# THE RESTRICTED BOLTZMANN MACHINE APPLIED TO THE VARIATIONAL MONTE CARLO

WRITTEN BY:

*Oliver Hebnes & Mohamed Ismail*

DEPARTMENT OF PHYSICS UIO

JUNE 7, 2020

**Abstract**

THE AIM OF THIS RESEARCH WAS TO IMPLEMENT A RESTRICTED BOLTZMANN MACHINE TO A QUANTUM MANY BODY SYSTEM INVOLVING UP TO TWO PARTICLES IN TWO DIMENSIONS, IN THE PURPOSE OF REPRODUCING THE EXACT CLOSED FORM EXPRESSIONS FOR THE GROUND STATE ENERGY FROM M. TAUT, PHYS. REV. A **48**, 3561 (1993). THE MACHINE LEARNING ALGORITHM IN QUESTION HAS BEEN OPTIMIZED WITH THE METHOD STOCHASTIC GRADIENT DESCENT.

THREE DIFFERENT SAMPLING METHODS HAVE BEEN IMPLEMENTED AND TESTED AGAINST EACH OTHER, NAMELY BRUTE FORCE, IMPORTANCE, AND GIBBS SAMPLING. FOR ANALYZING THE STATISTICAL DEVIATIONS, WE USED THE BLOCKING METHOD AND FOUND FOR THE CASE WITH ONE PARTICLE IN ONE DIMENSION THAT BRUTE FORCE ACHIEVED ACCURACY WITH NUMERICAL PRECISION OF $8.21 \cdot 10^{-8}$ FOR $E = 0.5$ A.U.

FOR THE SECOND CASE FOR TWO PARTICLES IN TWO DIMENSIONS WITH A REPULSIVE INTERACTION, WE FOUND ONCE AGAIN THAT BRUTE FORCE PROVED TO BE THE BEST SAMPLING METHOD, HOWEVER WITH A SIGNIFICANT LARGER DEVIANCY OF $1.58 \cdot 10^{-3}$ FOR COMPUTED $E = 3.071$ A.U., WHERE THE CORRECT ENERGY SHOULD BE $E = 3.0$ A.U.

# Contents

## 1 Introduction

In this project, we will study an one- and two-electron system in a two-dimensional harmonic oscillator, which is often referred to as a quantum dot.

The objective is to calculate the ground state energy of the system, by using a restricted Boltzmann machine (RBM) to shape our trial wave function. The idea of representing the wave function with a restricted Boltzmann machine was presented recently by G. Carleo and M. Troyer, Science 355, Issue 6325, pp. 602-606 (2017) [1]. They named such a wave function/network a *neural network quantum state* (NQS).

To calculate estimates of the ground state energy, we will minimize the energy of the system by utilizing the variational principle and optimize the RBM parameters by using gradient descent, such that we find the optimal wave function for the state, where we will use Variational Monte Carlo (VMC) with either the Metropolis algorithm or Gibbs sampling to move the system towards the most likely state. The methods and the discussion of the results will be showcased in the sections below.

## 2 Theory

### 2.1 Theoretical background and description of the physical system

We consider a system of electrons confined in a pure two-dimensional isotropic harmonic oscillator potential, with an idealized total Hamiltonian given by

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

The system we are observing will include fermions with a oscillator frequency $\omega$, as seen in the above Hamiltonian. The first part of the Hamiltonian includes a standard harmonic oscillator part

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

while the second part is the repulsive interaction between two electrons given by

$$\hat{H}_1 = \sum_{i<j} \frac{1}{r_{ij}},$$

with the distance between electrons given by $r_{ij} = |\boldsymbol{r}_i - \boldsymbol{r}_j|$. We define the modulus of the positions of the fermions (for a given electron $i$) as $r_i = \sqrt{r_{i_x}^2 + r_{i_y}^2}$. We have not accounted for Pauli's exclusion principle, thus an experiment with the repulsive interaction term will not include more than two fermions in a quantum dot yielding an energy of 3 a.u. in this project, yet an experiment without the repulsive interaction can be extended up to multiple particles and dimensions [2]. And as the last project, the analytical value for only one non-interactive particle in one dimension is given as 0.5 a.u [3].

## 2.2 Representing the wave function with a neural network.

To study a system as described above, one would need a decent ansatz for a trial wave function. There exists many possible ways of representing a wave function for such a system, but in this project we will attempt to design our trial wave function around a neural network.

### 2.2.1 Restricted Boltzmann Machine (RBM)

In this project we will be utilizing a restricted Boltzmann machine, which can also be interpreted as a stochastic recurrent neural network or just a generative neural network model. It is a two layer network where the first layer is called visible nodes and the second layer is called hidden nodes, and since there is no connection between the nodes in the same layer we will call it restricted. It is possible to implement several RBMs to build a deep belief network, however, in this project we will be focusing on only one hidden layer while varying the number of hidden nodes in this layer.

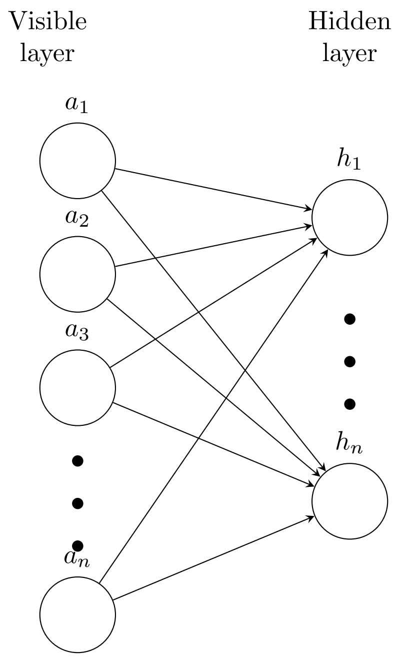


Figure 1: A schematic drawing of a restricted Boltzmann machine with one visible layer and one hidden layer. The lines between the nodes are representing interactions through a weight matrix. The amount of nodes in the visible layer corresponds to the number of particles times the number of dimensions, while the number of hidden nodes is arbitrary.

There are several extinctions of a deep neural network (DNN) and an RBM. One of the more important ones is the DNN's lack of ability to model an underlying and unknown probability distribution, while an RBM can sample from a *probability distribution* by learning its parametric model. Thus, the network does not produce an output directly, but a probability distribution from which we can generate an output. In our case this distribution corresponds to the wave function and the output we wish to generate are the positions taken by the particles in our system.

The joint probability distribution is defined as

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

where $Z$ is the partition function/normalization constant

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

Here $E$ is known as the energy of a configuration of the nodes which gives the specifics of the relation between the hidden and visible nodes, while $T_0$ we will ignore and set as 1.

### 2.2.2 Gaussian-Binary RBM

There are a number of different RBMs, with the most common one has a transparent nickname as "binary-binary", which means the hidden and visible nodes only take on binary values. Since our model samples from a joint probability distribution of positions, we would like them to be continuous. Thus, we will be utilizing an RBM named "Gaussian-binary", with the energy of the configuration of nodes defined as

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

If $\sigma_i = \sigma$ then

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

Here $\mathbf{X}$ are the visible nodes (the position coordinates), $\mathbf{H}$ are the hidden nodes, $\mathbf{a}$ are the visible biases, $\mathbf{b}$ are the hidden biases and $\mathbf{W}$ is a matrix containing the weights characterizing the connection of each visible node to a hidden node. The dimension of visible nodes and its bias is given as the number of particles ($P$) times the number of dimensions ($D$), $M = P \times D$, while the number of hidden nodes ($N$) can be chosen freely, however resulting in a weight matrix of size $M \times N$.

### 2.2.3 The Wave Function

To find the marginal probability $F_{rbm}(X)$ we set:

$$
\begin{aligned}
F_{rbm}(\mathbf{X}) &= \sum_{\mathbf{h}} F_{rbm}(\mathbf{X},\mathbf{h}) && (6)\\
&= \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{X},\mathbf{h})} && (7)
\end{aligned}
$$

This is used to represent the wave function,

$$
\begin{aligned}
\Psi(\mathbf{X}) &= F_{rbm}(\mathbf{X}) && (8)\\
&= \frac{1}{Z}\sum_{\{h_j\}} e^{-E(\mathbf{X},\mathbf{h})} && (9)\\
&= \frac{1}{Z}\sum_{\{h_j\}} e^{-\sum_i^M \frac{(X_i-a_i)^2}{2\sigma^2}+\sum_j^N b_jh_j+\sum_{i,j}^{M,N}\frac{X_iw_{ij}h_j}{\sigma^2}} && (10)\\
&= \frac{1}{Z}e^{-\sum_i^M \frac{(X_i-a_i)^2}{2\sigma^2}}\prod_j^N(1+e^{b_j+\sum_i^M \frac{X_iw_{ij}}{\sigma^2}}). && (11)
\end{aligned}
$$

## 2.3 Scaling

The practicalities of using scaled units are very much needed in a system where we use several different constants and measurements. Given our Hamiltonian

$$
\begin{aligned}
H &= \sum_i\Big(-\frac{\hbar^2}{2m}\nabla_i^2+V_{ext}\Big)+\sum_{i<j}V_{int}\\
&= \frac{1}{2}\sum_i\Big(-\frac{\hbar^2}{m}\nabla_i^2+(m\omega^2r_i^2)\Big)+\sum_{i<j}\frac{1}{r_{ij}}
\end{aligned}
$$

we introduce energy in units of $\hbar\omega$, such that we get a new Hamiltonian $H' = H/\hbar\omega$. In addition, we scale every dimension such that we get for an arbitrary dimension

$$r_i' = r_i\cdot\Big(\frac{m\omega}{\hbar}\Big)^{1/2}$$

By applying the new units will result in the new Hamiltonian

$$
\begin{aligned}
H' &= \frac{1}{2}\sum_i\Big(\frac{\hbar}{m\omega}\nabla_i^2+\frac{m\omega^2}{\hbar\omega}(r_i'^2)\frac{\hbar}{m\omega}\Big)+\sum_{i<j}\frac{1}{r_{ij}}\\
&= \frac{1}{2}\sum_i\Big(\nabla_i'^2+r_i'^2\Big)+\sum_{i<j}\frac{1}{r_{ij}}
\end{aligned}
$$

The same result can also be obtained by a slightly different method, which is setting $m=\hbar=\omega=e=m_e=c=1$. It is known as using dimensionless units and natural units, and is well known in the computational science linguistic and culture.

## 2.4 Variational Principle

We want to find the ground state energy of our system for a given trial wavefunction $\Psi_T$.

The variational principle states that for a given trial wavefunction $\Psi_T$, the expectation value of the Hamiltonian, $H$ will be an upper bound to the ground state energy, $E_0$ of the system

$$E_0 \leq E = \frac{\int \Psi_T^* H \Psi_T d\boldsymbol{r}}{\int \Psi_T^* \Psi_T d\boldsymbol{r}} \tag{12}$$

The complicated part of the variational principle is finding the trial wavefunction that minimizes the energy. However, by introducing variational parameters, we can minimize the energy with respect to these parameters.

## 2.5 Monte Carlo

By utilising the variational principle we can define a quantity called the local energy $E_L$, given as

$$E_L = \frac{1}{\Psi_T} H \Psi_T \tag{13}$$

such that we can use equation 12 and rewrite the energy to

$$E = \int |\Psi_T^2| E_L d\boldsymbol{r} \tag{14}$$

This integral can be solved by using Monte Carlo integration. In simple terms this means that we are approximating an integral to a sum.

$$\langle x \rangle = \int x p(x) dx \approx \frac{1}{N} \sum_{i=1}^{N} x_i p(x_i)$$

Where $\langle x \rangle$ is the sample mean, $x$ is a random variable, $p(x)$ is a probability distribution function (PDF) and $N$ is the number of Monte Carlo samples that is chosen to approximate the integral. By taking advantage of the law of large numbers, it is possible to show that the approximated integral goes toward the true value, when $N$ is very large.

$$\langle x \rangle = \int x p(x) dx = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} x_i p(x_i)$$

In our case this translates into $E_L = x$, and our probability distribution function is the wavefunction squared, $|\Psi_T|^2 = p(x)$.

$$\langle E \rangle = \int E_L |\Psi_T^2| d\boldsymbol{r} \approx \frac{1}{N} \sum_{i=1}^{N} E_{L,i} |\Psi_T(\boldsymbol{r}_i)^2|$$

## 2.6 Local Energy

Like it was mentioned earlier, one can calculate the energy of the system with a Monte Carlo simulation. To accomplish that, we need to calculate the local energy, $E_L$. Since the local energy is

$$\begin{aligned} E_L &= \frac{1}{\Psi_T} H \Psi_T \\ &= \frac{1}{\Psi_T} \sum_i^N \left( \frac{-\hbar^2}{2m} \nabla_i^2 + V_{ext}(\mathbf{r}_i) \right) \Psi_T + \sum_{i<j}^N V_{int}(\mathbf{r}_i, \mathbf{r}_j) \Psi_T, \end{aligned}$$

the most heaviest calculation will be of the second derivative of the trial wavefunction. However, there exists a trick which uses the relation of the derivative of the wavefunction with the logarithm,

$$\frac{1}{\Psi_T} \frac{\partial \Psi_T}{\partial \alpha_i} = \frac{\partial \ln \Psi_T}{\partial \alpha_i}$$

which will result in the local energy

$$E_L = \frac{1}{2} \sum_m^M (-(\frac{\partial}{\partial v_m} \ln \Psi)^2 - \frac{\partial^2}{\partial v_m^2} \ln \Psi + \omega^2 v_m^2) + \sum_{p<q} \frac{1}{r_{pq}},$$

where we sum over every particle's coordinate. The full derivation can be found in appendix A.

## 2.7 Statistical Analysis

Other quantities that are important for Monte Carlo calculations in addition to the calculation of the integral $\langle E \rangle$ are the variance $\sigma_E^2$ and the standard deviation $\sigma_E$.

The variance $\sigma_E^2$ is defined as

$$\sigma_E^2 = \frac{1}{N} \sum_{i=1}^N (E_{L,i} - \langle E \rangle)^2$$

and is a measure of the error in the calculation of the energy.

The standard deviation, on the other hand, is defined as the square root of the variance.

$$\sigma_E = \sqrt{\sigma_E^2}$$

Assuming that the above is a result acquired for a fixed value of N to be a measurement, that would make it possible to recalculate this for different measurements. Each such measurement can be denoted $l$ and in every measurement we have $[E_{l,1}, E_{l,2}, \ldots, E_{l,N}]$ data points, that produces a set of averages $\langle E_l \rangle$ which is defined as

$$\langle E_l \rangle = \frac{1}{N} \sum_{i=1}^N E_{l,i}$$

If we repeat the measurements $M$ times, the mean of all the measurements will be

$$\langle E_M \rangle = \frac{1}{M} \sum_{l=1}^{M} \langle E_l \rangle$$

we can then define the total variance of these series of measurements as

$$\sigma_M^2 = \frac{1}{M} \sum_{l=1}^{M} (\langle E_l \rangle - \langle E_M \rangle)^2$$

and this can be brought down into

$$\sigma_M^2 = \frac{\sigma_E^2}{N} + \text{covariance} \tag{15}$$

where $\sigma_E^2$ is the sample variance over all the different measurements, defined as

$$\sigma_E^2 = \frac{1}{MN^2} \sum_{l=1}^{M} \sum_{i=1}^{N} (E_{l,i} - \langle E_M \rangle)^2 \tag{16}$$

and the covariance is the correlation between the data points, and it is defined as

$$\text{covariance} = \frac{2}{MN^2} \sum_{l=1}^{M} \sum_{i<j}^{N} (E_{l,i} - \langle E_M \rangle)(E_{l,j} - \langle E_M \rangle) \tag{17}$$

However, when there are no correlation in the data set, that makes life easier, because then the variance can be roughly approximated to

$$\sigma_M^2 \approx \frac{1}{N} \Big( \langle E^2 \rangle - \langle E \rangle^2 \Big) = \frac{\sigma_E^2}{N} \tag{18}$$

where we don't have to evaluate the covariance which is a double sum. Then it is easy to see that the standard deviation is

$$\sigma_M = \sqrt{\sigma_M^2} = \sqrt{\frac{\sigma_E^2}{N}} = \frac{\sigma_E}{\sqrt{N}}.$$

From this it is possible to see that the standard deviation is proportional to the inverse square root of the amount of measurements

$$\sigma_M \sim \frac{1}{\sqrt{N}}.$$

In such a situation where we sample infinitely many times, $N \to \infty$, the standard deviation would go towards zero and the calculation of the energy would in theory converge to the exact answer.

In the case where there exists correlation in the data set, the covariance needs to be calculated, or else the estimation of the variance will be highly underestimated, which leads to an over-optimistic estimation of the uncertainty $\sigma_E$.

## 2.8 Markov Chains

When studying a physical system which evolves towards equilibrium and we want to understand how the system evolves with time, then Markov chains is the preferred way of accomplishing this.

Markov processes is a random walk with a selected probability for making a move. The new move is independent of the previous history of the system. The Markov process is used repeatedly in Monte Carlo simulations in order to generate new random states. The reason for choosing a Markov process is that when it is run for a long enough time, starting with a random state, we will eventually reach the most likely state of the system.

For the Markov processes, the time development of a systems PDF, is defined as

$$w_i(t+1) = \sum_j W_{i\to j} w_j(t)$$

or in matrix form

$$\hat{w}(t+1) = \hat{W}\hat{w}(t)$$

where $w_i(t)$ is the time dependent PDF of the state $i$, $W_{i\to j}$ is the transition probability of going from state $j$ to $i$ and both quantities are normalized. If we have that $\hat{w}(t=\infty) = \hat{W}\hat{w}(t=\infty)$, we say that we have reached the most likely state of the system, the so-called steady state also referred to the equilibrium state.

By rewriting the transition probability $W_{i\to j}$ into a product of two probabilities

$$W_{i\to j} = T_{i\to j} A_{i\to j}$$

where $T_{i\to j}$ is the probability for making the transition from state $j$ to state $i$ (suggesting moves), and $A_{i\to j}$ is the probability for accepting the proposed move from state $j$ to the state $i$ (accepting or rejecting moves).

We can then express the time development $w_i$ as

$$w_i(t+1) = \sum_j \Big[ w_j(t) T_{i\to j} A_{i\to j} + w_i(t) T_{j\to i} \big(1 - A_{j\to i}\big) \Big]$$

assuming that $T$ and $A$ are time-independent.

By utilizing that all the probabilities are normalised (as mentioned above), one can rewrite the equation above as

$$w_i(t+1) = w_i(t) + \sum_j \Big[ w_j(t) T_{i\to j} A_{i\to j} - w_i(t) T_{j\to i} A_{j\to i} \Big]$$

$$w_i(t+1) - w_i(t) = \Big[ w_j(t) T_{i\to j} A_{i\to j} - w_i(t) T_{j\to i} A_{j\to i} \Big]$$

In the limit $t \to \infty$, it can be shown that

$$w_i(t+1) = w_i \quad \text{and} \quad w_i(t) = w_i$$

which leads to

$$\sum_j w_j(t) T_{i\to j} A_{i\to j} = \sum_j w_i(t) T_{j\to i} A_{j\to i}$$

However, the condition that the rates should equal each other is in general not sufficient to guarantee that we, after many simulations, generate the correct distribution. We may risk to end up with so-called cyclic solutions. To avoid this issue, one introduces an additional condition, namely that of detailed balance [4].

$$\begin{aligned} W_{i\to j} w_j &= W_{j\to i} w_i \\ w_j(t) T_{i\to j} A_{i\to j} &= w_i(t) T_{j\to i} A_{j\to i} \\ \frac{T_{i\to j} A_{i\to j}}{T_{j\to i} A_{j\to i}} &= \frac{w_i}{w_j} \end{aligned}$$

## 3 Methods

### 3.1 Restricted Boltzmann machine

The neural network of this project is a restricted Boltzmann machine, which is in the category of reinforcement learning. The algorithm in simple terms is as follows.

**Restricted Boltzmann Machine Algorithm**

1. Initialize nodes, weights and biases according to a probability distribution.
2. Run the Variational Monte Carlo Algorithm.
3. Calculate gradients and update weights and biases according to a given learning rate.
4. Repeat step 2-3 until convergence has been reached.

One does not simply need to wait until convergence to stop as this might be truly computationally heavy, and one possibility is to stop after a well-tested number of iterations.

## 3.2 Variational Monte Carlo

In this project we will use the widely known Monte Carlo algorithm to simulate random walks in a given volume space. The procedure is as follows.

**Variational Monte Carlo Algorithm**

1. Initialise the system at $N$ random position $\mathbf{r} = (\mathbf{r}_1, .., \mathbf{r}_N)$. These points will act as our particles and will serve as the basis of our system.

   We also have to fix the number of Monte Carlo steps.

2. Initialise the energy and variance and start a Monte Carlo calculation.

   I Choose a random particle and change its configuration.

   II Use the Metropolis-Hastings sampling algorithm to accept or reject this new configuration, or use Gibbs sampling algorithm.

   III If accepted, we update to the new configuration and update the averages.

3. Repeat step I - III until convergence of energy.

4. Finish and compute the final averages.

The direction and amount of change of a particle's configuration is chosen by random, and thus every change will be independent from each other.

## 3.3 Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm takes advantage of the Markov processes and its task is to sample a normalised probability distribution by a stochastic process. In this project, we do two different implementations where both are dependent on the acceptance criteria below.

$$\frac{A_{i\to j}}{A_{j\to i}} = \frac{w_i T_{j\to i}}{w_j T_{i\to j}} \tag{19}$$

### 3.3.1 Brute Force Metropolis

The Brute Force Metropolis algorithm is a straight forward implementation of the Metropolis algorithm. This algorithm makes the assumption that the probability of jumping from state $i$ to $j$ is the same as jumping from $j$ to $i$, $T(j \to i) = T(i \to j)$, which leads to

$$\frac{A_{i\to j}}{A_{j\to i}} = \frac{w_i}{w_j}$$

Now we can calculate the probability $|\Psi_T(\mathbf{r})|^2$ given the new configuration, which is governed by a variable step length, and we find the ratio between the new and old position.

$$q(\mathbf{r}_i, \mathbf{r}_{i+1}) = \frac{w(\mathbf{r}_{i+1})}{w(\mathbf{r}_i)} = \frac{|\Psi_T(\mathbf{r}_{i+1})|^2}{|\Psi_T(\mathbf{r}_i)|^2}$$

If the ratio $q$ is bigger than 1, the Metropolis will accept the move, while if its less than 1 it will accept the move if its bigger than a random generated number $r$ from a uniform distribution in the interval $[0, 1]$. However, if it is lower, it will reject the move.

$$\text{New configuration} = \begin{cases} \text{Accept}, & \text{if } q > r \\ \text{Reject}, & \text{if } q \leq r \end{cases}$$

### 3.3.2 Importance sampling

For a diffusion process characterized by a time-dependent probability density for one particle, the Fokker-Planck equation reads

$$\frac{\partial \mathbf{P}}{\partial t} = D\nabla\Big(\nabla - \mathbf{F}\Big)\mathbf{P}$$

where $\mathbf{F}$ is the drift force given in equation 34 in the appendix, while D is the diffusion coefficient, which is 0.5 in natural units for our case. The new positions in space are found by the Langevin equation using Euler's method,

$$\frac{\partial \mathbf{r}}{\partial t} = D\mathbf{F} + \boldsymbol{\eta}$$

where $\boldsymbol{\eta}$ is a random variable for every dimension. This yields a new position

$$\mathbf{r}_{i+1} = \mathbf{r}_i + D\mathbf{F}\Delta t + \boldsymbol{\xi}\sqrt{\Delta t}$$

where $\boldsymbol{\xi}$ is a random variable from a normal distribution in the interval $[0, 1]$ for each dimension and $\Delta t$ is an adjustable time step. The drift force (also known as the quantum force) gives which direction the particles are moving towards, which the new suggestion of position are dependent on. This can be considered an improvement compared to the brute force algorithm, where a new proposed configuration is not dependent on where the particles are and thus will have the same probability of moving in every direction.

The Fokker-Planck equation yields a transition probability given by Green's function

$$G(\mathbf{r}_{i+1}, \mathbf{r}_i, \Delta t) = \frac{1}{(4\pi D\Delta t)^{3/2}} \exp\Big(-(\mathbf{r}_{i+1} - \mathbf{r}_i - D\Delta t\mathbf{F})^2/4D\Delta t\Big) = T_{i\to j}$$

which in turn means that we get a new acceptance criteria.

$$q(\mathbf{r}_{i+1}, \mathbf{r}_i, \Delta t) = \frac{G(\mathbf{r}_i, \mathbf{r}_{i+1}, \Delta t)|\Psi_T(\mathbf{r}_{i+1})|^2}{G(\mathbf{r}_{i+1}, \mathbf{r}_i, \Delta t)|\Psi_T(\mathbf{r}_i)|^2}$$

## 3.4 Gibbs Sampling

A third way to sample is by assuming that the wavefunction is positive definite, such that we can use the RBM to represent the squared wavefunction. This will result in a probability, and will make it possible to sample from the model using Gibbs sampling.

$$
\begin{aligned}
&|\Psi(\mathbf{X})|^2 = F_{rbm}(\mathbf{X}) \\
&\Rightarrow \Psi(\mathbf{X}) = \frac{1}{\sqrt{Z}} e^{-\sum_i^M \frac{(X_i - a_i)^2}{4\sigma^2}} \prod_j^N \sqrt{1 + e^{b_j + \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}}
\end{aligned}
$$

We will be sampling from the joint probability of $\mathbf{x}$ and $\mathbf{h}$. The samples will then model the probability distribution $|\Psi(\mathbf{x})|^2$. The updated samples are generated according to the conditional probabilities $P(X_i|\mathbf{h})$ and $P(H_j|\mathbf{x})$ respectively and accepted with the probability of 1. In mathematical terms, this means that

$$
P(H_j = 1|\mathbf{x}) = \frac{1}{1 + e^{-b_j - \frac{\mathbf{x}^T \mathbf{w}_{\cdot j}}{\sigma^2}}}
$$

and

$$
P(H_j = 0|\mathbf{x}) = \frac{1}{1 + e^{b_j + \frac{\mathbf{x}^T \mathbf{w}_{\cdot j}}{\sigma^2}}}.
$$

## 3.5 Minimization methods

There exists a legion of challenges in the computational science world when it comes to reducing computational time and effort. How to efficiently find optimal parameters for different systems are undoubtedly one of the, if not the most, demanded solution to scientists in the field of this study. Gradient descent is one of the methods that can cause salvation for scientists for some systems, while just as easily exponentially increase the frustration for others since the range of parameters can range from a few to several thousands in complicated systems. In our system we are fortunate enough to have only only a few variational parameters.

### 3.5.1 Gradient Descent

The basic idea behind is that a function decreases fastest in the negative direction of the gradient of the same function. We want to minimise the ground state energy with respect to the weights and biases as the variational parameters $\alpha_i$ by using its former value to suggest the next value $\alpha_i^+$, as shown in the iterative formula

$$
\alpha_i^+ = \alpha_i - \lambda \frac{d\langle E(\alpha_i)\rangle}{d\alpha_i}
$$

where $\alpha_i = a_1, ..., a_M, b_1, ..., b_N, w_{11}, ..., w_{MN}$ and $\lambda > 0$ is the step length, which is also known as learning rate in the machine learning world. We already know the expression for the local energy as

$$\langle E(\alpha_i)\rangle = \frac{\langle \Psi_T(\alpha_i)|H|\Psi_T(\alpha_i)\rangle}{\langle \Psi_T(\alpha_i)|\Psi_T(\alpha_i)\rangle}$$

and then we differentiate with respect to $\alpha_i$ by using the chain rule and the hermiticity of the Hamiltonian and arrive at the expression

$$\frac{dE(\alpha_i)}{d\alpha_i} = 2\Big\langle E_L(\alpha_i)\frac{1}{\Psi_T(\alpha_i)}\frac{d\Psi_T}{d\alpha_i}\Big\rangle - 2\big\langle E_L(\alpha_i)\big\rangle\Big\langle \frac{1}{\Psi_T(\alpha_i)}\frac{d\Psi_T}{d\alpha_i}\Big\rangle.$$

We need to do this for all the variational parameters, which are $a, b$ and $w$. The gradients to compute are

$$\frac{\partial}{\partial a_m}\ln\Psi = \frac{1}{\sigma^2}(X_m - a_m) \tag{20}$$

$$\frac{\partial}{\partial b_n}\ln\Psi = \frac{1}{e^{-b_n-\frac{1}{\sigma^2}\sum_i^M X_i w_{in}}+1} \tag{21}$$

$$\frac{\partial}{\partial w_{mn}}\ln\Psi = \frac{X_m}{\sigma^2(e^{-b_n-\frac{1}{\sigma^2}\sum_i^M X_i w_{in}}+1)} \tag{22}$$

and if $\Psi = \sqrt{F_{rbm}}$ then the difference is only that we multiply by a factor of $\frac{1}{2}$.

As this is an iterative parameter tuning, one must also take an iterative approach when to stop iterating. We stop iterating as the difference in the calculated energies are less than a given tolerance, or after a given number of iterations as the worlds most powerful machineries are far out of reach for this humble project.

Ideally, we would like the sequence to converge towards the global minimum, however, we do not know if we have encountered a global or local minimum. If the function that is being minimized is convex, we know that every local minimum must also be a global minimum. In addition, gradient descent is deterministic, which means that it will converge to a local minima, if we do not have a good initial guess. Another drawback is that the algorithm is very sensitive to the step length $\lambda$. If it is too low, the algorithm will converge very slowly and the computational effort will be too big to defend its use, while if it is too large it might not find the minima at all.

## 3.6 Statistical analysis methods

Monte Carlo is a method which makes a great foundation for sampling a vast amount of data with a strong correlation between the data points for the interacting system. Distressingly, this will make the statistical analysis and the calculation of the covariance in equation 17 impractical considering the double summation. Consequently, the tools that will be equipped for post analysis must meet this requirement level. As luck would have it, there exists a method named blocking which can be of aid in these times.

### 3.6.1 Blocking

Blocking is a technique which gets increasingly accurate with a larger amount of observable data. With the stationary time series $\mathbf{X} = \{x_1, .., x_n\}$, which is our data set where it is assumed that $n = 2^d$ for some integer $d > 1$, the idea behind blocking is a blocking transformation, where we take the mean of subsequent pair of elements from the data set and form a new vector. This is done recursively as

$$\mathbf{X}'_i = \frac{1}{2}\Big(\mathbf{X}_{2i} + \mathbf{X}_{2i+1}\Big)$$

where $i$ is the iteration of the elements in the previous vector. Each of the new vectors are what we would call a block, and the transformed data set will have $n' = n/2$ data points. Then we calculate the variance and covariance for every new time series until we have gone through all of the data. When this process is repeated, the total variance will stagnate until it is not increased any more. Then we have reached our destination, where the blocks are no longer correlated and we can use the standard deviation as a good estimate for the error. For more in-depth derivations we can recommend [5].

# 4 Implementation

Programs used in this project can be found on our github repository [6]. Here it is included a Jupyter Notebook for easy and reproducable results, and the technicalities included in visualizing data from the variational Monte Carlo simulations. The code skeleton was initialized by Morten Ledum [7], and in addition the programs are inspired by the codes found in the course's github repository [4]. For the convenience of a linear algebra library written in C++, it has been implemented a Boltzmann machine based on the library Armadillo [8] [9].

# 5 Results

## 5.1 Non-repulsive case

### 5.1.1 The search for optimal parameters

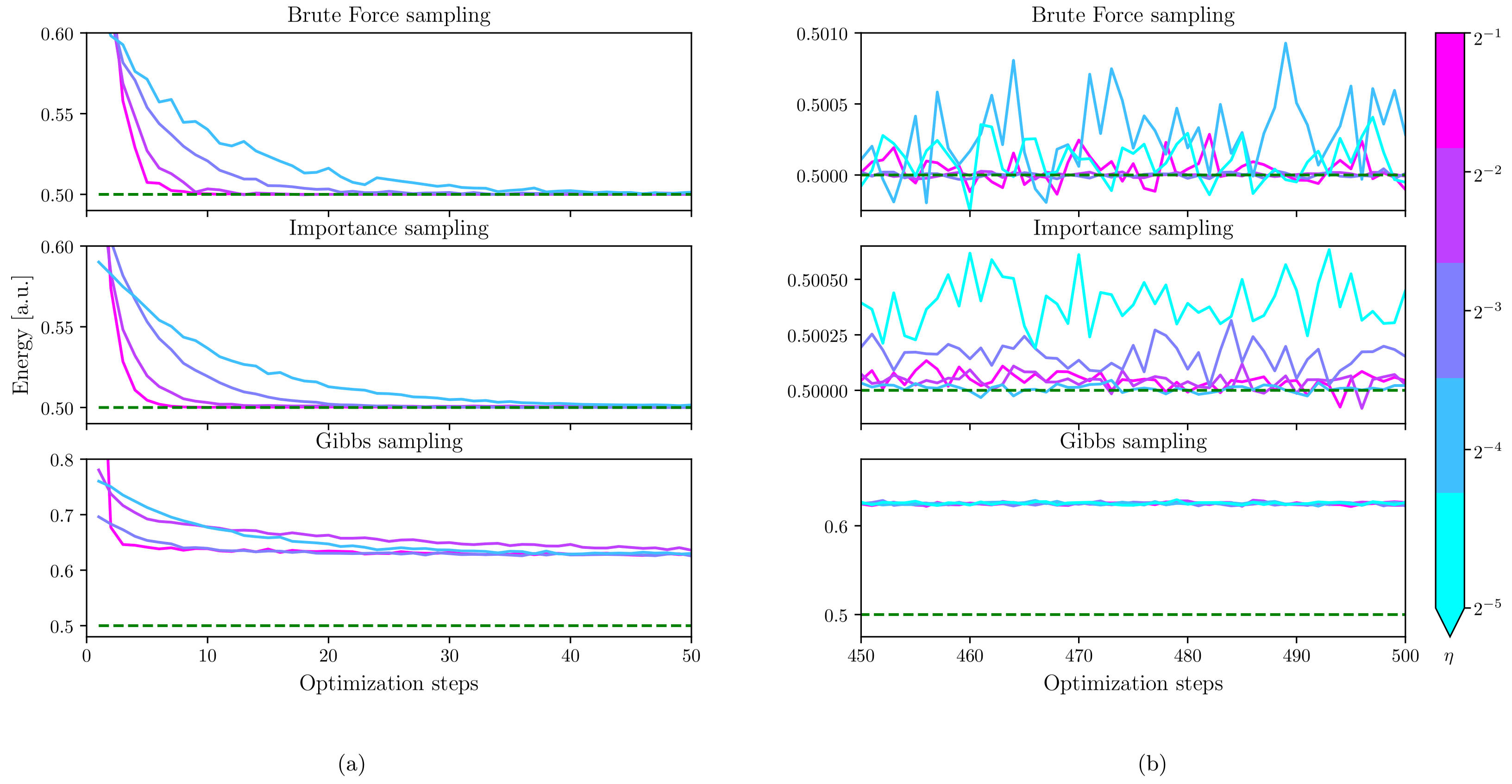
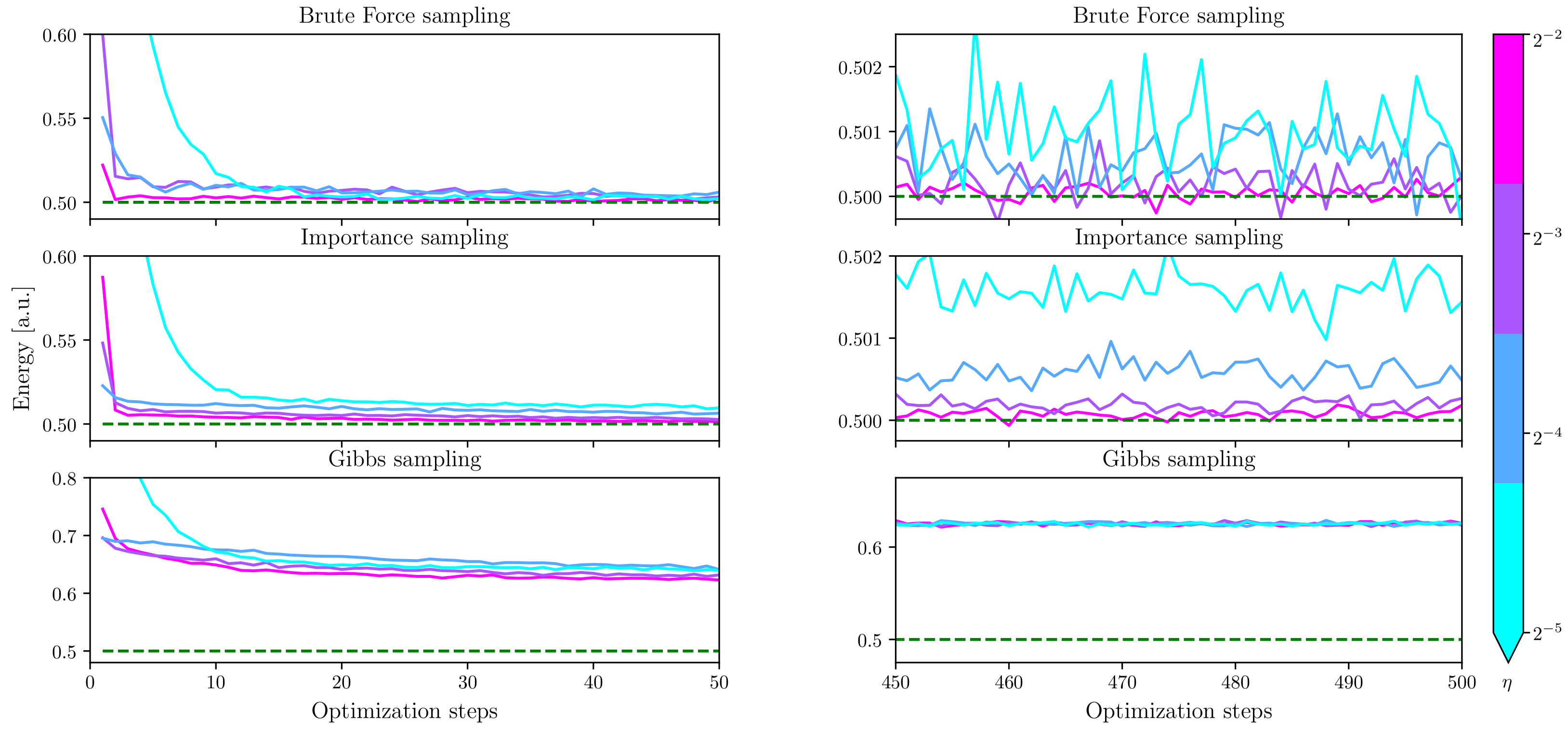
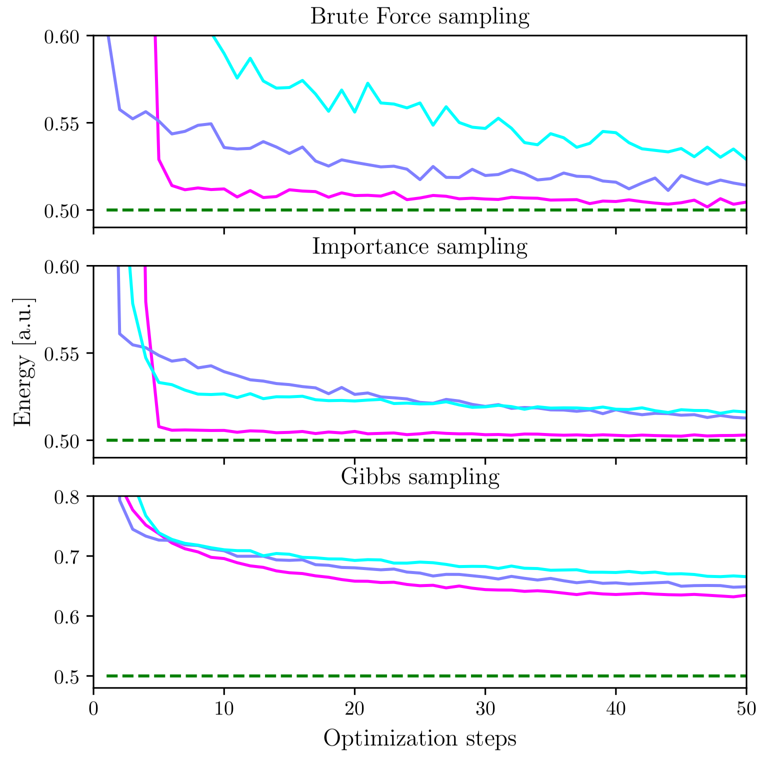


Figure 2: This figure shows how the RBM with 2 hidden nodes is converging towards the energy minimum for (a) the first 50 and (b) the last 50 optimization iterations for three different sampling methods for one particle in one dimension. Another parameter of interest is the number of Monte Carlo cycles, which is set to $2^{17}$. The green dotted line is the analytical solution to the energy.

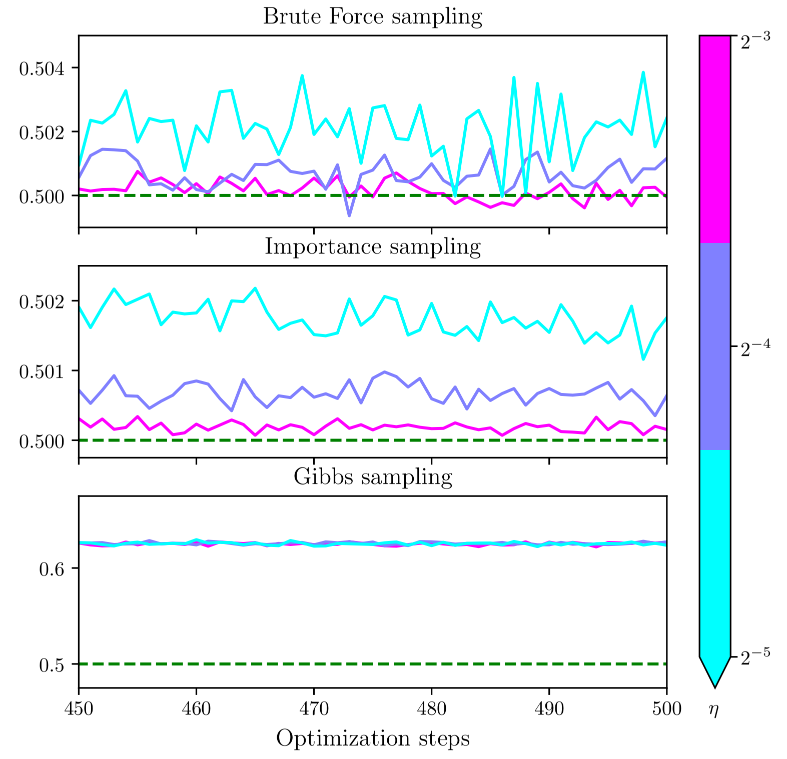


(a) (b)

Figure 3: This figure shows how the RBM with 10 hidden nodes is converging towards the energy minimum for (a) the first 50 and (b) the last 50 optimization iterations for three different sampling methods for one particle in one dimension. Another parameter of interest is the number of Monte Carlo cycles, which is set to $2^{17}$. The green dotted line is the analytical solution to the energy.

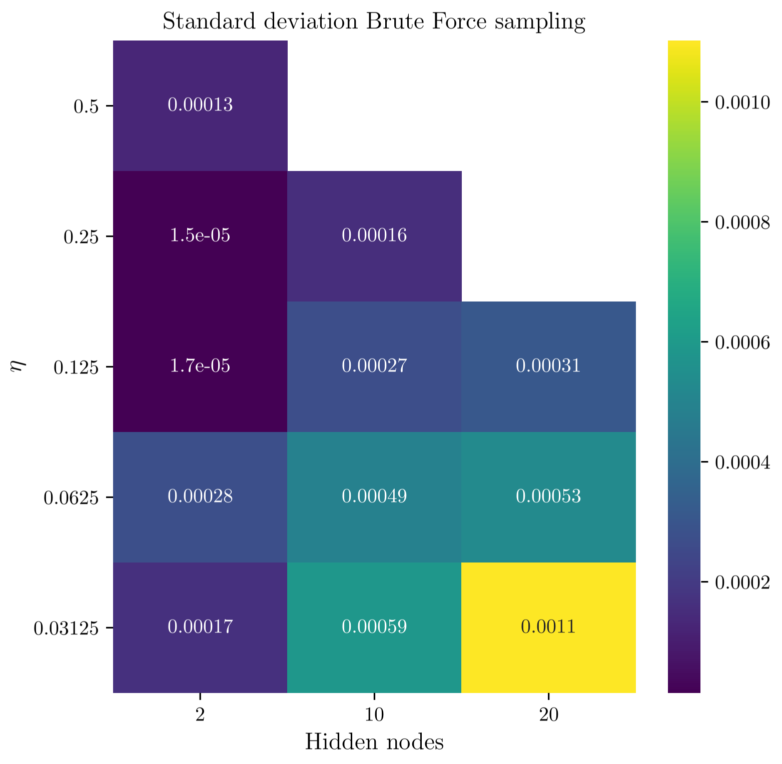


(a)

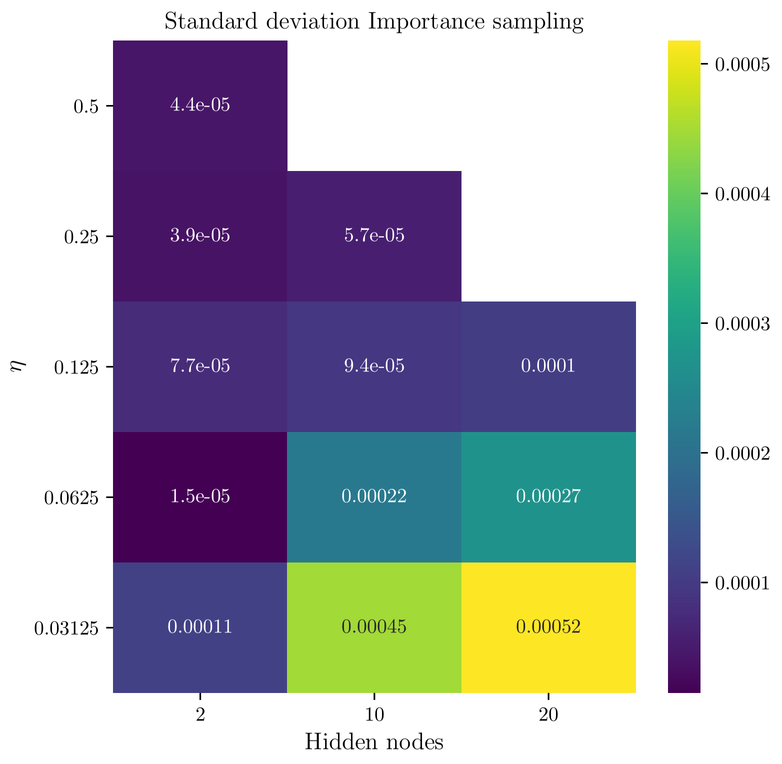


(b)

Figure 4: This figure shows how the RBM with 20 hidden nodes is converging towards the energy minimum for (a) the first 50 and (b) the last 50 optimization iterations for three different sampling methods for one particle in one dimension. Another parameter of interest is the number of Monte Carlo cycles, which is set to $2^{17}$. The green dotted line is the analytical solution to the energy.



(a)



(b)

Figure 5: Two heatmaps visualizing the standard deviation of the last 100 optimization iterations of (a) Brute Force sampling and (b) Importance sampling from the same data as figures 2, 3 and 4. The lacking values represents an unstable $\eta$ for the given number of hidden nodes.

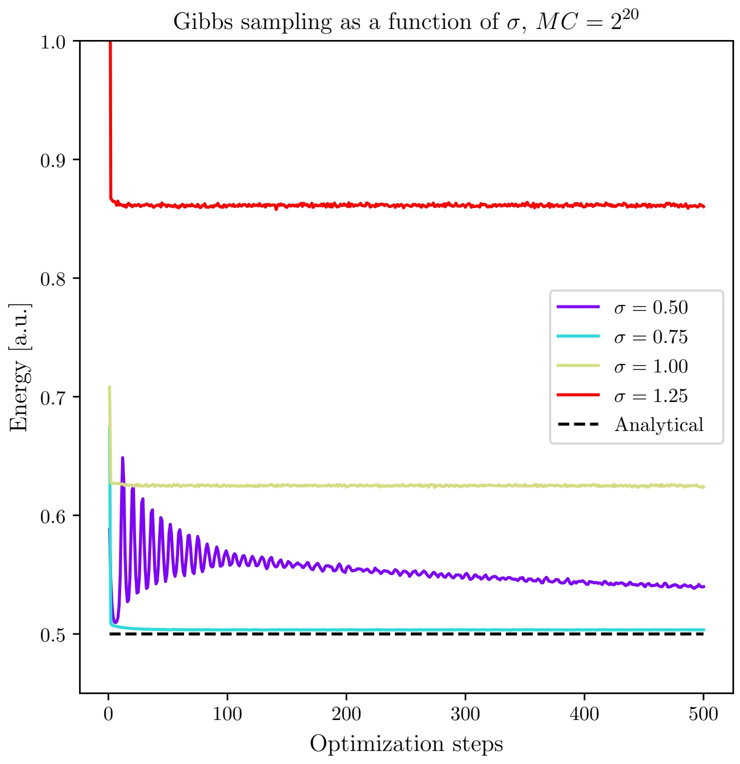


Figure 6: A figure visualizing four optimization runs for different values of $\sigma$ using Gibbs sampling for one particle in one dimension. The black dotted line is the analytical solution to the energy.

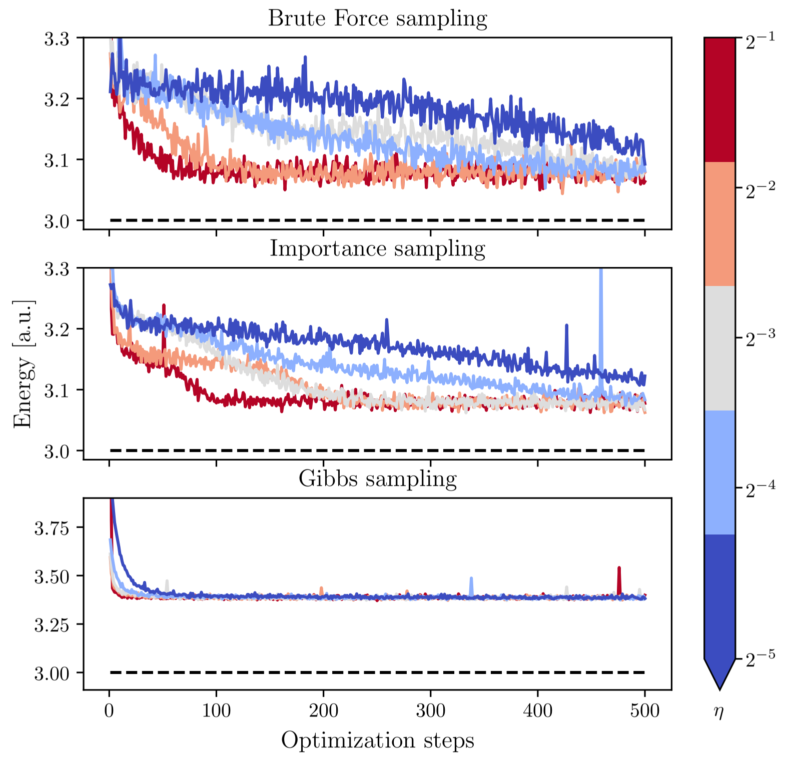
### 5.1.2 Putting the best parameters to the test

| | **Brute Force** | **Importance sampling** | **Gibbs sampling** |
|---|---|---|---|
| Hidden nodes | 2 | 2 | 2 |
| Learning rate | $2^{-2}$ | $2^{-4}$ | $2^{-2}$ |
| Particles | 1 | 1 | 1 |
| Dimensions | 1 | 1 | 1 |
| Mean | 0.50000 | 0.50004 | 0.50354 |
| Standard deviation | $8.21 \cdot 10^{-8}$ | $2.00 \cdot 10^{-6}$ | $1.90 \cdot 10^{-5}$ |
| Variance | $6.75 \cdot 10^{-14}$ | $5.00 \cdot 10^{-12}$ | $8.22 \cdot 10^{-10}$ |

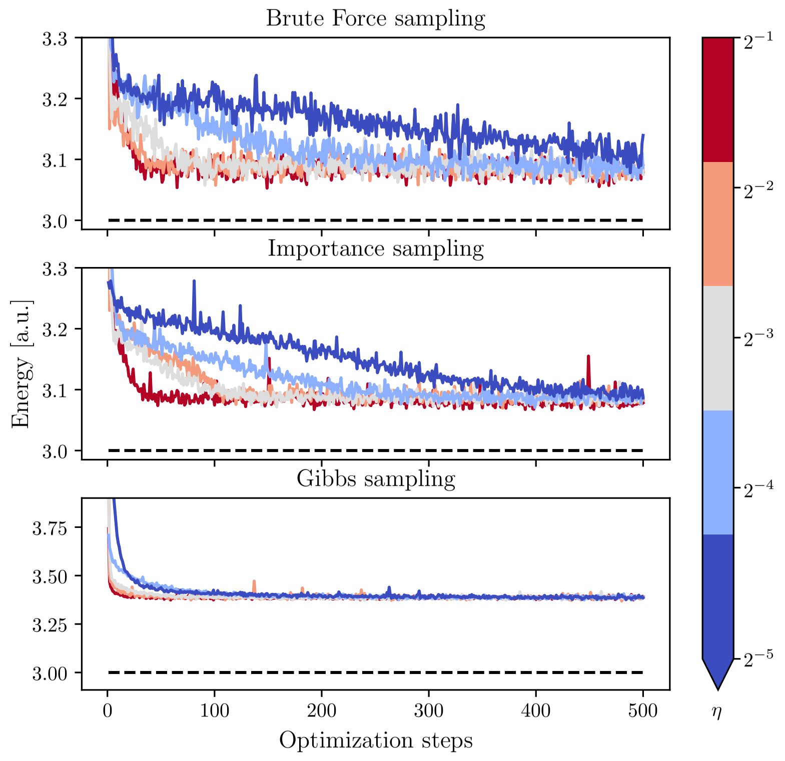
Table 1: Statistical analysis of the optimization process of $2^{20}$ MC cycles and the 500th optimization cycle by using the best parameters found in the previous section. The statistical tool of choice is blocking.

## 5.2 Interaction

### 5.2.1 The search for optimal parameters

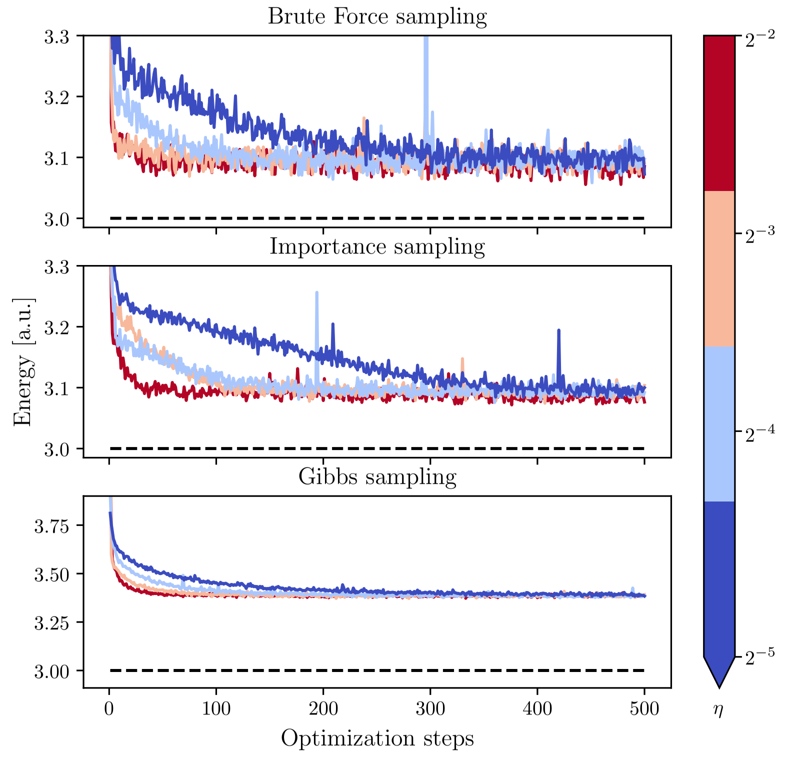


(a)

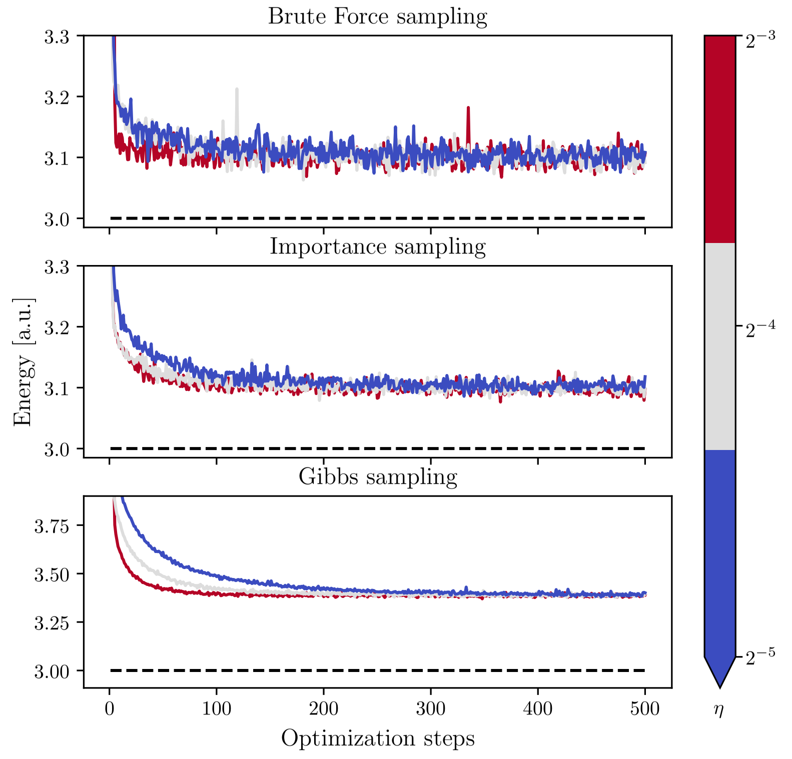


(b)

Figure 7: This figure shows how the RBM is converging towards the energy minimum for (a) 2 and (b) 5 hidden nodes during 500 optimization iterations for three different sampling methods for two particle in two dimension. The dotted black line is the exact energy $E = 3.0$ a.u. Another parameter of interest is the number of Monte Carlo cycles, which is set as $2^{17}$.

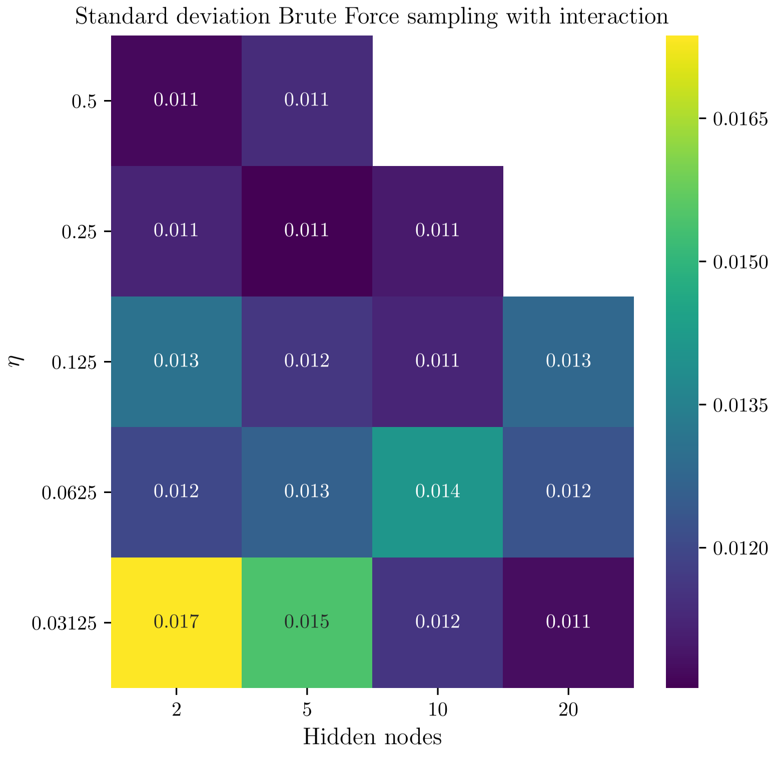


(a)

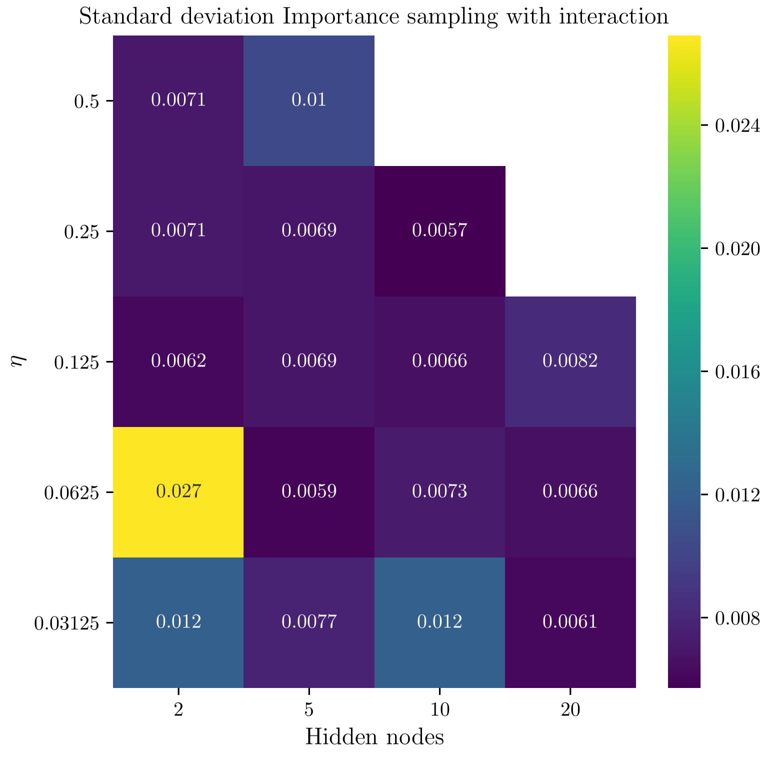


(b)

Figure 8: This figure shows how the RBM is converging towards the energy minimum for (a) 10 and (b) 20 hidden nodes during 500 optimization iterations for three different sampling methods for two particle in two dimension. The dotted black line is the exact energy $E = 3.0$ a.u. Another parameter of interest is the number of Monte Carlo cycles, which is set as $2^{17}$.



(a)



(b)

Figure 9: Two heatmaps visualizing the standard deviation of the last 100 optimization iterations of (a) Brute Force sampling and (b) Importance sampling from the same data as the four figures in figure **??**. The data includes interaction and two particles in two dimensions. The lacking values represents an unstable $\eta$ for the given number of hidden nodes.

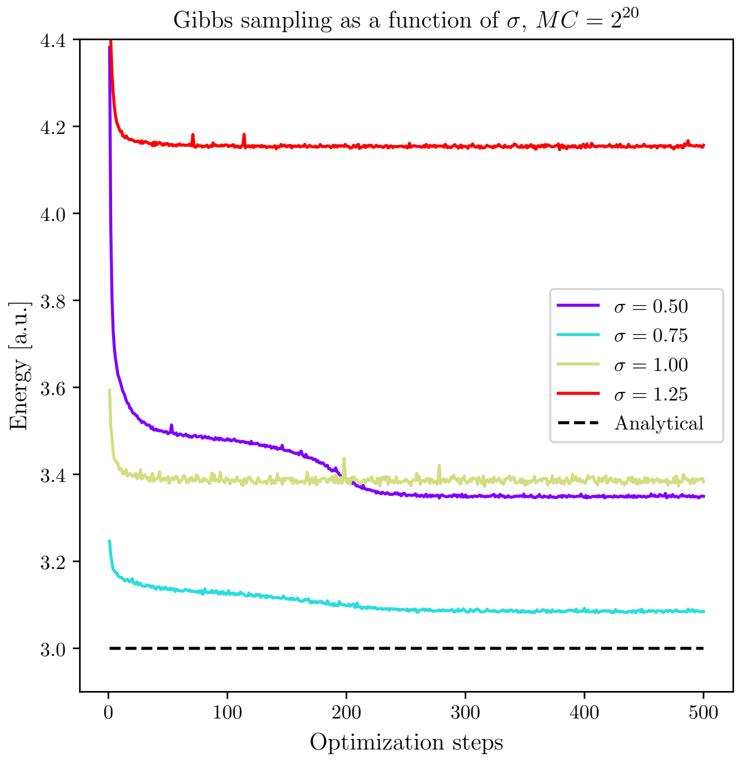


Figure 10: A figure visualizing four optimization runs for different values of $\sigma$ using Gibbs sampling with interaction and two particles in two dimensions. The black dotted line is the analytical solution to the energy.

### 5.2.2 Putting the best parameters to the test

| | **Brute Force** | **Importance sampling** | **Gibbs sampling** |
|---|---|---|---|
| Hidden nodes | 2 | 2 | 2 |
| Learning rate | $2^{-2}$ | $2^{-4}$ | $2^{-2}$ |
| Particles | 2 | 2 | 2 |
| Dimensions | 2 | 2 | 2 |
| Mean energy | 3.071 | 3.087 | 3.080 |
| Standard deviation | $1.58 \cdot 10^{-3}$ | $1.11 \cdot 10^{-3}$ | $5.81 \cdot 10^{-4}$ |
| Variance | $2.00 \cdot 10^{-6}$ | $1.00 \cdot 10^{-6}$ | $3.38 \cdot 10^{-7}$ |

Table 2: Statistical analysis of the optimization process of $2^{20}$ MC cycles and the 500th optimization cycle by using the best parameters found in the previous section. The statistical tool of choice is blocking.

## 6 Discussion

### 6.1 Excluding repulsive interaction

For the case without interaction, we observe different values for the learning value $\eta$ in figure 2 to 4. We observe that for the 50 first optimization iterations we see a slower convergence for smaller learning rates for all sampling methods. This is especially clear for figure 3a, and it is easy to see the trend following the larger number of hidden units for both 2a and 4a. This is also an expected result, since the learning rate is controlling how fast the weights and biases changes from one iteration to the next.

From the figures of the last 50 optimization iterations, we can also tell that the algorithm has adjusted its parameters well since both Brute Force and Importance sampling are getting close to the analytical answer in figure 2b, 3b and 4b. For Importance sampling it is possible to detect a general trend where a high learning rate results in higher accuracy, as the energies seems to oscillate in smaller amplitudes and closer to the analytical answer. This is also a trend that can be seen from the standard deviation of the last 100 optimization iterations given in figure 5b. In addition, it seems like a small number of hidden nodes is also an adequate parameter that shows great potential. However, one can also find a discrepancy from the trend which shows that the learning rate $\eta = 2^{-4}$ together with two hidden nodes seems to outperform the other set of parameters. This can be the result from an extraordinary good combination of the random initialization of weights, biases and positions together with the chosen learning rate and number of hidden nodes. The chosen parameters might show great prospects for this case, but that does not mean it will show the same great prospect for another task.

It is also possible to see the same trend for the Brute Force sampling method, where we achieve the smallest standard deviation of the last 100 optimization iterations with the combination of learning rate $\eta = 2^{-2}$ and 2 hidden nodes. But as we turn our attention towards the results from Gibbs sampling, we see that the learning rate has an effect in how fast the energy converges, but not in accuracy. This confirms that Gibbs sampling is quite independent on the choice of number of hidden nodes. Nonetheless, we see that by changing the value of the parameter $\sigma$, we can achieve a far greater result, as seen in figure 6. We observe that Gibbs sampling is very dependent on the choice of $\sigma$, and that the optimal choice in this case is $\sigma = 0.75$. The result from choosing $\sigma = 1.00$ is the same result as found in figures 2 to 4, and explains its lack of accuracy compared to the two other sampling methods. In addition, we find an oscillating optimization run for $\sigma = 0.50$. From the figure it is possible to tell that the program did not reach convergence within 500 optimization iterations, but since it has not reached convergence, we cannot rule out that it is a better or worse parameter than $\sigma = 0.75$. Despite that, finding an efficient program is of significant importance and thus we can rule it out of the parameter search.

From using the optimal parameters discussed above, we obtain the results with optimal parameters in table 1. Brute Force sampling achieves exact energy as mean from the last optimization cycle, with a variance close to numerical precision. Importance sampling renders a solid mean value close to Brute Force's value, while Gibbs sampling does not produce the same accuracy as the others.

## 6.2 Including repulsive interaction

For the interactive scenario of two particles in two dimensions, we observe a similar trend of the 20 different sets of parameters in figures 7-8 as we saw in the non-interactive case, that an increasing learning rate results in faster convergence. In addition, we observe another general trend that by increasing the number of hidden nodes will also result in a faster convergence in the beginning of the optimization process. This is clearly seen with learning rate $\eta = 2^{-5}$, as it does not reach convergence for 2 hidden nodes, but does indeed converge a lot faster for 20 hidden nodes.

As we saw for the non-interaction case, we find that the learning rate and number of hidden nodes does not affect the method Gibbs sampling, and we find once again that $\sigma = 0.75$ is the best option, as seen in figure 10. For Brute Force sampling we find several equally good sets of parameters from figure 9 as well as for Importance sampling, but from a computationally efficient perspective we will try to use the smallest amount of nodes and largest learning value while trying to maintain a fast convergence.

The system is of a higher complexity now than without interaction, and the RBM does not achieve the same precision as it did in the latter scenario. Brute Force sampling does perform best on this one as well as seen in figure 2, while Gibbs sampling is second best.

## 7 Conclusion

In this project, we have implemented three different sampling methods named Brute Force, Importance and Gibbs sampling to compete against each other in a restricted Boltzmann machine applied to a quantum many body problem. Observing the first issue of a single particle in one dimension, we find that Brute Force sampling, after an extensive parameter search, obtains the same mean energy as the analytical solution $E = 0.5$ a.u. with a standard deviation of $8.21 \cdot 10^{-8}$.

For the second part of the project, we included the interaction between two particles in two dimensions and found that we do not achieve the same precision as we did in the non-interacting scenario, however, the Brute Force method outperformed the other methods in this case as well, with mean energy $E = 3.071$ a.u. and standard deviation of $1.58 \cdot 10^{-3}$.

## 8 Future aspects

For future improvements we would like to get our hands on some proper machinery to run our RBM on multiple cores and for a longer amount of time. A Monte Carlo simulation is known to be run for a long time and considering that we are doing it a few hundred times every time we are testing parameters, it would save the students behind this paper a few lifetimes of CPU-cycles.

Another interesting aspect would be to implement another optimizer that is not named gradient descent. This might help with earlier convergence in addition to achieving better accuracy.

As this is also a continuation of project 1, it would be interesting to see how the one-body density would look like for this scenario and compare it with a standard variational Monte Carlo program. It would be interesting to see how the system behaves while adding more particles, and taking into account of more than two electrons for Pauli's exclusion principle.

# Appendices

## A Local Energy - E$_L$

The local energy is defined as

$$E_L = \frac{1}{\Psi_T(\boldsymbol{r})} H\Psi_T(\boldsymbol{r}), \quad H = \sum_{i=1}^{N}\left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right) + \sum_{i<j}\frac{1}{r_{ij}} \tag{23}$$

where $H$ is the Hamiltonian for the system, and the first summation term represents the standard harmonic oscillator part and the latter the repulsive interaction between two electrons.

This gives us the following expression for the local energy

$$E_L = \frac{1}{\Psi}(\sum_{p}^{P}(-\frac{1}{2}\nabla_p^2 + \frac{1}{2}\omega^2 r_p^2) + \sum_{p<q}\frac{1}{r_{pq}})\Psi \tag{24}$$

$$= -\frac{1}{2}\frac{1}{\Psi}\sum_{p}^{P}\nabla_p^2\Psi + \frac{1}{2}\omega^2\sum_{p}^{P} r_p^2 + \sum_{p<q}\frac{1}{r_{pq}} \tag{25}$$

$$= -\frac{1}{2}\frac{1}{\Psi}\sum_{p}^{P}\sum_{d}^{D}\frac{\partial^2\Psi}{\partial x_{pd}^2} + \frac{1}{2}\omega^2\sum_{p}^{P} r_p^2 + \sum_{p<q}\frac{1}{r_{pq}} \tag{26}$$

Where $P$ in the summation over is the number of particles and $D$ is the number of dimensions.

By taking the logarithm of the wave function

$$\ln\Psi_T(\mathbf{X}) = -\ln Z - \sum_{m}^{M}\frac{(X_m - a_m)^2}{2\sigma^2} + \sum_{n}^{N}\ln(1 + e^{b_n + \sum_i^M \frac{X_i w_{in}}{\sigma^2}}). \tag{27}$$

we can utilize the relation

$$\frac{1}{\Psi_T}\frac{\partial\Psi_T}{\partial\alpha_i} = \frac{\partial\ln\Psi_T}{\partial\alpha_i} \tag{28}$$

and simplify the expression for the local energy

$$E_L = \frac{1}{2}\sum_{p}^{P}\sum_{d}^{D}(-(\frac{\partial}{\partial x_{pd}}\ln\Psi)^2 - \frac{\partial^2}{\partial x_{pd}^2}\ln\Psi + \omega^2 x_{pd}^2) + \sum_{p<q}\frac{1}{r_{pq}}. \tag{29}$$

Letting each visible node in the Boltzmann machine represent one coordinate of one particle, we obtain

$$E_L = \frac{1}{2}\sum_{m}^{M}(-(\frac{\partial}{\partial v_m}\ln\Psi)^2 - \frac{\partial^2}{\partial v_m^2}\ln\Psi + \omega^2 v_m^2) + \sum_{p<q}\frac{1}{r_{pq}}, \tag{30}$$

where we have that

$$\frac{\partial}{\partial x_m}\ln\Psi = -\frac{1}{\sigma^2}(x_m - a_m) + \frac{1}{\sigma^2}\sum_{n}^{N}\frac{w_{mn}}{e^{-b_n - \frac{1}{\sigma^2}\sum_i^M x_i w_{in}} + 1} \tag{31}$$

and

$$\frac{\partial^2}{\partial x_m^2}\ln\Psi = -\frac{1}{\sigma^2} + \frac{1}{\sigma^4}\sum_n^N \omega_{mn}^2 \frac{e^{b_n+\frac{1}{\sigma^2}\sum_i^M x_i w_{in}}}{(e^{b_n+\frac{1}{\sigma^2}\sum_i^M x_i w_{in}}+1)^2}. \tag{122}$$

If we use $\Psi = \sqrt{F_{rbm}}$ we instead obtain

$$\frac{\partial}{\partial x_m}\ln\Psi = -\frac{1}{2\sigma^2}(x_m - a_m) + \frac{1}{2\sigma^2}\sum_n^N \frac{w_{mn}}{e^{-b_n-\frac{1}{\sigma^2}\sum_i^M x_i w_{in}}+1} \tag{32}$$

and

$$\frac{\partial^2}{\partial x_m^2}\ln\Psi = -\frac{1}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_n^N \omega_{mn}^2 \frac{e^{b_n+\frac{1}{\sigma^2}\sum_i^M x_i w_{in}}}{(e^{b_n+\frac{1}{\sigma^2}\sum_i^M x_i w_{in}}+1)^2}. \tag{33}$$

Where the difference between this equation and the previous one is that we multiply by a factor $\frac{1}{2}$.

## B Quantum Force

In this section we will derive the quantum force and the expression for it is given as

$$F(\boldsymbol{r}) = \frac{2\nabla\Psi_T}{\Psi_T}$$

The importance sampling algorithm requires the use of the quantum force. The most heavy calculation of the quantum force is due to the first derivative of the wave function. However, that expression was derived above,

$$\begin{aligned}\frac{1}{\Psi_T}\nabla_k\Psi_T &= \nabla_k \ln\Psi_T \\ &= \frac{1}{\sigma^2}(x_m - a_m) + \frac{1}{\sigma^2}\sum_n^N \frac{w_{mn}}{e^{-b_n-\frac{1}{\sigma^2}\sum_i^M x_i w_{in}}+1}\end{aligned}$$

This means that the expression for the quantum force is

$$F(\boldsymbol{r}) = -2\left(\frac{1}{\sigma^2}(x_m - a_m) + \frac{1}{\sigma^2}\sum_n^N \frac{w_{mn}}{e^{-b_n-\frac{1}{\sigma^2}\sum_i^M x_i w_{in}}+1}\right). \tag{34}$$

## References

[1] G. Carleo and M. Troyer, Science **355**, Issue 6325, pp. 602-606 (2017)

[2] M. Taut, Phys. Rev. A **48**, 3561 - 3566 (1993)

[3] Mohamed Ismail, Oliver Hebnes
Contains all code and figures for project 1, as well as a Jupyter notebook for easy reproducibiliy.
`https://github.com/Moejay10/FYS4411/tree/master/Project1`

[4] Morten Hjorth-Jensen, Github Repository,
`https://github.com/CompPhysics/ComputationalPhysics2`

[5] Marius Jonsson,
Standard error estimation by an automated blocking method,
Phys. Rev. E98, 15.10.2018
`https://journals.aps.org/pre/abstract/10.1103/PhysRevE.98.043304`

[6] Mohamed Ismail, Oliver Hebnes
Contains all code and figures for project 2, as well as a Jupyter notebook for easy reproducibiliy.
`https://github.com/Moejay10/FYS4411/tree/master/Project2`

[7] Morten Ledum, Github Repository,
Contains the initialised code skeleton used in this project.
`https://github.com/mortele/variational-monte-carlo-fys4411`

[8] Conrad Sanderson and Ryan Curtin. Armadillo: a template-based C++ library for linear algebra. Journal of Open Source Software, Vol. 1, pp. 26, 2016.

[9] Conrad Sanderson and Ryan Curtin. A User-Friendly Hybrid Sparse Matrix Class in C++. Lecture Notes in Computer Science (LNCS), Vol. 10931, pp. 422-430, 2018.

[10] Jørgen Høgberget (2013), Quantum Monte-Carlo Studies of Generalized Many-body Systems, Master Thesis. Fysisk Institutt, Universitetet i Oslo.