The Restricted Boltzmann Machine Applied to the Quantum Many-Body Problem

Dahl, Jon Kristian,[1] Hardersen, Alida,[1] and Sønderland, Per-Dimitri[1]

[1] *University of Oslo*

(Dated: June 6, 2021)

We have calculated the ground state energy of one and two particles confined in a pure isotropic harmonic oscillator potential in one, two, and three dimensions using variational Monte Carlo with a restricted Boltzmann machine representing the many-body wave function. Using the Metropolis-Hastings algorithm with both the brute-force and importance sampling techniques we found that the addition of importance sampling only gave an increase in computation time without considerable improvements in the energy estimate or it's precision. With 2^{18} MC cycles, the energy estimates found are 1.85% (56.4 s) and 1.61% (39.3 s) of the theoretical value for importance and brute force respectively. We were successful in reproducing the ground state energy values of the interacting system of two particles in three dimensions for different values of ω presented in the paper by M. Taut [9] with the largest difference being 0.09 a.u. for $1/\omega = 4$.

I. INTRODUCTION

The study of quantum-mechanical many-body systems has been a large challenge in physics because it is difficult, impossible even, to find exact solutions to the complex many-body wave function. However, huge progress have been made using modern computational algorithms to generate approximate but accurate simulations of such systems, among these the use of Quantum Monte Carlo methods. In 2017, G. Carleo and M. Troyer [2] presented the idea of using a neural network, more precisely a restricted Boltzmann machine (RBM), to represent the many-body wave functions of the Ising model and Heisenberg model. They dubbed such a representation of many-body wave functions the neural-network quantum states (NQS). In theory, given a sufficient number of hidden units, the network should be capable of representing any desired probability distribution with appropriate accuracy [7].

In this project we will apply an RMB to a system of interacting electrons confined in a harmonic oscillator trap. The parameters of the RBM are optimized using variational Monte Carlo (VMC), where we employ the Metropolis-Hastings algorithm to sample particle positions. We will be using both the brute-force approach as well as the addition of importance sampling, and present a comparison of the performance of the two methods. The minimization method we will be using in the VMC is a gradient descent (GD) method based on the variational principle which says that minimizing the energy of the quantum mechanical system should lead to the ground state wave function.

We will begin by presenting the underlying theory of the physical system and an account of the theoretical background of the RBM. We will then present the various methods used in our code, before moving on to present and discuss our results. Mathematical derivations of analytical expressions can be found in the Appendix, along with supplementary figures and tables. All source code is available in the GitHub repository: https://github.com/JonKDahl/FYS4411_V21/tree/main/project2

II. THEORY

A. The Physical System

We consider a system of electrons which are confined in a pure isotropic harmonic oscillator (HO) potential with oscillator frequency ω. The idealised total Hamiltonian using atomic units (a.u.) ($\hbar = c = e = m_e = 1$) is given by,

$$\hat{H} = \underbrace{\sum_i \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right)}_{H_0} + \underbrace{\sum_{i<j} \frac{1}{r_{ij}}}_{H_1} \tag{1}$$

The first term H_0 represents the harmonic oscillator part of the Hamiltonian, while the last term H_1 represents the repulsive Coulomb interaction between two electrons with a distance $r_{ij} = |r_i - r_j|$. If we only consider the HO contribution of the Hamiltonian, H_0, the spatial wave function for one electron in two-dimensions is,

$$\phi_{n_x,n_y}(x,y) = A H_{n_x}(\sqrt{\omega}x) H_{n_y}(\sqrt{\omega}y) e^{-\frac{\omega(x^2+y^2)}{2}} \tag{2}$$

where $H_{n_x}(\sqrt{\omega}x)$ are Hermite polynomials, A is a normalisation constant and n_x, n_y are the principal quantum numbers. The energy is[1]

$$\epsilon_{n_x,n_y} = \omega(n_x + n_y + 1) \tag{3}$$

So for the lowest lying state with $n_x = n_y = 0$ we have $\epsilon_{0,0} = \omega$, and the spatial part of the wave function reduces to,

$$\phi_{0,0}(x,y) = A e^{-\frac{\omega(x^2+y^2)}{2}} \tag{4}$$

We can denote the spin component of the wave function using the spinors $\chi_\uparrow$ and $\chi_\downarrow$ (for spin up and down

[1] We know that for HO we have $E_n = \omega\left(n + \frac{1}{2}\right)$ so the ground state energy for one electron in one dimension is $E_0 = \frac{1}{2}\omega$

respectively) which in the ground state gives the total single particle wave functions,

$$\Psi_\uparrow(r_i) = \phi_{0,0}(r_i)\chi_\uparrow \quad \text{and} \quad \Psi_\downarrow(r_i) = \phi_{0,0}(r_i)\chi_\downarrow \tag{5}$$

When considering a two-electron system in the ground state of the HO potential, the Pauli exclusion principle [6] states that two identical fermions cannot occupy the same quantum state within a quantum system. We therefore require that the overall wave function is anti-symmetric under the interchange of electrons. This overall wave function is given by the Slater determinant [8],

$$\begin{aligned}\Psi(1,2) &= \frac{1}{\sqrt{2!}}\begin{vmatrix}\Psi_\uparrow(1) & \Psi_\uparrow(2)\\ \Psi_\downarrow(1) & \Psi_\downarrow(2)\end{vmatrix}\\ &= \frac{1}{\sqrt{2}}\left[\Psi_\uparrow(1)\Psi_\downarrow(2) - \Psi_\uparrow(2)\Psi_\downarrow(1)\right] \end{aligned}\tag{6}$$

using the expressions for the single particle wavefunctions eq. (5) with the spatial component in the ground state given by eq. (4) we can write this as,

$$\Psi(1,2) = \frac{1}{\sqrt{2}}\phi_{0,0}(1)\phi_{0,0}(2)\left[\chi_\uparrow(1)\chi_\downarrow(2) - \chi_\uparrow(2)\chi_\downarrow(1)\right]$$

because the spatial component is symmetric under the interchange of particles, $\phi_{0,0}(1)\phi_{0,0}(2) = \phi_{0,0}(2)\phi_{0,0}(1)$. To meet the requirement of an overall wave function that is anti-symmetric, the spin component must be anti-symmetric and therefore the system is in a singlet state (i.e. spin quantum number $s = 0$). The ground-state energy is then (denoting $\chi(1,2) = \chi_\uparrow(1)\chi_\downarrow(2) - \chi_\uparrow(2)\chi_\downarrow(1)$),

$$\begin{aligned}\hat{H}\Psi(1,2) &= \frac{1}{\sqrt{2}}\left[\phi_{0,0}(2)\hat{H}\phi_{0,0}(1) + \phi_{0,0}(1)\hat{H}\phi_{0,0}(2)\right]\chi(1,2)\\ &= \frac{1}{\sqrt{2}}\left[\phi_{0,0}(2)E_1\phi_{0,0}(1) + \phi_{0,0}(1)E_2\phi_{0,0}(2)\right]\chi(1,2)\\ &= (E_1+E_2)\frac{1}{\sqrt{2}}\phi_{0,0}(1)\phi_{0,0}(2)\chi(1,2)\\ &= (E_1+E_2)\Psi(1,2)\end{aligned}\tag{7}$$

using $E_1 = E_2 = \epsilon_{0,0} = \omega$ the ground state energy for two electrons in a two-dimensional HO potential is 2ω. The general expression for the energy of the non-interacting system is given by

$$E = \frac{1}{2}PD, \tag{8}$$

where $P = [1,2]$ is the number of particles and $D = [1,2,3]$ is the number of dimensions. When including the interaction part of the Hamiltonian, the energy for a system of two electrons in two-dimensions is 3ω [9].

B. Artifical Neural Network

Artifical neural networks (ANN) are a wide range of various computation structures that are built around the concept of the neuron, also called a node, perceptron or unit:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b_i\right) = f(z). \tag{9}$$

The neuron is essentially a linear combination of inputs x_i weighted by w_i and transformed by $f(.)$, where the latter is called an activation function. If $f(z) = z$, that is, the identity activation function, then eq. (9) would simply be ordinary linear regression. However, where the neuron as used in a neural network differs from linear regression is by $f(.)$ being a *non-linear* function for one or more neurons. For example the sigmoid activation function is one such non-linear function

$$f(z) = \frac{1}{1+e^{-z}}. \tag{10}$$

1. *Feed-Forward Neural Network*

Furthermore, unlike linear regression, the inputs of eq. (9) are not necessarily fixed, but can themselves be non-linear functions that depend on a linear combination of inputs. Meaning that beginning with feature-data inputs, we apply a series of transformations called *feed-forward* propagation. This is often schematically illustrated as multiple layers of neurons, and the structure is called a feed-forward neural network (FFNN). The weights w of the network can be iteratively adjusted, or "trained", with the method of *gradient descent* through a procedure called *backpropagation*. The FFNN, and its derivatives, is arguably the most popular network architecture, used for a large variety of problems. These types of networks are called *discriminative* models. However, we shall not dwell too much on this type of network as they have limitations in regards to the problem at hand.

C. Discriminative and Generative Models

Discriminative models learn the direct map between inputs x and outputs y so that they can be subsequently applied to distinguish between outputs according to which inputs have been provided and then yield a prediction to the user. In other words, a discriminative model seeks to learn the decision boundary between classes. Essentially this means estimating the conditional distribution $p(y|x)$, where a target output y is conditioned on the feature input x [1]. On the other hand we have *generative* models, which are the focus of this article. These types of models seek to estimate the joint distribution $p(x, y)$. Learning not just the map between input and outputs, but a characterization of the entire population of examples observed during the training process. For many purposes, solving a more general problem than necessary might be excessive. However, for our purpose the advantage of learning the joint probability distribution is that we can generate completely new examples of input–output pairs. Generative models are used extensively for the purpose of *unsupervised learning* in quantum mechanical problems. Unsupervised means unlabeled data, without given targets y, so we do not tell the model what patterns it should try to learn as in supervised learning, instead we let the model discover these patterns on its own.

D. Restricted Boltzmann Machine

The Boltzmann machine (BM) is a neural network and is thus also built around the concept of the neuron as described in section II B, and it is a generative model as described in section II C. The architecture of this specific artificial neural network is that of a bipartite graph, meaning that the neurons can be divided into two disjoint and independent sets, where non-linear transformations are applied in between the two sets. The two sets, or layers, of neurons consist of the M-dimensional vector $\mathbf{x}$ called the *visible layer*, and N-dimensional vector $\mathbf{h}$ called the *hidden layer*. The visible layer represents a system of inputs, or after the network has been trained, a reconstructed output. The hidden layer on the other hand increases the model's expressive power. The system of neurons are interconnected by a set of weights constituting a $M \times N$ matrix. The layers each have bias vectors, $\mathbf{a}$ and $\mathbf{b}$, corresponding to the visible layer and the hidden layer, respectively. They thus conform to the dimensions of their respective layers with vector length M and N for the visible and hidden biases, respectively. The Boltzmann machine comes in two types, either it is *unrestricted* or *restricted*, where the two types refer to whether the individual units in the hidden layer have connections in between each other or not. It is the restricted Boltzmann machine (RBM) that will be the focus of our study, which has no connections within the hidden layer [4].

As mentioned in the introduction, the RBM was recently used with success by G. Carleo and M. Toyer [2] to a quantum mechanical spin lattice system in the context of the Ising model and Heisenberg model, and is used as a basis for this article. The intended goal is to train an RBM to obtain an approximation to an unknown probability distribution, in which we may sample from. Specifically we seek to obtain a surrogate distribution for the true wave function of the quantum mechanical problem at hand.

In eq. (9) we formulated the neuron, however, this was a deterministic neuron. Here we look at a stochastic neuron. The simplest example is that of a binary neuron, which unlike the deterministic neuron provides a certain probability of being 1 or 0,

$$p(y=1) = f\left(\sum_{i=1}^{n} w_i x_i + b_i\right) \tag{11}$$

$$p(y=0) = 1 - p(y=1). \tag{12}$$

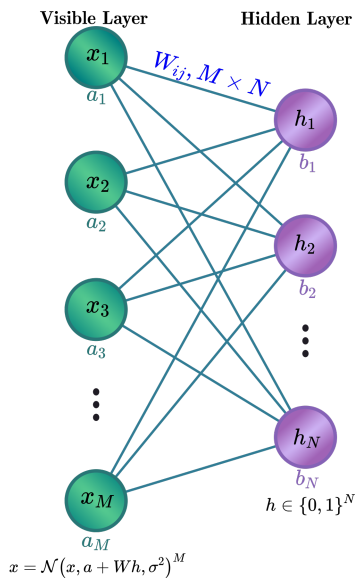

FIG. 1: Guassian-binary Boltzmann machine with an arbitrary number of visible neurons M and hidden neurons N.

If the RBM uses eq. (11) for both hidden and visible neurons it is called a *binary–binary* RBM.

However, in this article we seek a distribution over continuous space, hence an RBM that only yields binary values is not sufficient. We shall instead employ the Gaussian-binary RBM, as shown in fig. 1, where we use Gaussian activations in the visible neurons and binary activations in the hidden neurons. RBM's are named as such because they involve the application of the *Boltzmann distribution*, familiar from statistical physics, namely

$$p_{\text{rbm}}(\mathbf{x}, \mathbf{h}) = \frac{1}{Z} e^{-\frac{1}{T_0} E(\mathbf{x}, \mathbf{h})}, \tag{13}$$

where $\mathbf{x}$ is the input, $\mathbf{h}$ the hidden neurons and T_0 is the temperature, often set equal to one. For eq. (13) to be a proper distribution and yield normalized probabilities in the range $(0, 1)$, it requires a normalizing constant, namely Z. Lastly, $E(\mathbf{x}, \mathbf{h})$ is the *energy function*, which for our problem is defined as

$$E(\mathbf{x}, \mathbf{h}) = \sum_{i}^{M} \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_{j}^{N} b_j h_j - \sum_{i,j}^{N,M} \frac{x_i \omega_{ij} h_j}{\sigma_i^2}, \tag{14}$$

where M is the number of nodes in the visible layer and N the number of nodes in the hidden. As can be seen from the energy function, that the squared dependency $(x_i - a_i)^2$ is our Gaussian component, while the remaining expression linearly dependent on h_j is our binary component.

E. The Cost Function and Bayes Rule

For any machine learning problem one needs to define an optimization objective that the algorithm seeks to optimize. This is commonly called a *cost function*. In deterministic supervised learning this is typically some measure of distance between true target values y and predictions $\hat{y}$, such as the $\mathcal{C}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$. In fact, any such cost function $\mathcal{C}$ can be formulated probabilistically, specifically we have

$$\mathcal{C}\left(\hat{y}, y\right) \sim -\ln p\left(y|x, \theta\right), \tag{15}$$

where θ is the parameters of the model. The right-hand side is the negative log-likelihood, where the likelihood is a product of conditional probability distributions. The minimization of a cost function is thus akin to the maximization of a probability distribution, or the *maximum likelihood estimate*. The MSE for example is just proportional to the argument of a Gaussian with constant variance. In our case we are not dealing with labeled target values, so we must formulate a different measure suited for minimization. Let us see how eq. (15) relates to a generative model. Note, in section II C we stated that in a generative model we have a joint distribution, while in eq. (15) we have a conditional distribution. Through the *product rule* of probability theory we have that the joint distribution is

$$p(x, y) = p(y|x)p(x). \tag{16}$$

The difference between the two is the prior distribution $p(x)$, which is a probability distribution that expresses our prior belief about a problem. As before, the joint distribution necessitates a normalizing constant, Z to yield proper probabilities. By dividing the right hand side of eq. (16) by the normalizing constant we obtain

$$p(y|x) = \frac{p(x|y)p(x)}{Z}, \tag{17}$$

which is essentially *Bayes' rule*. The left hand side of eq. (17) is the normalized joint distribution, which is the posterior probability distribution of Bayesian statistics. By applying the negative logarithm of this distribution, as in eq. (15), we can formulate our cost function.

We do not know the true distribution, thus we must define a suited functional form that we can attempt to fit by variational inference. Fortunately, we have already defined such a distribution with $p_{\text{rbm}}(\mathbf{x}, \mathbf{h})$ in eq. (13). To continue for a moment with a Bayesian line of thought, for a variational Bayesian problem we would have to choose a prior distribution and preferably a likelihood distribution that we believe will produce a joint distribution that is sufficiently close, in form, to the true distribution, such that the algorithm can simply adjust the *latent* parameters, that is, the parameters of surrogate distribution, in order to obtain the best fit. Let us rewrite eq. (13) by including the energy function, and setting $T_0 = 1$, to see that we have indeed already chosen a prior and a likelihood,

$$\begin{aligned} p_{\text{rbm}}(\mathbf{x}, \mathbf{h}) &= \tfrac{1}{Z}\textstyle\sum_{h_j} e^{-E(\mathbf{x},\mathbf{h})} \\ &= \tfrac{1}{Z}\textstyle\sum_{h_j} e^{\sum_i^M \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_j^N b_j h_j - \sum_{i,j}^{N,M} \frac{x_i \omega_{ij} h_j}{\sigma_i^2}} \\ &= \tfrac{1}{Z} e^{\sum_i^M \frac{(x_i - a_i)^2}{2\sigma_i^2}} \textstyle\prod_j^N \left(1 + e^{b_j + \sum_i^M \frac{x_i \omega_{ij}}{\sigma^2}}\right), \end{aligned} \tag{18}$$

which is an expression consisting of two terms, namely an unnormalized Gaussian prior $e^{-\sum_i \frac{(x_i - a_i)^2}{2\sigma^2}}$, and the likelihood function $\prod_j \left(1 + e^{b_j + \sum_i \frac{x_i w_{ij}}{\sigma^2}}\right)$. The overall expression is normalized by Z, such that we can view the RBM distribution as a posterior distribution. However, we should note that unlike a fully Bayesian procedure, the prior here is not fixed. When the prior is trainable the procedure is often called *empirical Bayes*. The expression in eq. (18) is what we will be using as the NQS wave function

$$\psi(x) = \frac{1}{Z} e^{-\sum_i \frac{(x_i - a_i)^2}{2\sigma^2}} \prod_j \left(1 + e^{b_j + \sum_i \frac{x_i w_{ij}}{\sigma^2}}\right). \tag{19}$$

In our discussion on Bayes' rule in eq. (17) we stated that by using the negative logarithm we can obtain our cost function. We thus use eq. (15) and obtain

$$C\left(\{\theta_i\}\right) = \left\langle E\left(\boldsymbol{x}, \{\theta_i\}\right)\right\rangle_{\text{data}} + \log Z\left(\{\theta_i\}\right), \tag{20}$$

where we have taken the average over the data, while θ_i denotes a model parameter, part of all model parameters $\boldsymbol{\theta} = a_1, \ldots, a_M, b_1, \ldots, b_N, w_{11}, \ldots, w_{MN}$. The gradient of the cost function, which will come in handy for gradient descent (see section III D), is

$$\frac{\partial \mathcal{C}\left(\{\theta_i\}\right)}{\partial \theta_i} = \left\langle \frac{\partial E\left(x, \theta_i\right)}{\partial \theta_i} \right\rangle_{data} + \frac{\partial \log Z\left(\{\theta_i\}\right)}{\partial \theta_i}. \tag{21}$$

To make sure the joint distribution is indeed normalized, that is, $\iint \frac{p(\mathbf{x},\mathbf{h})}{Z} d\mathbf{x} d\mathbf{h} = 1$, we insert the expression in eq. (13) for the joint distribution and multiply by Z, which gives

$$Z = \iint \exp\left(-\frac{1}{T_0} E(\mathbf{x}, \mathbf{h})\right) d\mathbf{x} d\mathbf{h}. \tag{22}$$

III. METHODS

A. Variational Monte Carlo

The Variational Monte Carlo (VMC) method is often used to approximate the ground state energy of a quantum mechanical system by using the variational principle which states that for any given trial wave function, ψ_T, the energy is always larger than or equal to the ground state energy,

$$E = \frac{\langle \psi_T | \hat{H} | \psi_T \rangle}{\langle \psi_T | \psi_T \rangle} = \frac{\int dr\ \psi_T^*(r) \hat{H} \psi_T(r)}{\int dr\ |\psi_T|^2} \geq E_0 \tag{23}$$

We can use Monte Carlo (MC) integration with N cycles to avoid having to solve the integrals analytically by defining the normalized probability density function (pdf) $p(r)$ and the local energy E_L as,

$$p(r) \equiv \frac{|\psi_T(r)|^2}{\int dr\ |\psi_T(r)|^2} \quad \text{and} \quad E_L \equiv \frac{1}{\psi_T}\hat{H}\psi_T \tag{24}$$

this allows us to rewrite the variational energy E,

$$E = \int dr\ E_L(r)p(r) \approx \frac{1}{N}\sum_{i=1}^{N} E_L(r_i) \tag{25}$$

Thus, the variational energy can be estimated as the average value of the local energy over a sample of N points, r_i which are sampled from the pdf, $p(r_i)$. We do not have an expression for this pdf, so instead we will use Metropolis-Hastings sampling[3] to sample the positions of the electrons. Using the NQS wave function given in eq. (19) and the Hamiltonian in eq. (1) we can find an analytic expression for the local energy (see appendix A for details),

$$E_L = \sum_i \frac{1}{2}\left[-(\nabla_i \ln\psi)^2 - \nabla_i^2 \ln\psi + \omega^2 r_i\right] + \sum_{i<j}\frac{1}{r_{ij}} \tag{26}$$

where the analytical expressions for $\nabla_k \ln\psi$ and $\nabla_k^2 \ln\psi$ are given in appendix A. Note that in the results section of this project we will sometimes use the term *local energy* about the expectation value of the local energy.

B. Metropolis Sampling

A detailed description of the Metropolis-Hastings algorithm can be found in our previous project [3]. The algorithm allows us to sample the target probability distribution $p(r)$ by constructing a Markov chain that converges to $p(r)$. Starting at an initial state i with position r_i, the probability of transitioning to a new proposed state j is

$$P(r_j|r_i) = A(r_j|r_i)T(r_j|r_i) \tag{27}$$

where $A(r_j|r_i)$ is the probability that the proposed move is accepted and $T(r_j|r_i)$ is the probability of the proposed move. Whether a move is accepted is determined by the Metropolis acceptance criterion,

$$A(r_j|r_i) = \min\left[1, q(r_j, r_i)\right] \tag{28}$$

where $q(r_j, r_i)$ is the ratio of acceptance probabilities,

$$q(r_j, r_i) \equiv \frac{A(r_j|r_i)}{A(r_i|r_j)} = \frac{T(r_i|r_j)p(r_j)}{T(r_j|r_i)p(r_i)} \tag{29}$$

The proposed new state is automatically accepted if the ratio $q > 1$, but if $q \leq 1$ we compare it to a random number $x \in [0, 1]$ and if $q > x$ the move is accepted. This is to reduce the chance of getting stuck in a local minimum. The proposal transition probability $T(r_j|r_i)$ can be defined in numerous ways. In the present project we will compare the performance of the two following methods:

1. *Brute-force sampling*

We define the proposal transition probabilities $T(r_j|r_i) = T(r_i|r_j)$, meaning that they will cancel in the ratio in eq. (29) leaving the brute-force ratio as,

$$q(r_j, r_i) = \frac{p(r_j)}{p(r_i)} = \frac{|\psi_T(r_j)|^2}{|\psi_T(r_i)|^2} \tag{30}$$

In this method, the new proposed moves are chosen as

$$r_j = r_i + \xi\Delta r \tag{31}$$

where $\xi \in [-0.5, 0.5]$ is a stochastic variable drawn from a uniform distribution and Δr is the step size. In this project we choose to use a step size of $\Delta r = 1$. While effective in many cases, the brute-force approach suffers some downsides. Many of the proposed samples are rejected, meaning that we will end up sampling the same state several times which is an inefficient use of calculation time and resources. We refer to the ratio of accepted samples to total samples as the acceptance rate.

2. *Importance sampling*

Importance sampling aims to increase the acceptance rate by making the sample distribution dependent on the quantum force eq. (33). This will help us sample more relevant points by accounting for the direction of the moving particles. The importance sampling method is based on the Fokker-Planck and Langevin equations and is described in more detail in [3]. Based on the Fokker-Planck equation which describes the time evolution of the probability density of a particle that is under the influence of drift forces, we define the proposal transition probability using Greens function $T(r_j|r_i) = G(r_j, r_i, \Delta t)$, with

$$G(r_j, r_i, \Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \times \exp\left[\frac{-(r_j - r_i - D\Delta t F(r_i))^2}{4D\Delta t}\right] \tag{32}$$

here D is a diffusion coefficient which in the present project is set to $D = 0.5$, and $F(r_i)$ is the quantum force defined as,

$$F(r) = \frac{2\nabla\psi_T(r)}{\psi_T(r)} = 2\nabla\ln\psi_T \tag{33}$$

where the expression for $\nabla \ln \psi$ using the NQS wave function in eq. (19) is found in appendix A. The ratio in the acceptance criterion becomes,

$$q(r_j, r_i) = \frac{G(r_i, r_j, \Delta t)|\psi_t(r_j)|^2}{G(r_j, r_i, \Delta t)|\psi_t(r_i)|^2} \tag{34}$$

The new proposed positions are determined by solving Langevin's equation[2] using Euler's method,

$$r_j = r_i + DF(r_i)\Delta t + \zeta\sqrt{\Delta t} \tag{35}$$

where ζ is a stochastic variable drawn from a normal distribution and Δt is a chosen time step. In this project we choose a time step of $\Delta t = 0.05$.

C. Brute-force vs importance sampling

When choosing whether to use brute-force or importance sampling, we consider which of the methods is able to deliver a good estimate of the ground state energy with the lowest uncertainty in the shortest amount of time. Therefore, we compare the two methods by running them with all the same hyperparameters and see how fast they converge, within a tolerance, on a known ground state energy. We feed the random number generator with the same seed to make sure the two methods start on the same footing. We know that with importance sampling we sample from an improved distribution compared to the brute-force approach, but with the added computational cost of calculating Greens function eq. (32) and the quantum drift force eq. (33). Prior to the experiment, it is unclear whether the increased computational cost related to importance sampling will be balanced out by the improved sampling.

D. Gradient Descent

From the variational principle we know that minimizing the local energy should lead to the ground state wave function, i.e. we need to search for the parameter values for which the local energy is at a minimum. This is done using the gradient descent method which is commonly used in machine learning where we find the optimal parameters of some function by iteratively minimizing its gradient. The local energy, which is given in eq. (26), is dependent on the RBM weights (w) and biases (a and b) and in the following we will denote these parameters collectively as α, so $\alpha_i = a_1, ..., a_M, b_1, ..., b_N, w_{11}, ..., w_{MN}$. We make an initial guess for the values of the parameters and calculate the gradient of the expectation value of the local energy using these.

[2] Langevin: $\frac{\partial r(t)}{\partial t} = DF(r(t)) + \eta$

$$G_i = \frac{\partial \langle E_L \rangle}{\partial \alpha_i} = 2\left[\langle E_L \frac{1}{\psi}\frac{\partial \psi}{\partial \alpha_i}\rangle - \langle E_L \rangle \langle \frac{1}{\psi}\frac{\partial \psi}{\partial \alpha_i}\rangle\right] \tag{36}$$

We then update the parameter values by moving in the direction of the negative gradient, this way the new parameters chosen will lead to a smaller gradient value while the step between the new parameters ηG_i will become smaller and smaller as the gradient decreases.

$$\alpha_{i+1} = \alpha_i - \eta\ G_i \tag{37}$$

here η represents the learning rate and can be both constant or vary according to some function. In this project we will be testing both variants, the variable learning rate used is described in section III F. From the form of the gradient eq. (36) we need to find analytic expressions for the partial derivatives of the NQS wave function eq. (19) for each of the parameters in α, these are found in appendix B.

E. Initial distributions

The hidden biases ($\mathbf{b}$), visible biases ($\mathbf{a}$), and weights ($\mathbf{w}$) are all initialized with random numbers drawn from a normal distribution. These normal distributions are characterized by their means and standard deviations which we can adjust to see what effect the initial distributions have on the system. We will from here keep all three standard deviations equal and refer to them as σ_{init}. The means are all set to 0.

Since the biases and weights all act as variational parameters, we expect the initial distributions to have a large impact on the system. If we draw a set of initial parameters which are far from the ground state wave function, we expect the system to use more gradient descent steps before converging on the true answer. Consider now the wave function in equation (19). If we set all the biases and weights to zero ($\sigma_{\text{init}} = 0$) we get, for one particle in one dimension,

$$\psi = \frac{1}{Z} e^{-\frac{x}{2\sigma^2}} (1 + e^{0+0}), \tag{38}$$

and if we set $\sigma^2 = 1/\omega$ we get

$$\psi = \frac{2}{Z} e^{-\omega\frac{x}{2}} \tag{39}$$

which is the ground state wave function as seen in eq. (4) [3] [4]. The same applies for 1, 2, and 3 dimensions with 1

[3] Equation (4) is for 2, not 1 dimension, but simply remove the y coordinate and you have the ground state wave function for 1 dimension.

[4] Bear in mind that σ in equations (38) and (39) and σ_{init} are not the same. σ comes from the Gaussian part of the Gaussian-binary RBM, while σ_{init} comes from the initial distribution of biases and weights.

and 2 particles, but only for the non-interacting case. Due to the closer resemblance of the ground state wave function, we set $\sigma^2 = 1/\omega$ for all calculations in this experiment. Based on this, we expect the RBM to quickly converge to the correct non-interacting answer for small values σ_{init} and to reproduce the exact answer if we set $\sigma_{\text{init}} = 0$.

F. Variable learning rate

Gradient descent uses a learning rate (η) which decides the size of the gradient steps. A well chosen learning rate allows us to converge on a good answer quickly, and it may allow us to decrease the number of Monte Carlo cycles needed for this convergence. In general, we want to choose a large as possible learning rate since this will converge on the answer as quickly as possible. But we must be careful; a too large learning rate can give an explosive increase in the calculated local energy, giving bogus results or crashing the computation. In machine learning, when using gradient descent or any of its derivatives, one often wants to start with a large learning rate at the start of the calculations when the system is far from the desired answer, for then to decrease the learning rate as the system converges on the wanted value. If the learning rate is large when the system is close to converging on the answer, the next step might overshoot the wanted answer, and in the worst case this overshoot continues for several more gradient descent steps resulting in unusable results or a crashed computation.

To combat the problem of overshooting we have implemented a variable learning rate which decreases as the number of gradient descent steps increases. We have chosen to define the variable learning rate as

$$\eta(I) = \frac{\eta_{\text{init}} c}{I f_\eta + c}, \tag{40}$$

where η_{init} is the initial learning rate (at iteration $I = 0$), $I \in \mathbb{N}_0$ is the current gradient descent iteration, f_η is a factor and c is a constant which both decides the steepness and the shape of the learning rate decline. Equation (40) is a fairly simple expression where we easily can choose the initial learning rate, which is the motivation for using this exact form. By adjusting f_η we can change the aggressiveness of the decrease of the learning rate.

In figure 2 we see equation (40) plotted for different factors, f_η, as indicated in the plot legend. All factors use the same initial learning rate of $\eta_{\text{init}} = 0.18$ and constant $c = 1$, and we can decide how rapidly we want the learning rate to decrease by increasing f_η. We will later in this report see what significance this has.

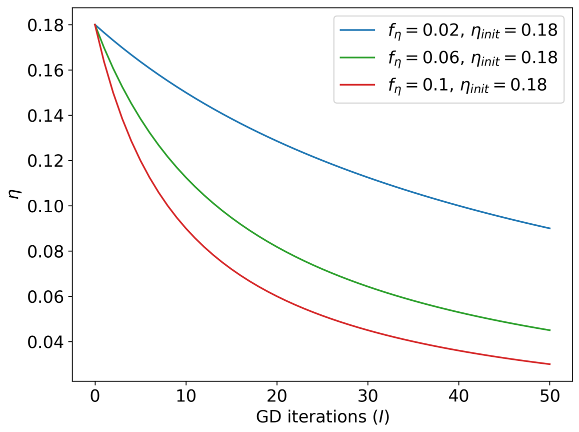

FIG. 2: Visualization of variable learning rate equation (40). The learning rate decreases with each new gradient descent iteration, making sure that smaller steps are performed as the system (hopefully) converges. The constant c is set to $c = 1$.

G. Error estimation

As described in section III A, the energy is estimated as the average value of the local energy in a sample of N measurements, where N is the number of MC cycles. The simulated measurements are created using Metropolis-Hastings sampling so each point is drawn from a PDF that is conditional on the previous point, making the measurements serially correlated. To account for this correlation we will be using the *blocking method* when evaluating the statistical errors on our measurements. The method used is based on the article by M. Jonsson [5] and the general idea is to reduce the correlation in the measurements by repeatedly applying blocking transformations until the correlation no longer influences the error estimate. We refer to our previous project [3] for a more detailed discussion of the method.

IV. RESULTS AND DISCUSSION

A. brute-force vs importance sampling

Table I shows a comparison between brute-force and importance sampling. We run both methods for the same hyperparameters as indicated in the table, and look at the time, in GD iterations and seconds, both methods use to come within a tolerance of 0.01 a.u. of the known answer. The different configurations won't land exactly on the tolerance, so we have included the relative difference to the exact answer at the given iteration where the calculated ground state energy meets the tolerance.

What is immediately apparent from the data in table I is that brute-force is faster, in seconds, than importance sampling at the same number of MC cycles. This attributes to the fact that importance sampling is dependent on calculating Greens function eq. (32) and the quantum drift force eq. (33) for each MC cycle. brute-

TABLE I: Comparison of brute-force and importance sampling. 1 particle in 1 dimension for $\omega = 1$ which has a known ground state energy of $E_{\text{gs}} = 0.5$ a.u.. We use a initial distribution of the biases and weights with $\sigma_{\text{init}} = 2$, and 2 hidden nodes in the RBM. The calculations are run until $|E_L - E_{\text{gs}}| \leq 0.01$ a.u. to indicate how fast the different configurations converge on the known answer. The relative difference $100 \cdot |E_L - E_{\text{gs}}|/E_{\text{gs}}$ is also displayed since the difference between the calculated energies and the exact answer for the different configurations won't be exactly equal to 0.01 a.u.. The times, in GD iterations and in seconds, are also listed.

Method	MC cycles	E_L [a.u.]	Rel. diff [%]	GD iterations	time [s]
Brute	262144	0.5080 ± 0.0002	1.6097	40	39.3
Importance	262144	0.5093 ± 0.0002	1.8549	39	56.4
Brute	131072	0.5088 ± 0.0003	1.7522	40	20.9
Importance	131072	0.5085 ± 0.0003	1.7014	38	28.6
Brute	65536	0.5070 ± 0.0004	1.4061	39	12.0
Importance	65536	0.5082 ± 0.0004	1.6404	39	17.3
Brute	32768	0.5066 ± 0.0006	1.3231	38	7.8
Importance	32768	0.5085 ± 0.0007	1.7100	36	10.1
Brute	16384	0.5091 ± 0.0007	1.8281	41	5.6
Importance	16384	0.5079 ± 0.0008	1.5884	38	7.2
Brute	8192	0.506 ± 0.002	1.2576	34	4.3
Importance	8192	0.507 ± 0.001	1.3916	38	5.1
Brute	4096	0.502 ± 0.003	0.3417	28	3.7
Importance	4096	0.509 ± 0.003	1.7084	31	4.3
Brute	2048	0.493 ± 0.005	1.4952	25	3.5
Importance	2048	0.490 ± 0.004	1.9822	27	3.9
Brute	1024	0.493 ± 0.005	1.4345	28	3.4
Importance	1024	0.508 ± 0.005	1.5776	31	3.8

force does not depend on the two mentioned calculations and is therefore less computational intensive per MC cycle.

Our next observation is that the blocking-estimated uncertainties are approximately equal, with brute-force beating importance sampling at a few configurations and vice versa. The improved sampling in importance sampling does not seem to improve the uncertainties. The uncertainties are strictly correlated to the number of MC cycles which we can clearly see in the data. At 1024 MC cycles we record uncertainties of 0.005 a.u. and at 262144 cycles we record 0.0002 a.u., for both methods. The uncertainties are strictly decreasing with increasing number of MC cycles for both methods.

The next interesting observation is the fact that importance sampling seems to use fewer GD iterations than brute-force, but only at the largest amount of MC cycles. At fewer MC cycles we in fact see the opposite trend, which might mean that importance sampling is the better choice for very large amounts of MC cycles, beyond what we explore in this experiment. The systems we study are relatively small systems with 1 or 2 particles. Few particles means less complexity and less complexity means that we need relatively few MC cycles to generate good results. For our use, brute-force is the preferred choice since it delivers approximately equal results as importance sampling in a shorter amount of time, but our data is still open to that importance sampling might be the preferred choice for larger systems. We will from here use brute-force for all of the following results.

The relative differences hover around approximately 1.5%, and we don't see any of the two methods performing better or worse. This is expected since the exact end energy value is dependent on how large the last gradient step is which has a stochastic dependency.

B. Reproducing analytical results in the non-interacting case

In table II we see the calculated ground state local energy in 1, 2, and 3 dimensions for 1 and 2 particles, along with the exact analytical results given by eq. (8), for the non-interacting case. The uncertainties are calculated using the blocking method. Consider first the $\sigma_{\text{init}} = 0.5$ column. We see that the RBM implementation is successful in reproducing the known solutions, though some of the values are off by more than the blocking-estimated uncertainty. This hints to the fact that there is some systematic uncertainty in the RBM which we have not accounted for. The calculations use 2^{20} MC cycles and 50 GD iterations. We experience that increasing the number of GD iterations beyond 50 has a negligible impact on the result. Increasing the number of MC cycles beyond 2^{20} has little to no overall effect on the energy values and the uncertainties. For the data in table II, a constant learning rate of $\eta = 0.05$ is used.

The convergence time is, as discussed in section III E, highly dependent on the distribution from which the initial values of the biases and weights are drawn. Convergence time here being measured in GD iterations. In table II we also include $\sigma_{\text{init}} = 0$ which means that all biases and weights are initially set to zero. We see that the exact values are produced in the first gradient de-

TABLE II: Calculated ground state local energies compared to exact analytical results. Energies are in atomic units. 2^{20} MC cycles with 50 GD iterations. σ_{init} is the standard deviation of the normal distribution from which the initial biases and weights are drawn. Uncertainties are estimated with the blocking method.

p, d	$E_L(\sigma_{\text{init}} = 0)$	$E_L(\sigma_{\text{init}} = 0.5)$	$E_{L,\text{exact}}$
1, 1	0.5 ± 0	0.5008 ± 0.0002	1/2
1, 2	1 ± 0	1.0206 ± 0.0009	1
1, 3	1.5 ± 0	1.5110 ± 0.0007	3/2
2, 1	1 ± 0	1.0036 ± 0.0004	1
2, 2	2 ± 0	2.023 ± 0.001	2
2, 3	3 ± 0	3.038 ± 0.001	3

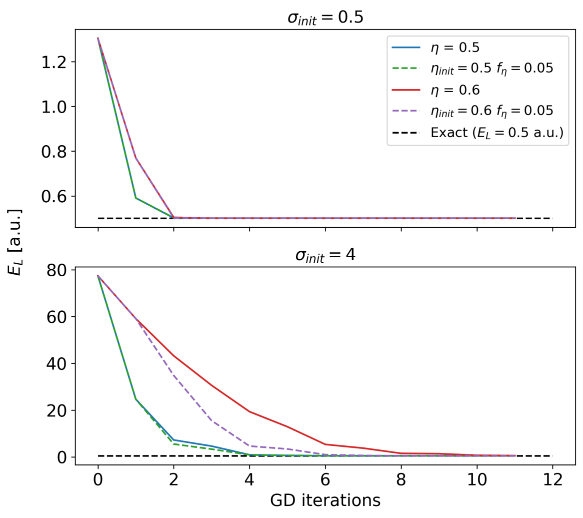

FIG. 3: Ground state local energy as a function of gradient descent iterations. 1 particle in 1 dimension with 2^{18} MC cycles. Top plot has a initial weights and bias distribution with $\sigma_{\text{init}} = 0.5$ and the bottom plot has $\sigma_{\text{init}} = 4$. Two pairs of learning rates are plotted for each initial distribution: A constant learning rate and its variable counterpart for $\eta = 0.5, 0.6$.

scent iteration, since the wave function, equation (19), is reduced to the wave function of the ground state, as seen in equation (39). One important thing to note is that with all biases and weights set to zero, the two terms in eq. (36) are equal, making the gradient zero. The RBM will therefore produce the exact same answer every gradient descent iteration which is confirmed by the zero uncertainties in table II. It might therefore be a bit misleading to conclude the uncertainty to be exactly zero since we have initialized the RBM at a configuration which is impossible to change.

C. Learning rate

Figure 3 shows the local energy as a function of GD iterations for 1 particle in 1 dimension, plotted for two constant learning rates and their variable brethren. Consider first the top plot where we have an initial distribution with $\sigma_{\text{init}} = 0.5$. The difference between the constant and variable learning rate is virtually unnoticeable. At this configuration we see no benefit in using a variable learning rate. This may attribute to the fact that the system converges on the known value of 0.5 a.u. in only two GD iterations, and the variable learning rate does not have time to correct itself in so few steps.

Consider now the bottom plot. Here we use an initial distribution with $\sigma_{\text{init}} = 4$ which is a considerably larger spread in initial values than the top plot. We see that the system uses more GD iterations than for $\sigma_{\text{init}} = 0.5$ which gives the variable learning rate time to decrease with increasing GD iterations, as seen in equation (40). We see a small improvement for $\eta = 0.6$, where the variable learning rate makes the system converge 1-2 GD iterations fewer than the constant learning rate. For $\eta = 0.5$ there is no improvement, and both variable and constant learning rate both converge in the same amount of GD iterations.

What we take from this is that a variable learning rate might improve the convergence time slightly if the initial values are far from the final configuration, but in no way a drastic improvement. Whats more, the parameters needed to see the improvement that the variable learning rate can give requires a bit of tuning of hyper-parameters and may not be reasonable to spend much time on. We experience that the best way to approach the problem of choosing the correct variable learning rate is to run the desired calculations with as large learning rate as possible. When the learning rate is so large that the system overflows, tune the learning rate down a bit and then make it variable with a factor of approximately $f_\eta = 0.05$.

A topic for a future study is to make a more detailed analysis of the factor f_η which decides how rapidly the learning rate decreases with increasing GD iterations. We might be able to choose an even larger initial learning rate, making the system converge quickly, if we compensate with a larger f_η making the large learning rate decrease rapidly to avoid overshooting the target value.

D. The number of hidden nodes

We now proceed to investigate how the number of hidden nodes (b) in the RBM will impact our results. In this part of the analysis we continue the use of a non-interacting system with one particle in one dimension. The weights and biases are initialized using a normal distribution with $\mu = 0$ and $\sigma_{\text{init}} = 1$. We use the brute-force sampling method with a step size of $\Delta r = 1$. The RBM is run for various values of hidden nodes in the in-

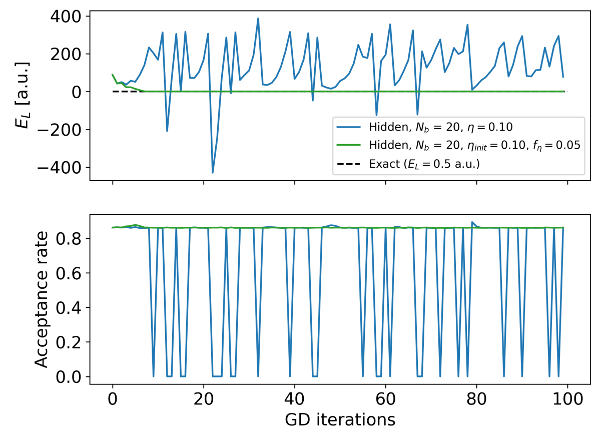

FIG. 4: Comparison of the energy and acceptance rate as a function of GD iterations using brute-force metropolis sampling with 2^{18} MC cycles for 1 particle in 1 dimension. The RBM's both have 20 hidden nodes, the initial distribution of the biases and weights is $\sigma_{init} = 1$ and both a constant and variable learning rate is used

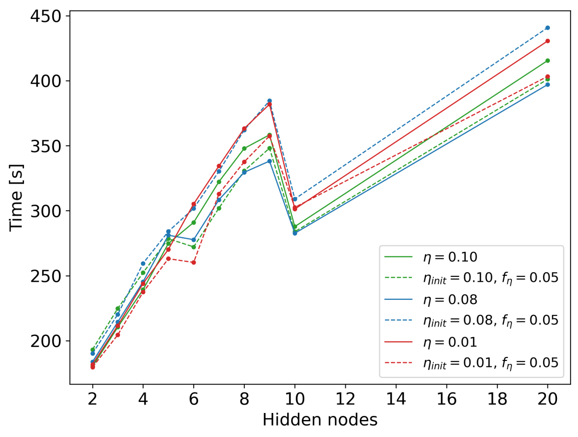

FIG. 5: Processing time for 100 GD iterations as a function of hidden nodes using brute-force sampling for 1 particle in 1 dimension with different learning rates. The number of MC cycles is 2^{18}, and we use an initial distribution of the biases and weights with $\sigma_{\text{init}} = 1$.

terval $[2, 20]$ with 100 GD iterations for three constant learning rates $\eta = 0.1, 0.08, 0.01$, as well as for three variable learning rates with $\eta_{\text{init}} = 0.1, 0.08, 0.01$ and $f_\eta = 0.05$. Tables of results from all these configurations can be found in appendix C 1.

Our initial discovery is that as the number of hidden nodes increases, we need to compensate with a decrease in learning rate to converge on the exact answer. An example of what happens for an RBM with 20 hidden nodes when the learning rate is too high is shown as a blue line in fig. 4. The top figure shows the local energy using both a constant learning rate (blue) and a variable learning rate (green) as a function of GD iterations, while the bottom figure shows the corresponding acceptance rates. Looking at the RBM with a constant learning rate $\eta = 0.1$ we can see that for some of the iterations, approximately 80% of the proposed moves are accepted while in some of the iterations almost none are accepted. This irregular behavior is a bad sign. We expect a well performing system to have a stable acceptance rate. When applying the variable learning rate (green) the RBM performs a lot better. We see in fig. 4 that the RBM with variable learning rate converges on the exact answer after approximately 7 iterations, and it has a stable acceptance rate at approximately 80%.

Figure 5 shows the total processing time for all 100 GD iterations as the number of hidden nodes increases for all six learning rates. We can see that the time increases with increasing number of nodes up to 9, before dropping at 10 nodes. We know that 10 nodes involves more computations than 8 so we would expect it to use more time. This may be hardware related and may have

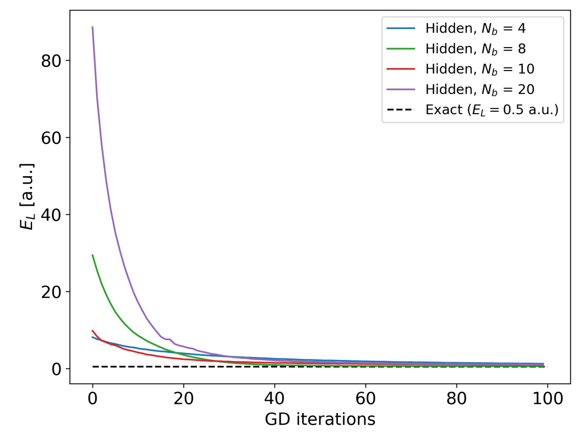

FIG. 6: Energy as a function of GD iterations using brute-force metropolis sampling with 2^{18} MC cycles for 1 particle in 1 dimension. The initial distribution of the biases and weights is $\sigma_{init} = 1$ for all RBM's and a variable learning rate with $\eta_{init} = 0.01$ and $f_\eta = 0.05$ is used.

something to do with how the PC allocates resources.

From the figure we can also see that for the three RBM's with constant learning rates (solid lines), the computing time is approximately the same when the number of hidden nodes is 5 or less. However, as we increase the number of hidden nodes the RBM with $\eta = 0.08$ (blue) is faster. This is also the fastest overall, while the RBM with a constant $\eta = 0.01$ (red) is the slowest regardless of the number of hidden nodes. When only looking at the RBM's with variable learning rates (dashed lines) the one with $\eta_{init} = 0.1$ (green) is the fastest for 6 or more hidden nodes, but it is important to keep in mind that the results for 20 hidden nodes in this case were inconclusive (see fig. 4) so this time measurement should be ignored. Using a variable learning rate with $\eta_{init} = 0.01$ shows promising results in terms of timing, but as can be seen from fig. 6 the energy does not seem to converge for any of the RBM's. In fact, the configuration that comes closest to the actual theoretical value is the one with 6 hidden nodes, where the relative error is 7% (table IX in appendix C 1). We either need to allow for more GD iterations, or choose a f_η that is higher. Overall, the combinations of hidden nodes and learning rate that gives the answer closest to the theoretical value of $E_L = 1/2$ is: 1. an RBM with constant learning rate $\eta = 0.08$ and 2 hidden nodes which yields an expectation value of the local energy of 0.50002 ± 0.00004 a.u. with a relative error of 0.003% in 183.7 seconds or 2. an RBM with 20 nodes using a variable learning rate with $\eta_{init} = 0.1$ and $f_\eta = 0.05$ which yields an expectation value of the local energy of 0.49999 ± 0.00003 a.u. with a relative error of 0.003% in 401.1 seconds. Of the two, the first configuration converges to a stable minimum energy after approximately 66 GD iterations (in 126 s) where the energy is 0.50000 ± 0.00004 a.u., while the second configuration converges after approximately 16 GD iterations (in 97.15 s) where the energy is 0.50000 ± 0.00003 a.u. and so this is the best configuration. However, we did not implement any stopping condition in our minimization algorithm meaning that when we set a number of GD iterations, the algorithm is forced to run for all of them. This in turn means that we need to set a high enough number of iterations to make absolutely sure the algorithm converges to the ground state energy, so even if the RBM with 20 nodes converges after fewer iterations, we may as well just use an RBM with fewer nodes to save time.

In fig. 6 we also see that the initial local energy is higher for 8 nodes than it is for 10 nodes. This is dependent on how the weights and biases are initialized.

E. Reproduce numbers from M. Taut

With a good understanding of many of the hyperparameters of the RBM, we now reproduce data from table 1 and figure 1 in M. Taut's work [9]. We consider a system of 2 particles in 3 dimensions for different oscillator frequencies (ω). Turn now to fig. 7, where we see local energy as a function of inverse oscillator frequency. We see a good reproduction of M. Tauts data. We find the largest difference at the largest oscillator frequency, $w = 1/4$, while all subsequent energy values are very close. The differences are listed in table III. The blocking-estimated uncertainties are as low as the fifth decimal place, reflecting the large amount of MC cycles used. We used 2^{20} cycles to produce these numbers, and we observe that the RBM produces fairly good results all the way down to 2^{12} cycles.

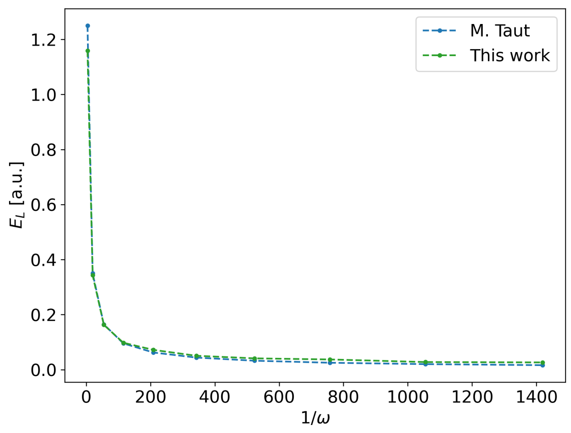

FIG. 7: Energy as a function of inverse potential oscillator frequency. Two interacting particles in 3 dimensions. 2^{20} MC cycles with 50 GD iterations, 4 hidden nodes, a variable learning rate of $\eta_{\text{init}} = 0.6$, $f_\eta = 0.05$ and an initial distribution with $\sigma_{\text{init}} = 1$. Calculations from this work is compared to data from figure 1 and table 1 of M. Taut's work [9].

TABLE III: Tabulated data from fig. 7, see figure text for details. Oscillator frequencies (ω) and $E_{L,\text{Taut}}$ are calculations from M. Taut [9].

$1/\omega$	E_L [a.u.]	$E_{L,\text{Taut}}$ [a.u.]	Diff [a.u.]
4.0	1.15870 ± 0.00051	1.25	0.09130
20.0	0.34327 ± 0.00025	0.35	0.00673
54.7386	0.16272 ± 0.00016	0.1644	0.00168
115.299	0.09776 ± 0.00010	0.0954	0.00236
208.803	0.07172 ± 0.00009	0.0622	0.00952
342.366	0.05036 ± 0.00007	0.0438	0.00656
523.102	0.04097 ± 0.00006	0.0324	0.00857
758.124	0.03692 ± 0.00005	0.025	0.01192
1054.54	0.02755 ± 0.00005	0.02	0.00755
1419.47	0.02620 ± 0.00003	0.0162	0.01000

V. CONCLUSION

In this experiment we have used variational MC with a Restricted Boltzmann Machine (RBM) to calculate the ground state energy of systems of one and two particles confined in a pure isotropic harmonic oscillator potential in one, two, and three dimensions. In using Metropolis-Hastings sampling, we found that importance sampling gave no clear advantage over the brute-force approach. The two methods produced approximately the same results and uncertainties at the same number of MC cycles, and brute-force, being less computationally heavy, used less time than importance sampling. Based on these findings, we chose the brute-force approach as our tool for the rest of the results.

We were fairly successful in reproducing the analytical ground state energies of the non-interacting systems of one and two particles, in one, two, and three dimensions. We found that with all biases and weights initially set to zero, the wave function used in the RBM reduces to the exact wave function of the ground state, yielding precisely the analytical results. Initializing the weights and biases with values drawn from a normal distribution with $\mu = 0$ and $\sigma_{\text{init}} = 0.5$ we found that the RBM was at most 0.038 a.u. (for two particles in three dimensions) from the exact answer, but with a blocking-estimated uncertainty lower than the difference, ± 0.001 a.u. for that case. In future work it would be interesting to explore more on how the initialization of weights and biases could affect the end results. Presently they are all initialized in the same way, with a normal distribution. But we are curious to see what may change if we use different types of distributions.

We found that using a variable learning rate which decreases with increasing GD iterations is helpful in specific cases. Our results show that when choosing a learning rate as large as possible, a variable learning rate of $\eta_{\text{init}} = 0.6$ and $f_\eta = 0.05$ for a one particle one dimension system with $\sigma_{\text{init}} = 4$ did improve the convergence time with 1 - 2 GD iterations. Though, we were able to converge faster on the correct answer by choosing another, constant, learning rate.

We found that RBM's with a higher number of hidden nodes requires a lower learning rate. The configuration that achieved the most accurate (the closest to theory) value for the ground state energy in the least amount of GD iterations is with 20 hidden nodes and a variable learning rate of $\eta_{init} = 0.1$ with $f_\eta = 0.05$. As we increase the number of hidden nodes the processing time grows, so in future studies it will be beneficial to include a stopping condition in the GD method so that we will be able to stop the RBM when it has reached the minimum energy.

In the final part of our results, we successfully reproduced ground state energies for interacting systems of two particles in three dimensions. We compared our data to M. Taut [9] and found a good match for most oscillator frequencies. The largest difference of $E_{\text{diff}} = 0.09130$ a.u. was present in the energy for the largest oscillator frequency of $\omega = 1/4$.

[1] Bishop, C. M. 2006, Pattern Recognition and Machine Learning (Springer)
[2] Carleo, G., & Troyer, M. 2017, Science, 355, 602, doi: 10.1126/science.aag2302
[3] Dahl, J. K., Hardersen, A., & Sønderland, P.-D. 2021, A Variational Monte Carlo Analysis of a Bose-Einstein Condensate. https://github.com/JonKDahl/FYS4411_V21/blob/main/project1/doc/project_1_fys4411.pdf
[4] Hjorth-Jensen, M. 2020, Computational Physics II FYS4411, lecture notes, Boltzmann Machines. http://compphysics.github.io/ComputationalPhysics2/doc/LectureNotes/_build/html/boltzmannmachines.html
[5] Jonsson, M. 2018, Phys. Rev. E, 98, 043304, doi: 10.1103/PhysRevE.98.043304
[6] Krane, K. S. 1987, Introductory Nuclear Physics, 3rd edn. (Wiley)
[7] Melko, R., Carleo, G., Carrasquilla, J., & Cirac, J. 2019, Nature Physics, 15, doi: 10.1038/s41567-019-0545-1
[8] Slater, J. C. 1929, Phys. Rev., 34, 1293, doi: 10.1103/PhysRev.34.1293
[9] Taut, M. 1993, Phys. Rev. A, 48, 3561, doi: 10.1103/PhysRevA.48.3561

Appendix A: Local Energy

As a reminder, the NQS wave function is given in equation (19) as,

$$\psi(X) = \frac{1}{Z} e^{-\sum_i \frac{(x_i - a_i)^2}{2\sigma^2}} \prod_j \left(1 + e^{b_j + \sum_i \frac{x_i w_{ij}}{\sigma^2}}\right) = \frac{1}{Z} e^{-\sum_i \frac{(x_i - a_i)^2}{2\sigma^2}} \prod_j (1 + e^{v_j}) \tag{A1}$$

Where we have, for the purpose of notation simplicity, defined the exponent as,

$$v_j = b_j + \sum_i \frac{x_i w_{ij}}{\sigma^2} \tag{A2}$$

The local energy is defined in equation (24) of section section III A is defined as $E_L = \frac{1}{\psi}\hat{H}\psi$. So with the Hamiltonian defined in equation (1) as

$$\hat{H} = \sum_i \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right) + \sum_{i<j} \frac{1}{r_{ij}} \tag{A3}$$

We can express the local energy in the following form,

$$E_L = \frac{1}{\psi} \sum_i (-\frac{1}{2}\nabla_i^2 \psi) + \sum_i \frac{1}{2}\omega^2 r_i^2 + \sum_{i<j} \frac{1}{r_{ij}} \tag{A4}$$

The first term represents the kinetic part of the local energy and this is what we need to find an analytic expression for. Using the relation $\nabla_i \ln\psi = \frac{1}{\psi}\nabla_i \psi$ we have,

$$\begin{aligned}
\nabla_k^2 \ln\psi &= \nabla_k \frac{1}{\psi} \nabla_k \psi \\
&= \left(\nabla_k \frac{1}{\psi}\right)(\nabla_k \psi) + \frac{1}{\psi}\nabla_k^2 \psi \\
&= -\frac{1}{\psi^2}(\nabla_k \psi)^2 + \frac{1}{\psi}\nabla_k^2 \psi \\
&= -(\nabla_k \ln\psi)^2 + \frac{1}{\psi}\nabla_k^2 \psi
\end{aligned} \tag{A5}$$

Which leads to the useful relation

$$\frac{1}{\psi}\nabla_k^2 \psi = \nabla_k^2 \ln\psi + (\nabla_k \ln\psi)^2 \tag{A6}$$

Inserting this into (A4) we have

$$E_L = \sum_i \frac{1}{2}\left(-(\nabla_i \ln\psi)^2 - \nabla_i^2 \ln\psi + \omega^2 r_i\right) + \sum_{i<j} \frac{1}{r_{ij}} \tag{A7}$$

which is the expression given in equation (26) of section III A. We need to find the analytic expressions for $\nabla_k^2 \ln\psi$ and $\nabla_k \ln\psi$. The logarithm of the NQS wave function is,

$$\ln\psi = -\ln Z - \sum_i \frac{(x_i - a_i)^2}{2\sigma^2} + \sum_j \ln\left[1 + e^{v_j}\right] \tag{A8}$$

And the derivatives are

$$\nabla_k \ln \psi = -\frac{x_k - a_k}{\sigma^2} + \sum_i \frac{w_{kj}}{\sigma^2} \frac{e^{b_j + \sum_i \frac{x_i w_{ij}}{\sigma^2}}}{1 + e^{b_j + \sum_j \frac{x_i w_{ij}}{\sigma^2}}} \tag{A9}$$

$$\nabla_k^2 \ln \psi = -\frac{1}{\sigma^2} + \frac{1}{\sigma^4} \sum_j w_{kj}^2 \frac{e^{b_j + \sum_i \frac{x_i w_{ij}}{\sigma^2}}}{(1 + e^{b_j + \sum_i \frac{x_i w_{ij}}{\sigma^2}})^2} \tag{A10}$$

Appendix B: Gradient

As described in section III D the partial derivative of the expectation value of the local energy wrt. the parameters $\alpha_i = a_1, ..., a_M, b_1, ..., b_N, w_{11}, ..., w_{MN}$ can be calculated as,

$$G_i = \frac{\partial \langle E_L \rangle}{\partial \alpha_i} = 2 \left[\langle E_L \frac{1}{\psi} \frac{\partial \psi}{\partial \alpha_i} \rangle - \langle E_L \rangle \langle \frac{1}{\psi} \frac{\partial \psi}{\partial \alpha_i} \rangle \right] \tag{B1}$$

Using the relation $\frac{1}{\psi}\frac{\partial \psi}{\partial \alpha_i} = \frac{\partial \ln \psi}{\partial \alpha_i}$ we can write this as,

$$G_i = 2 \left[\langle E_L \frac{\partial \ln \psi}{\partial \alpha_i} \rangle - \langle E_L \rangle \langle \frac{\partial \ln \psi}{\partial \alpha_i} \rangle \right] \tag{B2}$$

Using the NQS wave function in (19) whose logarithm is given in eq. A8, we find

$$\frac{\partial \ln \psi}{\partial a_k} = \frac{x_k - a_k}{\sigma^2} \tag{B3}$$

$$\frac{\partial \ln \psi}{\partial b_k} = \frac{e^{b_k + \sum_i \frac{x_i w_{ik}}{\sigma^2}}}{1 + e^{b_k + \sum_i \frac{x_i w_{ik}}{\sigma^2}}} \tag{B4}$$

$$\frac{\partial \ln \psi}{\partial w_{kl}} = \frac{x_k}{\sigma^2} \frac{e^{b_l + \sum_i \frac{x_i w_{il}}{\sigma^2}}}{1 + e^{b_l + \sum_i \frac{x_i w_{il}}{\sigma^2}}} \tag{B5}$$

Appendix C: Tables

1. Dependence on number of hidden nodes

TABLE IV: Expectation values for the ground state energy for 1 particle in 1 dimension calculated using $N = 2^{18}$ MC cycles with 100 GD iterations using a constant learning rate of $\eta = 0.1$. We use a initial distribution of the biases and weights with $\sigma_{init} = 1$.

No. hidden	E_L [a.u.]	Rel. Diff [%]	time [s]
2	0.50006 ± 0.00004	0.01	180.31
3	0.5024 ± 0.0001	0.49	210.46
4	0.5039 ± 0.0002	0.78	239.29
5	0.5046 ± 0.0002	0.93	274.79
6	0.5024 ± 0.0002	0.50	290.98
7	0.5043 ± 0.0002	0.87	322.32
8	0.5047 ± 0.0002	0.96	347.91
9	0.5061 ± 0.0002	1.23	358.34
10	0.5055 ± 0.0002	1.11	287.77
20	78.28 ± 0.02	15556.65	415.48

TABLE V: Expectation values for the ground state energy for 1 particle in 1 dimension calculated using $N = 2^{18}$ MC cycles with 100 GD iterations using a constant learning rate of $\eta = 0.08$. We use a initial distribution of the biases and weights with $\sigma_{init} = 1$.

No. hidden	E_L [a.u.]	Rel. Diff [%]	time [s]
2	0.50002 ± 0.00004	0.003	183.71
3	0.5028 ± 0.0002	0.58	214.40
4	0.5049 ± 0.0002	0.99	245.34
5	0.5061 ± 0.0002	1.22	281.11
6	0.5038 ± 0.0002	0.77	277.71
7	0.5054 ± 0.0002	1.10	308.43
8	0.5065 ± 0.0002	1.30	329.50
9	0.5085 ± 0.0003	1.70	338.17
10	0.5082 ± 0.0003	1.66	282.66
20	0.50012 ± 0.00005	0.02	397.02

TABLE VI: Expectation values for the ground state energy for 1 particle in 1 dimension calculated using $N = 2^{18}$ MC cycles with 100 GD iterations using a constant learning rate of $\eta = 0.01$. We use a initial distribution of the biases and weights with $\sigma_{init} = 1$.

No. hidden	E_L [a.u.]	Rel. Diff [%]	time [s]
2	0.60294 ± 0.0007	20.59	181.81
3	0.50833 ± 0.0003	1.67	211.67
4	0.52262 ± 0.0004	4.52	243.99
5	0.56266 ± 0.0007	12.53	270.22
6	0.51876 ± 0.0004	3.75	305.19
7	0.58834 ± 0.0009	17.67	334.41
8	0.55649 ± 0.0007	11.30	363.23
9	0.60728 ± 0.0009	21.46	381.87
10	0.63503 ± 0.001	27.01	301.26
20	0.60830 ± 0.0009	21.66	430.54

TABLE VII: Expectation values for the ground state energy for 1 particle in 1 dimension calculated using $N = 2^{18}$ MC cycles with 100 GD iterations using a variable learning rate with $\eta_{init} = 0.1$ and $f_\eta = 0.05$. Initial distribution of the biases and weights is $\sigma_{init} = 1$.

No. hidden	E_L [a.u.]	Rel. Diff [%]	time [s]
2	0.50012 ± 0.00004	0.02	193.22
3	0.5047 ± 0.0002	0.96	224.84
4	0.5096 ± 0.0003	1.92	252.29
5	0.5164 ± 0.0004	3.29	278.41
6	0.5065 ± 0.0002	1.30	272.17
7	0.5163 ± 0.0004	3.27	301.98
8	0.5146 ± 0.0004	2.93	330.70
9	0.5194 ± 0.0004	3.88	348.09
10	0.5194 ± 0.0004	3.89	283.84
20	0.49999 ± 0.00003	0.003	401.11

TABLE VIII: Expectation values for the ground state energy for 1 particle in 1 dimension calculated using $N = 2^{18}$ MC cycles with 100 GD iterations using a variable learning rate with $\eta_{init} = 0.08$ and $f_\eta = 0.05$. Initial distribution of the biases and weights is $\sigma_{init} = 1$.

No. hidden	E_L [a.u.]	Rel. Diff [%]	time [s]
2	0.50035 ± 0.00006	0.07	190.15
3	0.5053 ± 0.0002	1.06	220.25
4	0.5116 ± 0.0003	2.33	259.52
5	0.5212 ± 0.0004	4.25	284.03
6	0.5091 ± 0.0003	1.82	301.74
7	0.5201 ± 0.0004	4.03	330.29
8	0.5198 ± 0.0004	3.96	362.10
9	0.5259 ± 0.0005	5.19	384.55
10	0.5305 ± 0.0005	6.11	309.03
20	0.5038 ± 0.0002	0.78	440.79

TABLE IX: Expectation values for the ground state energy for 1 particle in 1 dimension calculated using $N = 2^{18}$ MC cycles with 100 GD iterations using a variable learning rate with $\eta_{init} = 0.01$ and $f_\eta = 0.05$. Initial distribution of the biases and weights is $\sigma_{init} = 1$.

No. hidden	E_L [a.u.]	Rel. Diff [%]	time [s]
2	1.90545 ± 0.002	281.09	179.65
3	0.58362 ± 0.0006	16.72	204.51
4	1.26897 ± 0.002	153.79	237.37
5	0.61574 ± 0.001	23.15	263.07
6	0.53498 ± 0.0005	7.00	260.21
7	0.76713 ± 0.001	53.43	313.00
8	0.61836 ± 0.001	23.67	337.56
9	0.87659 ± 0.002	75.32	357.50
10	0.99661 ± 0.002	99.32	302.52
20	1.03545 ± 0.002	107.09	403.33