 5, which gave a stable results matching the analytical solution. Due to this the investigation of σ with the Metropolis- and Metropolis-Hastings algorithm seemed dispensable and was kept equal to 1 throughout the article. On the other hand, the optimal value of σ was not found when using gibbs sampling. Therefore different values of σ were used to compute the energy, where the results can be seen in figure 6. The minimum energy was found when using $\sigma = 0.7$. Therefore $\sigma = 0.7$ was used throughout the article.

Learning rate and hidden nodes: Choosing the optimal learning rate in the optimization algorithm and number of hidden nodes could be crucial to achieve the best results. The learning rate was therefore investigated by running the simulation for various learning rates and amount of hidden nodes, in addition the error computed by the blocking method was also computed to ensure that the error was not large when choosing the learning rate and hidden nodes for the final calculations.

The investigation of learning rate and hidden nodes when using the Metropolis algorithm can be seen in figure 7. The case which gave the lowest energy was the simulation with 14 hidden nodes and a learning rate of 0.101. The error for this specific case ended up as $6.8 \cdot 10^{-5}$ which is not too high, but still a bit high compared to the other simulations in the non interacting case using the Metropolis algorithm. Having a quick look at the same figure again shows that having less hidden nodes in the system tend to give the lowest error, which shows that a higher amount of hidden nodes might bring some instability to the system.

Searching for the best learning rate and optimal number of hidden nodes in the Metropolis-Hastings case gave the lowest energy when the learning rate was set to 0.001 and the number of hidden nodes was set to 14, which can be seen in 8. These parameters resulted in an error of $2.5 \cdot 10^{-5}$. Also here the energy seems to become a bit unstable when the number of hidden nodes increases, in addition to the learning rate decreasing.

The final sampler which was the gibbs sampler was also used to find the optimal parameters according to the learning rate and the number of hidden nodes. From figure 9 the optimal parameters proved to be the simulations ran with a learning rate of 0.251 and 10 hidden nodes, which gave an error of $4 \cdot 10^{-5}$.

All the methods resulted in heatmaps showing multiple amounts of hidden nodes and learning rates that proved to give satisfactory results, which gives the impression that the learning rate and number of hidden nodes is not that critical to get acceptable results as long as the simulation is ran for enough RBM cycles.

5.1.2 Calculation of the Local Energies

Having a look at table 2 showing the results for 1 particle in the system in multiple dimensions with the optimal parameters found, reveals the results to be really close to the analytical solutions in all three dimensions for all three sampling methods.

The error computed by the blocking method also revealed to be quiet small in the first dimension, but increasing when expanding the system to more dimensions. The deviation in 1 dimension appeared to be on the order of 10^{-5} and 10^{-6} for all the three sampling methods, then increasing the dimension to 2 revealed deviations on the order of 10^{-3} and 10^{-4}. Lastly increasing the dimensions to 3 revealed deviations on the order of 10^{-2} and 10^{-3}. Naturally there is some deviations within most of such quantum mechanical simulations, so the deviations seem to be adequate.

Timewise, running the simulations seem to increase the time needed to run the simulations a small bit within each sampling method, but to such a small degree that running the simulations wont be a problem timewise unless the system was to be simulated for larger quantities of particles.

The results for the non interacting system revealed to be satisfactory with results matching the analytical solutions to a large degree. The same procedure was therefore performed on the interacting particle case.

5.2 Interacting Particles

5.2.1 Experimenting with different parameters

Step length: The energy seen in figure 10 using the Metropolis algorithm appear to be a bit more sensitive to the step length than in the non interacting case. The lowest energy computed was introduced with a step length of 1.2, but due too the low acceptance rate of approximately 83% this step length was neglected. Instead by choosing a step length of 0.5 which had an energy which was approximately 0.0025 a.u higher, resulted in an acceptance rate which was approximately 10% higher. Therefore the step length of 0.5 was chosen when using the Metropolis algorithm in the interacting case.

Having a look at the plot in figure 11 using the Metropolis-Hasting algorithm shows that the method is more stable for various amounts of time steps than the Metropolis algorithm. Using a time step of 0.4 which has a local minimum in the energy plot, gives an acceptance rate of a bit less than 95%, which is a bit high but by decreasing the time step would make the energy increase quiet steeply. A time step of 0.4 was therefore chosen when using the Metropolis-Hastings algorithm with interacting particles.

Learning rate and hidden nodes: The investigation of learning rate and hidden nodes when using the Metropolis algorithm with interacting particles can be seen in figure 12. The case which gave the lowest energy was the simulation with 8 hidden nodes and a learning rate of 0.351. The error for this specific case ended up as 0.051 which is a bit higher than the non interacting case which is naturally due to the increase in particles, dimensions in addition to interaction.

Searching for the best learning rate and optimal number of hidden nodes in the Metropolis-Hastings case gave the lowest energy when the learning rate was set to 0.101 and the number of hidden nodes was set to 16, which can be seen in 13. These parameters resulted in an error of 0.011. Having a glance at the same figure again shows that the errors of the computations are quiet even spread out the heatmap, with some cases which stands a bit out.

The final sampler which was the gibbs sampler was also used to find the optimal parameters according to the learning rate and the number of hidden nodes. From figure 14 the optimal parameters proved to be the simulations ran with a learning rate of 0.051 and 2 hidden nodes, which gave an error of 0.006. The heatmap presenting the error in the same figure shows that the error is quiet stable, but has a few more cases that stands out when both the learning rate and the hidden nodes are high.

5.2.2 Calculation of the Local Energies

Having a look at table 4 showing the results for 2 particle in the system of 2 multiple dimensions with the optimal parameters found, shows that the results are acceptable.

The energy computed by using the Metropolis algorithm resulted in an energy of 3.26 a.u. with a deviation of $2.6 \cdot 10^{-2}$. The difference compared to the analytical solution is 8.7%.

The energy computed by using Metropolis-Hastings resulted in an energy of 3.25 a.u. with a deviation of $5.2 \cdot 10^{-3}$. The difference compared to the analytical solution is 8.3%.

The energy computed by using Gibbs sampling resulted in an energy of 3.26 a.u. with a deviation of $2.9 \cdot 10^{-3}$. The difference compared to the analytical solution is 8.7%.

By comparing the different sampling methods, the energy ended up with quiet similar energies, but the Metropolis-Hastings algorithm being a tiny bit better, while the Metropolis algorithm and Gibbs sampling giving the same energy. Comparing the deviations on the other hand, show that Metropolis-Hasting algorithm had the lowest deviation, with Gibbs sampling ending up in second place followed by the Metropolis algorithm last. It is worth mentioning that the results and deviations are quiet similar for all three methods, which might indicate that the dissimilarity from the analytical solution is not caused by the sampling methods, but rather the wave function chosen as the trial function.

6 Conclusion

After exploring quantum mechanical systems using the restricted Boltzmann machine, it has been shown how it is possible to simulate trapped particles in a harmonic oscillator trap in multiple dimensions, by using a neural network quantum state as the wave function. It was also shown how it is possible to optimize the parameters in the wave function by using the stochastic gradient descent method.

The sampling methods used was the Metropolis algorithm, Metropolis-Hastings algorithm and Gibbs sampling which proved to work well and gave approximately similar results. The simulations with all three samplers where the particles were interacting and not interacting, expressed satisfactory results.

In the non interacting case it was shown that the ground state energies for one particle in multiple dimensions were calculated with such a small error that claiming the results to be exact is near at hand. It was also revealed that when the particles were interacting in the system, the results were a bit of the analytical result, but still acceptable with all three sampling methods showing dissimilarities in the range of 8-9% for two particles in two dimensions. The dissimilarity in the results compared to the analytical solution has a considerable amount of chance of being reduced by fine tuning the parameters even more because the systems have shown to be a bit sensitive to some parameter changes especially in the interacting particle case.

6.1 Prospects for the future

The main prospects for the future would quiet naturally be to try fine tuning the parameters in a more sophisticated way, e.g. by finding the optimal σ by using SGD or gradient descent. Or one could also do a more detailed search for the optimal learning rate and number of hidden nodes, e.g. by inspecting the converging rate as the number of RBM cycles proceed or even combine the search with an optimization algorithm.

After working on this article, the most efficient way to push the computed energies even closer to the analytical solution of 3 a.u. feels quiet sensible to be by exploring and searching for a more appropriate wave function for the quantum mechanical system.

As most codes, there is always room for optimization. Optimizing the code where it is possible, would be satisfying, making the code run more efficiently saving computer resources.

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:

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

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

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 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:

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

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

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