Final version May 31, 2019

Preprint typeset using LaTeX style emulateapj v. 12/16/11

MACHINE LEARNING APPROACH APPLIED TO SOLUTION OF QUANTUM MECHANICAL MANY BODY SYSTEM PROJECT 2

Maria L. Markova and Vala M. Valsdóttir

Final version May 31, 2019

ABSTRACT

The present project is devoted to the simulation of a quantum dot and reproduction of its ground state energy by means of a neural generative network (NN). For this aim the restricted Boltzmann machine (RBM) was used. The application of the RBM implied learning a probability distribution (or a wave function) subsequently used for generation of a final output. The Variational Monte Carlo method coupled to the gradient descent (1) was used in order to obtain the ground state of the system. The brute force method, the importance sampling as well as the Gibbs sampling were tested in the framework of the VMC method.

The system of one or two electrons is considered in one and two dimensions with and without the Coulomb repulsion taken into account. The results obtained were compared with the the analytical benchmarks and were shown to reproduce them accurately for the brute force and importance sampling. Similar results for the Gibbs sampling are able to reproduce it only in case of proper standard deviation chosen for underlying distribution. All the results were shown to be strongly dependent on initial guesses of exploited parameters, such as characteristics of a neural network and learning process. Additional analysis of errors was carried out to improve statistical errors mentioned.Certain improvements of the code were made, including the MPI based parallelization of calculation processes.

Subject headings: Machine learning — Neural Network — Metropolis algorithm — Blocking Method — Local Energy — Gradient Descent Method — Variational Monte Carlo

1. INTRODUCTION

The N electron system in a given number of dimensions and certain potential gives rise to numerous problems to reflect on and to be solved numerically. This problems are being solved alongside the further development of various methods applied. However, some electron systems can also provide us with analytical solution for the corresponding Schrdinger equation. The simplest examples such as an electron in the Coulomb field of an atomic nucleus or an electron in the harmonic oscillator are those basic bricks our knowledge of quantum mechanics is build on. In the present project the attention is payed to the system of two electrons in an external harmonic oscillator potential. In this case the six dimensional Schrdinger equation might be reduced to find the real roots of a polynomial and provide us with analytical solution for the ground state for the chosen frequencies of harmonic oscillator (2). The Hamiltonian of such system can be presented as following (2):

$$H = -\frac{1}{2}\nabla_1^2 + \frac{1}{2}\omega^2\mathbf{r}_1^2 - \frac{1}{2}\nabla_2^2 + \frac{1}{2}\omega^2\mathbf{r}_2^2 + \frac{1}{|\mathbf{r}_1 - \mathbf{r}_2|}, \quad (1)$$

maria.markova@fys.uio.no,
vala.m.valsdottir@fys.uio.no
[1] Department of Physics, University of Oslo, P.O. Box 1048 Blindern, N-0316 Oslo, Norway

or choosing new variables $\mathbf{r} = \mathbf{r}_2 - \mathbf{r}_1$ and $\mathbf{R} = \frac{1}{2}(\mathbf{r}_1 + \mathbf{r}_2)$:

$$H = -\nabla_r^2 + \frac{1}{4}\omega^2\mathbf{r}_1^2 + \frac{1}{r} - \frac{1}{4}\nabla_R^2 + \omega^2\mathbf{R}^2 = H_r + H_R. \quad (2)$$

The corresponding wave function in the form:

$$\psi(1,2) = \phi(\mathbf{r})\xi(\mathbf{R})\chi(s_1, s_2). \quad (3)$$

As it is required by the Pauli principle, this combination should be antisymmetric, so that the solution picked has definite parity combined with the corresponding spin part.

The problem described has a large number of applications, for example, it might serve as a test for certain numerical algorithms for similar problem of N electrons subject to the Coulomb correlation in a given potential. The quantum dot itself is a widely used object which might be found in use in a large range of scientific areas. For instance, the optical properties of quantum dots make them suitable for immunolabelling, molecular imaging, and biomedical analysis (3; 4). Thus, the well developed methods of simulation of such systems might significantly facilitate technological breakthrough in some fields. The restricted Boltzmann machine was used in the present project to build up a suitable neural quantum state (NQS) to simulate the quantum dot system and find the corresponding ground state. The evolu-

tion of system parameters starts from an initial guess and is governed by the gradient descent method with the base of the variational Monte Carlo and described thoroughly in the section 3. Three different sampling methods such as brute force approach, importance sampling algorithm and Gibbs sampling were chosen and discussed in subsections 3.2.1,3.2.2, and 3.2.3.The analysis of the errors is performed by the blocking method and described in 3.4. The code is presented in details in the section 4. The results obtained are presented in section 5 and followed by the detailed discussion in section 6.

2. THEORETICAL BACKGROUND

2.1. *Review of the system*

The framework for the ground state search in the present project is set by P electrons in D dimensions moving in a harmonic oscillator potential. Such system can be described with the corresponding Hamiltonian:

$$\hat{H} = \sum_{i=1}^{N} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right) + \sum_{i<j} \frac{1}{r_{ij}}, \tag{4}$$

here the natural units were applied ($\hbar = c = e = m_e = 1$), so that the energy will be given in atomic units a.u., ω is a chosen frequency of a harmonic oscillator, $r_i = \sqrt{x^2 + y^2 + z^2}$, and the last term corresponds to the Coulomb repulsion of electrons on the distance $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$. However, maximum 2 dimensions and two electrons are going to be considered in the present project, since the analytical benchmarks are available for this case.

The wave function of the system considered is built so, that the Pauli principle is fulfilled. Indeed, the electrons as the fermions are restricted from occupying the same quantum state in the same quantum system. This fact is reflected by asymmetric nature of the wave function under exchange of coordinates of particles. And for the case of two electrons the wave function is given by the Slater determinant comprising of the single electron wave functions:

$$\Psi_{12}(1,2) = \begin{vmatrix} \phi_1(\mathbf{r}_1) & \phi_2(\mathbf{r}_1) \\ \phi_1(\mathbf{r}_2) & \phi_2(\mathbf{r}_2) \end{vmatrix} = \phi_1(\mathbf{r}_1)\phi_2(\mathbf{r}_2) - \phi_2(\mathbf{r}_1)\phi_1(\mathbf{r}_2). \tag{5}$$

In the harmonic oscillator in two dimensions in the non-interacting case , the spatial wave function can be expressed in terms of the Hermite polynomials $H_{n_x}(\sqrt{\omega x})$:

$$\Phi_{n_x,n_y}(x,y) = AH_{n_x}(\sqrt{\omega x})H_{n_y}(\sqrt{\omega y})exp(-(x^2+y^2)/2), \tag{6}$$

with the lowest state energy :

$$\epsilon_0 = \omega(n_x + n_y + 1) = \omega. \tag{7}$$

Since only two electrons are considered in the ground state, their spins must correspond to $\frac{1}{2}$ and $-\frac{1}{2}$ and make the antysimmetric component of the wave function 3. Hence, the spatial component of the wave function determined should be symmetric and can be further simplified as:

$$\Phi(x,y) = Cexp(-(x^2+y^2)/2), \tag{8}$$

where C is normalization constant. These considerations form the base for the further search of an optimal form of the wave function of the ground state.

2.2. *Machine learning approach for solving the problem*

In contrast to the problem considered in the project 1 (5), the aim of the project 2 is to show that the wave function might be successfully represented by the means of neural networks and machine learning.

The idea of the machine learning has been successfully applied for solving a wide range of problems from optical character recognition to natural language processing (6). And it rapidly becomes more and more tempting to apply it for solving complicated quantum mechanical problems as well. The whole concept of machine learning implies that a machine (computer) is made to modify its own actions to aim the greater accuracy, i.e. the chosen actions will become closer and closer to the correct actions which are expected (7). We could say that a machine is taught how to achieve a certain result given that it does not know initially what it should reproduce as an output. The ML approach arises from the combination of statistics, mathematics, physics, and basic principles of biology and neuroscience and refers to such property of intelligence as reasoning and logical deduction, learning and adapting.

The process of improvement through the practice might be performed in different strategies, such as supervised and unsupervised learning, reinforcement and evolutionary learning. The former implies that the correct solution is known and a machine tries to deduce the logic of how the result is obtained. The unsupervised learning is based on the lack of prior information on the system and all the relationships between its parts or characteristics should be found by machine itself. The intermediate type between these two, or reinforcement learning, is based on the critic introduced. In other words, the system is told that the result is wrong, but is not given any hints on how to improve. Hence, different possibilities should be tested unless the result is correct (certain tolerance should be included). On the other hand, the evolutionary learning provides a machine with a feedback on how good a current result is.

The machine learning approach is often used to provide an ansatz for the variational Monte Carlo calculations. It might be crucial for fermion systems where the applications of neural networks eliminates the sign problem difficulties (8). The present project considers an example of ML-VMC symbiosis - the application of the restricted Boltzmann machine as the ground state VMC wave function. Similar approach was recently presented in (9).

2.3. *Neural networks as a computational model*

The human brain has always been a thrilling mystery bothering our minds and inspiring not only specialists in neurophysiology, but mathematicians as well. As the result, the latter have posed a question: "The nature has created such a sophisticated and effective system, why can't we benefit from it? Why can't we use the idea the human brain is based on to make other systems to operate similarly and solve the problems we are interested in?" However, this inevitably requires a suitable mathematical framework. The basics of such theory were first mentioned by W. S. McCulloch and W. H. Pitts in 1943, who managed to describe the character of nervous activity, neural events and corresponding relations between them in terms of propositional logic (10).

Each neuron in a brain possesses a soma and axon which form junctions (synapses) with those for surrounding neurons. A certain threshold governs the propagation of a signal: if an excitation exceeds this threshold, the impulse of electrical signal is initiated and propagated further, yielding an output.Otherwise, the excitation yields a zero output.

Form the perspective of mathematics, neurons can be represented by separate nodes in a layer, interrelated and communicating so than one can build the series of connected layers transferring a signal and modifying it to get an appropriate output. This fact might be written in the following the form (11):

$$y = f\left(\sum_{i=1}^{n} \omega_i x_i + b_i\right) = f(u_i), \tag{9}$$

where y is an output of the neuron, $x_1, x_2, ..., x_n$ are the input signals which are modified with the corresponding weighting factors ω_i and biased with b_i. The f function is called the activation function and it should fulfill the requirements. Considering the simplest type of a neural network, a feed-forward neural network (FFNN),where an information moves in one direction only, the activation function should be non-constant, bounded, monotonically-increasing, and continuous.

The most commonly used structure of a neural network implies at several types of layers: an input layer, an output layer and a certain number of hidden layers in between. Each layer comprises of a chosen number of nodes connected to the nodes in other layers and/or each other. The signal transferred from a node to a node is weighted, i.e. a node would transfer a certain value proportional to the input signal, and gradual change of weights as well as biases would lead to a changing output. Here the idea of learning is actually applied to neural networks. The gradual update of weights and biases during the learning process allows system to fit the required result better. The degree of how good our system reproduces the result is reflected by the cost function, which gives the total error of our network across all the samples. Thus, the while training process aims at finding the optimal parameters (weights and biases) that minimize a chosen cost function. In the case of this project,

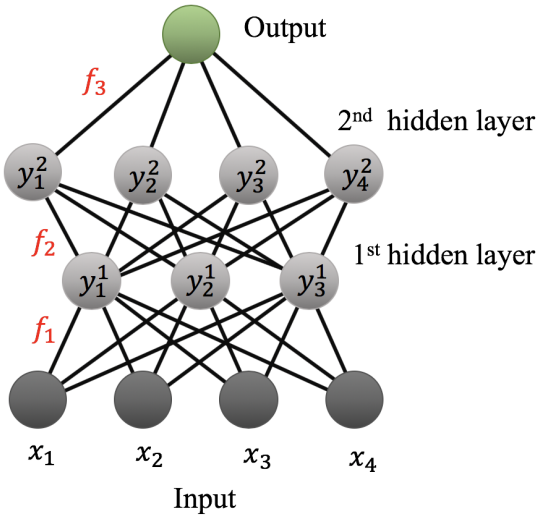

FIG. 1.— An example of a neural network with two hidden layers.

the cost function is given by an expectation value of the energy of a trial wave function, and such minimization leads to the ground state of the system. This cost function will be described in more details in the subsequent sections.

2.4. *Restricted Boltzmann machine*

The generative models are widely used for unsupervised solving quantum mechanical problems, since they are able to represent and sample from one of the most important quantum mechanical properties of a system - a probability distribution. This probability distribution is subsequently used to calculate the values of interest.The restricted Boltzman machine was recently and successfully used by G. Carleo and M. Toyer (9), where it was applied to the quantum mechanical spin lattice system in the framework of the Ising model and Heisenberg model. The present project demonstrates another example of an application of the restricted Boltzmann machine. the RBM provides us with two basic types of layers: the visible layer presented by the M dimensional vector $\mathbf{x}$ and the hidden layer $\mathbf{h}$ in N dimensions. The former might reflect either an input system is provided with or an output it should reconstruct. In its own turn, the latter aims to perform the "improvement"of calculation itself. The system of layers is interrelated by the interaction weights, which is a matrix $M \times N$. Both types of layers are biased with corresponding visible bias $\mathbf{a}$ of the same dimension as $\mathbf{x}$ and the hidden N-dimensional bias $\mathbf{b}$. The term "restricted" refers to an absence of connections between the different nodes within the same layer, however, each node in the visible layer related to each layer in the hidden layer (see FIG 2).

For the present project the Gaussian-binary RBM is chosen. The energy function in this case is (for M nodes

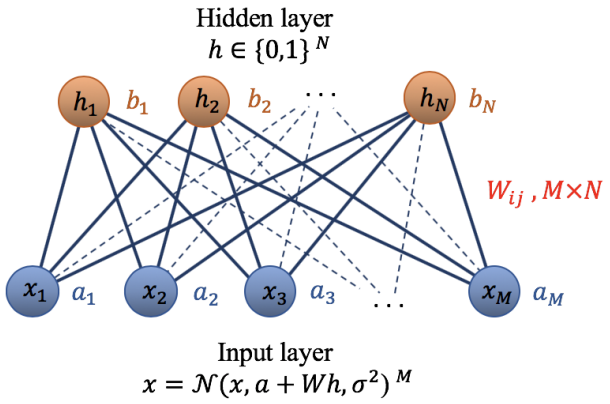

FIG. 2.— The chosen Gaussian-binary Boltzmann machine with arbitrary number of nodes within the visible layer M and the hidden layer N.

in a visible layer and N nodes in a hidden layer):

$$E(\mathbf{x},\mathbf{h}) = \sum_i^M \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_j^N b_j h_j - \sum_{i,j}^{N,M} \frac{x_i \omega_{ij} h_j}{\sigma_i^2}, \quad (10)$$

and can be used in the joint distribution :

$$F_{rbm}(\mathbf{x},\mathbf{h}) = \frac{1}{Z} exp(-\frac{1}{T_0} E(\mathbf{x},\mathbf{h})), \quad (11)$$

where the normalization constant Z can be expressed as:

$$Z = \int\int exp(-\frac{1}{T_0} E(\mathbf{x},\mathbf{h})) d\mathbf{x} d\mathbf{h}. \quad (12)$$

The temperature dependence is excluded in a large variety of cases, it the present project it is set to 1.

The joint probability might be used to obtain the marginal probability:

$$F_{rbm}(\mathbf{x}) = \sum_{\mathbf{h}} F_{rbm}(\mathbf{x},\mathbf{h}) = \frac{1}{Z} \sum_{\mathbf{h}} exp(-E(\mathbf{x},\mathbf{h})) \quad (13)$$

This marginal probability allows us to define the wave function of the system called neural network quantum state (NQS):

$$\begin{aligned}
\Psi(\mathbf{x}) &= F_{rbm}(\mathbf{x}) = \\
&= \frac{1}{Z} \sum_{h_j} exp(-E(\mathbf{x},\mathbf{h})) = \\
&= \frac{1}{Z} \sum_{h_j} exp(\sum_i^M \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_j^N b_j h_j - \sum_{i,j}^{N,M} \frac{x_i \omega_{ij} h_j}{\sigma_i^2}) = \\
&= \frac{1}{Z} exp(\sum_i^M \frac{(x_i - a_i)^2}{2\sigma_i^2}) \prod_j^N (1 + exp(b_j + \sum_i^M \frac{x_i \omega_{ij}}{\sigma^2})),
\end{aligned} \quad (14)$$

this form of the wave function underlies in the following calculations.

2.5. *Local energy of the system and its evaluation*

The whole study evolves around finding the ground state energy, luckily it can be calculated analytically in this case, as it is shown in appendixes B and C (10,11). The variational principle claims that the $E_{exact} \leq \langle E \rangle$. Where $\langle E \rangle$ is the expectation value of the local energy as a function of the variational parameter α. The task at hand is then to find the minimum of the expectation value for the energy, and thus an estimation of the ground state energy. So for a function $f(x)$ the standard statistical definition of the expectation value is

$$\langle f \rangle = \int f(x) p(x) dx, \quad (15)$$

but the quantum mechanical value for the expectation value of the energy is

$$\langle E \rangle = \langle H \rangle = \frac{\int \Psi^*(\vec{r}) H \Psi(\vec{r}) d\vec{r}}{\int \Psi^*(\vec{r}) \Psi(\vec{r}) d\vec{r}}. \quad (16)$$

To make the expectation value for the energy look more like the standard statistical definition stated above, the local energy is defined as

$$E_L(\vec{r}) = \frac{1}{\Psi_T(\vec{r})} \hat{H} \Psi_T(\vec{r}). \quad (17)$$

This enables us to write the energy as

$$\langle E \rangle = \int |\Psi_T^2| \, E_L d\vec{r}. \quad (18)$$

Since the study involves large numbers, Bernoullis law can be applied and the integral thus becomes

$$\langle E \rangle = \int \Psi_T^2 E_L(r) dr \approx \frac{1}{M} \sum_{i=1}^n E(\vec{R}_i, \vec{\alpha}), \quad (19)$$

where the number M is the number of Monte Carlo cycles.

The further problems one faces are related to the analytical calculation of the local energy for a specific case of P particles in D dimensions. The visible layer will then be a vector of dimension $M = P \cdot D$ and the hidden layer will be a vector of chosen length N. The local energy can be represented as follows:

$$E_L = \sum_{k=1}^M (E_{kin}^k + E_{pot}^k) + \sum_{j>i} E_{int}^{ij}, \quad (20)$$

For the first two cases considered, the brute force approach and the importance sampling, the derivation of the kinetic energy part with the wave function chosen (14) will take the following form:

$$E_{kin}^k = -\frac{1}{2} \frac{1}{\Psi_T} \nabla_k \Psi_T = -\frac{1}{2} ((\nabla_k ln \Psi_T)^2 + \nabla_k^2 ln \Psi_T), \quad (21)$$

where the components of the gradient and the Laplasian are:

$$\frac{\partial}{\partial x_k} \ln \Psi_T = -\frac{x_k - a_k}{\sigma^2} + \sum_j^N sigmoid(u(j)) \frac{w_{kj}}{\sigma^2} \quad (22)$$

$$\frac{\partial^2}{\partial x_k^2}\ln\Psi_T = -\frac{1}{\sigma^2}+\sum_j^N sigmoid(-u(j))\times sigmoid(u(j))\times\frac{w_{kj}^2}{\sigma^4} \quad (23)$$

For the sake of simplification the logistic sigmoid function is considered:

$$sigmoid(u(j)) = \frac{1}{1+e^{-b_j-\sum_{i=1}^M \frac{x_i w_{ij}}{\sigma^2}}} \quad (24)$$

with $u(j) = -b_j - \sum_{i=1}^M \frac{x_i w_{ij}}{\sigma^2}$. Other details of derivation of the local energy are presented in 10 and 11.

2.6. *Analytical expressions for local energy*

The form of the spatial component of the wave function considered in the previous sections might be rewritten in the following form for two dimensional case and one electron:

$$\Phi_{n_x,n_y}(x,y) = AH_{n_x}(\sqrt{\omega x})H_{n_y}(\sqrt{\omega y})exp(-\frac{(x^2+y^2)}{2}) = \\ = \Phi_{n_x}(x)\cdot\Phi_{n_y}(y). \quad (25)$$

Being limited with the case of one dimension and one electron, the eigenvalues of the Hamiltonian (4) are dependent on the principal quantum number n_x for one dimension:

$$E_{n_x} = \hbar\omega(n_x+\frac{1}{2}) = \omega(n_x+\frac{1}{2}). \quad (26)$$

Let's find an expression for the eigenvalues of the Hamiltonian (4) for two dimensional case:

$$H\Phi_{n_x,n_y} = H\Phi_{n_x}\cdot\Phi_{n_y} + H\Phi_{n_y}\cdot\Phi_{n_x} = \\ = E_{n_x}\Phi_{n_x,n_y} + E_{n_y}\Phi_{n_x,n_y} = \omega(n_x+n_y+1), \quad (27)$$

thus, the ground state energy will correspond to $n_x = n_y = 0$ and yield:

$$E_0 = \omega. \quad (28)$$

For the two electron system in two dimensions the wave function can be presented by the composition of single particle wave functions for both electrons:

$$\Phi_{1,2}(\mathbf{r}_1,\mathbf{r}_2) = \Phi_1(\mathbf{r}_1)\cdot\Phi_2(\mathbf{r}_2) = \Phi_{1\ n_{x_1},n_{y_1}}\cdot\Phi_{2\ n_{x_2},n_{y_2}}, \quad (29)$$

and hence the eigenvalues of the Hamiltonian are:

$$H\Phi_{1,2} = H\Phi_1\cdot\Phi_2 + H\Phi_2\cdot\Phi_1 = \\ = E_1\Phi_{1,2} + E_2\Phi_{1,2} = \\ = \omega(n_{x_1}+n_{y_1}+n_{x_2}+n_{y_2}+2), \quad (30)$$

and the corresponding ground state energy is:

$$E_0 = 2\omega. \quad (31)$$

These expressions are used as the benchmarks for the results presented in the project, since the maximum number of dimensions and particles considered will not exceed 2 dimensions and particles correspondingly.

3. METHOD

3.1. *Variational Monte Carlo method*

The numerical Monte Carlo methods are the the widely used in the large variety of areas (physics, chemistry, biology,engineering etc).These methods imply the statistical simulation which allows to omit writing all differential equations that describe a system and its dynamics, since the process is simulated directly. If the given system can be described by a certain probability distribution functions, the random sampling based on these functions allows to pass through the number of steps and achieve the solution. However, the separate results of these samplings should be accumulated the way the final result will be obtained in the most efficient way, that is why Monte Carlo technique should be based on a combination of some sampling technique and a selection algorithm, or, in the present work, the Metropolis algorithm which governs the movement towards the most probable result. Unless this algorithm is implemented the step sizes and directions for movement of components of the system (particles in our case) are independent and random, thus the movement towards the equilibrium state is much less efficient.

The number of particles can be large, and one will not be able to obtain the exact solutions of a given many-body problem. The conventional integration methods (e.g. Gauss-Legendre method) are not efficient when the large number of dimensions is involved. In this case the quantum Monte Carlo methods are an excellent tool for studying of quantum mechanical systems or calculation various expectation values of interest i.e. calculation of multi-dimensional integrals. The method used in the present work is variational Monte Carlo approach for the estimation of the ground state energy of a system of N particles. Basing on the variation principle which for the given Hamiltonian and a trial wave function $\Psi_T(\mathbf{R},\alpha)$ claims (1):

$$E[H(\alpha)] = \frac{\int d\mathbf{R}\Psi_T^*(\mathbf{R},\alpha)\hat{H}(\mathbf{R})\Psi_T(\mathbf{R},\alpha)}{\int d\mathbf{R}\Psi_T^*(\mathbf{R},\alpha)\Psi_T(\mathbf{R},\alpha)} \geq E_0, \quad (32)$$

here E_0 is the true ground state energy of a system,α is an arbitrary parameter or a set of parameters. This scheme implies the following principal steps:

1. Initialization of the trial wave of the system function $\Psi_T(\mathbf{R},\alpha)$ dependent on the variational parameter which is assumed to be as close to the true value as possible. Fixing of the number of Monte Carlo cycles and number of variational parameters and setting of initial position of particles for a given α.

2. Initialization of the energy and variance and start of the Monte Carlo cycles.

(a) Proposal of the new step for one particle or all particles $\mathbf{R}' = \mathbf{R} + step \cdot r$, where r is a random variable $\in [0,1]$.

(b) Calculation of the trial wave function and the probability distribution function for the new position.

(c) Test of the new position in the Metropolis algorithm, the step is either rejected or accepted.

(d) If the step is accepted, the initial position is assigned with the proposed step $\mathbf{R} = \mathbf{R}'$.

(e) Update of the energy and variance.

3. End of Monte Carlo cycles, final calculation of the averages of interest.

4. Variation over the range of α selected and the repetition of steps described above to find the value corresponding to the minimum energy or application of the minimization algorithm.

In the present work these steps were performed for two different ways of sampling - the brute force sampling and the importance sampling,as well as an additional Gibbs sampling, which are described in subsequent sections.

3.2. *Metropolis algorithm*

The Metropolis algorithm is the central driving algorithm in the Monte Carlo methods including the variational Monte Carlo exploited. It allows to sample a normalized probability distribution by a stochastic process (12). The base for the Metropolis algorithm is the Markovian process and the evolution of the probability distribution function in a random walk of the system from the state j to the state i is described by the transition matrix $W_{i\to j}$ and is not known initially. It can be decomposed into two multiplied probabilities:$A_{i\to j}$ - the probability of accepting a move from j towards i and $T_{i\to j}$ the probability of making the step from the state j to the state i:

$$W_{j\to i} = A_{j\to i} \cdot T_{j\to i}, \tag{33}$$

and for the the probability of the system to occur in the state i after the n steps denoted as $P_i^{(n)}$ can be obtained in two following ways: on the n-th step the system undergoes the accepted step from each of j states which the system can pass through or it attempts to undergo the transition from the state i to all the j states but the corresponding step is rejected. This can be mathematically written as:

$$P_i^{(n)} = \sum_j (P_j^{(n-1)} T_{j\to i} A_{j\to i} + P_i^{(n-1)} T_{i\to j}(1 - A_{i\to j})), \tag{34}$$

as the probability of making a transition is $\sum_j T_{j\to i} = 1$, thus:

$$P_i^{(n)} = P_i^{(n-1)} + \sum_j (P_j^{(n-1)} T_{j\to i} A_{j\to i} - P_i^{(n-1)} T_{i\to j} A_{i\to j}). \tag{35}$$

The system approaches the equilibrium position as the time goes to infinity, at the same time $P_i^{(n)}$ approaches the p_i – the value of desired probability distribution,that implies:

$$\lim_{t\to\infty} P_i^{(n)} = p_i = \lim_{t\to\infty} P_i^{(n-1)} + \sum_j (\lim_{t\to\infty} P_j^{(n-1)} T_{j\to i} A_{j\to i} - \\ - \lim_{t\to\infty} P_i^{(n-1)} T_{i\to j} A_{i\to j}) = p_i + \sum_j (p_j T_{j\to i} A_{j\to i} - \\ - p_i T_{i\to j} A_{i\to j}), \tag{36}$$

and thus we get the following:

$$\sum_j (p_j T_{j\to i} A_{j\to i} - p_i T_{i\to j} A_{i\to j}) = 0, \tag{37}$$

which can be rewritten in the following way:

$$\frac{A_{j\to i}}{A_{i\to j}} = \frac{p_i T_{i\to j}}{p_j T_{j\to i}}. \tag{38}$$

This relation defines the condition governing the movement towards the most probable state and in the present work is is treated in two different approaches which involve the two different ways of sampling – the brute force and the importance sampling.

3.2.1. *The brute force Metropolis algorithm*

The base of the brute force Metropolis algorithm is an assumption that the transition probability is symmetric i.e. $T_{i\to j} = T_{j\to i}$.The acceptance probabilities are still not known and it defines the ratio of different probabilities. However, the main idea of the algorithm considered is to move to the equilibrium state in the most efficient way, or, it terms of proposed steps, to have the maximum possible number of steps accepted. In the scope of present work the larger acceptance rate will correspond to the smaller value of energy of the state we are moving to, since the probability of the latter is higher.The ratio (38) will be larger than 1 in case we are moving towards the more probable state and it is smaller than 1 in the opposite case. this can be reflected by the following condition for the brute force approach (the normalization factors are eliminated $P(\mathbf{R}) = |\Psi_T(\mathbf{R})|^2 / \int |\Psi_T(\mathbf{R})|^2 d\mathbf{R}$):

$$w = \frac{p_i}{p_j} = \frac{|\Psi_T(\mathbf{R}_i)|^2}{|\Psi_T(\mathbf{R}_j)|^2} > 1, \tag{39}$$

in this case the new step from the j state to the i state (for the further equations rthe corresponding positions are renamed as $\mathbf{R}$ and $\mathbf{R}'$) is accepted. However, it will be accepted in case:

$$w = \frac{p_i}{p_j} = \frac{|\Psi_T(\mathbf{R}')|^2}{|\Psi_T(\mathbf{R})|^2} \geq s, \tag{40}$$

where s is the random number and s$\in [0,1]$. Thus, the brute force approach will be the following in the case of the given problem:in each Monte Carlo cycle the new position is proposed:

$$\mathbf{R}' = \mathbf{R} + s \cdot \Delta R, \tag{41}$$

here s$\in [0,1]$ and ΔR is user defined step size for each particle in each direction, and then to calculate the wave functions for both new and old positions. Then the condition (40) is checked, if it is fulfilled, the step is accepted for a subsequent use, if no, the position remains the same. In both cases the energy and the variance are updated and then the calculation goes to the next Monte Carlo cycle. The drawback of this algorithm is the ΔR step size parameter which should be properly tuned to obtain the proper result for a given number of Monte Carlo cycles. The step proposal is not defined by the wave function of the system, and since it is manually adjusted, the brute force approach is not the most efficient way of point sampling.As a usual case the ΔR is chosen thus $\approx 50\%$ of Monte Carlo cycles (or step proposals) are accepted.

3.2.2. *The importance sampling*

The drawback of the previous algorithm can be eliminated, i.e. instead of choosing the the step size parameter and finding an appropriate value, the step can be defined by the trial wave function of the system. This algorithm is based on the Fokker-Planck equation and the Langevin equation. The latter equation is used in Monte Carlo calculations to define the new step proposed which will be dependent on the transition probability for a given point in space which will correspond to the initial state of the system. The new trial position is given as the solution of the Langevin equation (in one dimension) (1):

$$\frac{\partial x(t)}{\partial t} = DF(x(t)) + \eta, \tag{42}$$

where D has a meaning of factor in kinetic energy and equals to $1/2$, $F(x(t))$ is the drift force, η is the random variable. In general, the physical meaning of the following: a moving particle will constantly collide with other particles and these collisions will more likely occur on the front side preventing the particle from moving forward. This will result in a systematic force, proportional to velocity of a particle and having an opposite direction. In addition, the stochastic force is also present. Combining these factors, the evolution of the particle movement can be derived. The solution of the Langevin equation can be obtained by Euler's method and has the following form in one dimension:

$$x' = x + DF(x)\Delta t + \xi\sqrt{\Delta t}, \tag{43}$$

here ξ is gaussian random variable and the only parameter to be chosen is the time step Δt . The values of $\Delta t \in [0.001, 0.01]$ results in the relatively stable value of energy for the ground state.

On the other hand, the constant diffusion process a particle undergoes in all directions isotropically can be associated with the time-dependent probability density $P(\mathbf{R}, t)$. Taking into account that the initial condition of the system changes thus the system evolves to achieve the equilibrium state, it can be shown that the Fokker-Planck equation for $P(\mathbf{R}, t)$ takes place in many-dimensional case:

$$\frac{\partial P(\mathbf{R}, t)}{\partial t} = \sum_i D\frac{\partial}{\partial x_i}\left(\frac{\partial}{\partial x_i} - F_i\right)P(\mathbf{R}, t), \tag{44}$$

where x_i and F_i are the i-th components of the $\mathbf{R}$ position vector of the system and the drift force (or quantum force),which can be obtained from the Fokker-Planck equation in the form $\mathbf{F} = f()\frac{\partial P}{\partial \mathbf{x}}$ with the assumption of stationary densities :

$$\mathbf{F} = 2\frac{1}{\Psi_T}\nabla\Psi_T. \tag{45}$$

The importance sampling algorithm implies that the transition probabilities $T_{j\to i}$ (38) can no longer be omitted. And as the Langevin equation allows to obtain the new trial position, the Fokker-Planck equation provides with the transition probabilities in the form of the Green function (written in one dimension):

$$G(x', x, t) = \frac{1}{(4\pi D\Delta t)^{3N/2}}\exp(-(x'-x-D\Delta t F(x))^2/4D\Delta t). \tag{46}$$

In this case the Metropolis algorithm is called the Metropolis-Hastings algorithm and assumes the condition for the selection of steps in form of:

$$w = \frac{G(\mathbf{R}, \mathbf{R}', t)|\Psi_T(\mathbf{R'})|^2}{G(\mathbf{R}', \mathbf{R}, t)|\Psi_T(\mathbf{R})|^2} \geq s, \tag{47}$$

with $s \in [0, 1]$, and if this requirement is satisfied, the accepted move is accepted and used in subsequent calculations, otherwise the position remains the same.

3.2.3. *The Gibbs Sampling*

For this particular method the wave function is represented as $\Psi(x) = \sqrt{(F_{rbm})}$ not as before, being $\Psi(x) = F_{rbm}$ (11). It is the probability of $P_{rbm}(x, h)$ that is sampled in a two step process and then the probability density $|\Psi(\mathbf{x})|^2$ is modelled as wished. These probabilities are called conditional probabilities because they accept with the probability of 1, so the probability is either 1 or 0, accepted or not accepted. Since the Gaussian RBM is used with the Gibbs sampling the two conditional probabilities are:

$$P\left(x_i|\mathbf{h}\right) = \mathcal{N}(x_i; a_i + \sum_j h_j w_{ij}, \sigma^2) \tag{48}$$

for the visible unit and

$$P\left(h_j = 1|\mathbf{x}\right) = \frac{1}{1 + e^{-b_j - \frac{1}{\sigma^2}\sum_i x_i w_{ij}}} \tag{49}$$

for the hidden nodes. The visible layer follows a normal distribution while the hidden follow a logistic sigmoid function. The visible and the hidden nodes are dependent on one another as shown in the equation above: if $h = 1$ the probability is equal to the sigmoid function if $h = 0$ the step is not accepted. However the nodes are independent of other nodes within the same layer. These probabilities can be referred to as activation functions in the neural network since they either accept or not.

3.3. *Energy minimization and gradient descent method*

The local energy computed in each Monte Carlo iteration is the function of several parameters - biases a_i and b_i, weights W_i and σ_i from now on just referred to as α_i. These parameters are a part of the probability distribution the RBM is supposed to learn during Monte Carlo iterations and they are therefore related to the guess for the form of the trial wave function. a_i is the visible biases and b_i are the hidden biases and W_i is a matrix containing the weights characterizing the connection of each visible node to a hidden node. In principle, one can variate these parameters over a chosen region of values, i.e. to calculate local energy $E_L(\alpha_i)$ for each variational parameter and find the minimum $E_L(\alpha_i)$ graphically. It is done by optimizing the parameters mentioned above with the gradient descent method (or steepest descent method). This plain method is,nevertheless, quite time consuming when it is not initially clear where the minimum can be found. Its form applied to our particular problem is the following:

$$\hat{\alpha_i} = \alpha_i - \eta \cdot \frac{d\langle E(\alpha_i)\rangle}{d\alpha_i}, \tag{50}$$

where η is the learning rate in machine learning. The underlying idea is that $E_L(\alpha_i)$ decreases faster if one moves in variational parameter space towards the direction of negative gradient $\frac{d<E(\alpha_i)>}{d\alpha_i}$ and ideally the sequence of α_i converges to a global minimum of a given convex function. Despite relatively uncomplicated implementation this method is quite sensitive to a choice of initial guess for a parameter,learning rate and, moreover, it can be computationally expensive for large number of data involved. It also treats all directions in variational parameter space uniformly,faces struggles dealing with saddle points and defines only local minima in more complicated functions. This might become sensible problems when the machine learning is involved. However, in the present work the implementation excludes all the mentioned problems and there is no significant problem to choose the initial guess properly.

Implementation of the gradient descent in present work implies calculation of the following derivatives:

$$\bar{E_L}(\alpha_i) = \frac{d\langle E(\alpha_i)\rangle}{d\alpha_i}, \tag{51}$$

and as the local energy is presented as:

$$\langle E(\alpha_i)\rangle = \frac{\langle \Psi_T(\alpha_i)|\, \hat{H}\, |\Psi_T(\alpha_i)\rangle}{\langle \Psi_T(\alpha_i)|\Psi_T(\alpha_i)\rangle}, \tag{52}$$

differentiating one can obtain:

$$\bar{E_L}(\alpha_i) = 2\left[\langle E_L(\alpha_i)\frac{\bar{\Psi}_T(\alpha_i)}{\Psi_T(\alpha_i)}\rangle - \langle E_L(\alpha_i)\rangle\langle\frac{\bar{\Psi}_T(\alpha_i)}{\Psi_T(\alpha_i)}\rangle\right], \tag{53}$$

here $\bar{\Psi}_T(\alpha_i) = \frac{d\Psi_T(\alpha_i)}{d\alpha_i}$.

3.4. *Blocking method for variance estimation*

As it was already mentioned in the subsection **??** an appropriate method for obtaining the variance σ^2 in Monte Carlo experiments is required. For this aim the resampling methods are the inalienable tools which allow to efficiently obtain the variance by repeatedly drawing different samples from the given data set. Such popular methods as independent bootstrap and the jackknife (special case of independent bootstrap) can be used for independent, identically distributed random variables. However, for the large data sets they are not applicable because of increasing complexity, and the blocking method, which becomes even more accurate for increasing number of data, can be applied instead.

Let's consider the data set (stationary time series) $\vec{x} = \{x_1 + x_2 + ... + x_n\}$, where $n = 2^d$ and $d > 1$ is some integer. The blocking transformations involve this initial vector to create the new vector $\vec{x}_1$ by taking the mean of subsequent pair of elements from $\vec{x}$. Taking the $\vec{x}_1$ vector as the base and repeating procedure for other base vectors we get in each iteration one can end up after k blocking transformations with d new vectors with elements:

$$\begin{aligned}(\vec{x}_0)_k &= (\vec{x})_k \\ (\vec{x}_{i+1})_k &= \frac{(\vec{x}_i)_{2k-1} + (\vec{x}_i)_{2k}}{2}\end{aligned} \tag{54}$$

for $1 \leq i \leq d-1$ and it can be shown that if the components of $\vec{x}$ are stationary time series, then the components of $\vec{x}_i$ with $1 \leq i \leq d-1$ are the stationary time series as well.And if the initially chosen time series is asymptotically uncorrelated the kth series is also asymptotically uncorrelated. It can be shown that the variance of $\vec{x}_k$ is given as:

$$V(\bar{x}_k) = \frac{\sigma_k^2}{n_k} + \frac{2}{n_k}\sum_{h=1}^{n_k-1}\left(1 - \frac{h}{n_k}\right)\gamma_k(h) = \frac{\sigma_k^2}{n_k} + e_k, \tag{55}$$

where e_k is called truncation error and $\gamma_{k+1}(h) = cov((x_{k+1})_i, x_{k+1})_j)$ and $h = |i-j|$,$\gamma_k(0) = \sigma_k^2$. It can be also proven that $V(\bar{x}_k) = V(\bar{x})$ for $0 \leq k \leq d-1$ and thus the variance of the sample mean we are looking for all $0 \leq k \leq d-1$ is:

$$V(\bar{x}) = \frac{\sigma_k^2}{n_k} + e_k. \tag{56}$$

As the number of blocking procedures increases (i.e. more measurements is inserted in each subsequent block) the e_k can be made small enough to make $V(\bar{x}) = \frac{\sigma_k^2}{n_k}$ a good estimate for the variance.

4. CODE AND IMPLEMENTATION

The knowledge on classes and their implementation obtained in processing the Project 1 were sufficient to implement classes in the present project as well. The code aims at the simulation of P particles (electrons) system and its characteristics in D dimensional harmonic oscillator with and without interactions. The *main.cpp* file contains initialization of all parameters defying the system which are then sent to the central function contained in the *gradientdescent.cpp* file. Parameters to be chosen are the number of electrons, dimensions,

which give a the number of nodes within the input layer, number of MC cycles, and number of iterations. On the other hand, user may also vary the number of hidden nodes and trace, how it affects a time of calculations. The most interesting part of the project involves changes in the learning rate, time step for importance sampling algorithm, and step length for the brute force method, which can be also set in *main.cpp*. Most of the calculations are performed with $\omega = 1$ and $\sigma = 1$, and they are also set before being sent to the *gradientdescent.cpp* file. Finally, the type of sampling should be chosen before implementing the gradient descent. The GD function of

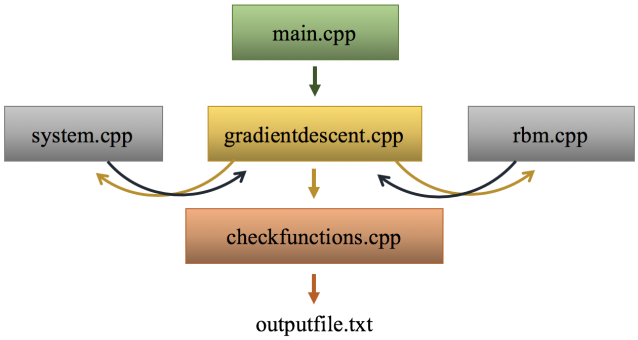

FIG. 3.— The scheme of the code with classes and all the connections between different classes.

the *gradientdescent.cpp* file builds up the basic mechanism for the cost function (local energy) minimization. This optimization is performed by the main loop over the number of iterations the gradient descent requires to reach an approximate minimum value of the local energy. The main loop over the chosen number of iterations contains the loop over the VMC cycles and an option of writing an output of VMC cycles into a file. The two supplementary classes are used to perform the calculations: the *rbm* class (*rbm.cpp*) and the *system* class (*system.cpp*). The former contains the M dimensional vector $\mathbf{x}$ filled with positions and N dimensional $\mathbf{h}$ vector of hidden nodes as well as corresponding biases and weights. The principal purpose of this class is to set random positions, hidden nodes, biases and weights (*RBM::setup_rweights()*, *RBM::setup_rposition()*,and *RBM::setup_hidden_layer()*).On the other hand, it also contains sampling functions *RBM::pick_x(int i_M)* and *RBM::pick_h(int i_N)* required, when the Gibbs sampling is chosen in main.The *system* class class contains the cost function in the form of local energy, all the gradients required in the optimization, and supplementary functions, such as logistic sigmoid and the Green function called when the Metropolis sampling is switched on. Switching between different sampling methods are performed within the gradient descent loop and the MC loop.

In contrast to the program performed in the project 1 ((5)),the standard template library for linear algebra *Eigen* was chosen instead of the *matrix* class. The latter produced several issues being slightly more bulky, than *Eigen*. Vector form of all the objects and calculations performed facilitates reading of the code lines, making it much more understandable for a user. Additional check functions are added in order to provide user with a benchmark of a result obtained and check the divergence of the local energy from an analytical result for each specific case. If a difference of local energy and analytical energy calculated exceeds a certain tolerance, user will see a notification in the terminal.

The main variables used are performed the problem are presented in the block below:

```
//Parameters used for the program
    int N = 1; // Number of hidden nodes
    (2,3,..20 in the programm)
    int P =  1; // Number of particles (either 1
     or 2)
    int D = 1; //Number of dimensions (either 1
    or 2)
    double sigma = 1; // standard deviation of
    the distribution
    double omega = 1; // Frequency of HO
    int NumofMC = 100000; //  Number of MC
    cycles
    int numVar = 500; // Number of iterations in
     GD
    int samp=1; // (0 for brute force, 1 for
    Matropolis-Hastings, 2 for Gibbs)
    bool interact = false; //interaction
    switcher
    double stepsize = 0.1; // step size for
    brute force
    double timestep = 0.1; // time step for
    Metropolis-Hastings
    double learnRate = 0.2; // Learning rate
    double diffConst = 0.5; // Diffusion
    constant
```

The following block demonstrates principal structure of the main.cpp file, where all parameters are set.

```
#include "system.h"
#include "gradientdescent.h"
#include <iostream>
#include <Eigen/Dense>
#include "mpi.h"

using namespace Eigen;

using namespace std;

int main()
{ MPI_Init(nullptr, nullptr);

Setting of parameters

GradientDescent(Parameters);

  MPI_Finalize();
}
```

The block below contains principal scheme of the supplementary class in rbm.cpp file.

```
RBM::RBM(Param){
setup_RBM(Param);
}// Outer constructor

void RBM::setup_RBM(Param){
    setup_rweights();
    setup_rposition();
    setup_hidden_layer();
}// Main constructor
void RBM::setup_rweights(){}// Setting up random
     a,b,W
void RBM::setup_rposition(){}// Setting up
    random x
```

```
void RBM::setup_hidden_layer(){}//Setting up
    random hidden layer
double RBM::pick_x(int i_M){}//x sampling for
    Gibbs
double RBM::pick_h(int i_N){}//h sampling for
    Gibbs
```

Principal structure of another supplementary class in system.cpp file containing all the functions required for the GD and VMC.

```
System::System(Param){}//Constructor
double System::localEnergy(Param){}//Local
    energy calculation
VectorXd System::grad_a(){}//Gradients in the
    vector form
double System::QuantumForce(){}//Quantum force
double System::GreenFunction(){}//Green function
```

All the calculations, including VMC loop and the gradient descent loop are presented in gradientdescent.cpp file with the following structure:

```
void GradientDescent(Param){
RBM R(Param);//Setting of RBM
System Hamiltonian(Param);//Setting of the
    system
for (int var = 0; var < numOfvar; var++){
  Initial positions and variables setting;
        for (int cycl = 0; cycl < MCcycles /
    n_procs; cycl ++){
        Sampling cases, switching;
        Energy is updated and accumulated;
    }
    Averaging;
    GD procedure;
    Writing to file;
  }
}
```

In addition, the parallelization of the code was carried out by the means of MPI message-passing library. Two MPI processes were distinguished, that was followed by the reduction of CPU time by an approximate factor of 2.

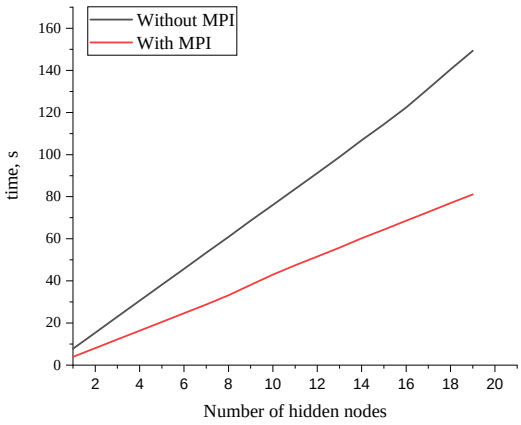

FIG. 4.— Comparison of CPU times for the runs with and without parallelization for different numbers of hidden nodes. Run for 1 iteration.

The figure 4 demonstrates main run courses with and without parallelization for different number of hidden nodes. In average each node added reduces speed of calculation, so that the time required for running one iteration is proportional to the number of nodes used. Hence, an improved speed comprises of both parallezation of a process as well as an appropriate number of hidden nodes used. For the case of a quantum dot the results revealed improvement with the lower number of hidden nodes accompanied with increased speed.

5. RESULTS

The following section is devoted to results calculated by the means of machine learning approach implemented in the program described above. First of all, the simpliest case of one electron in one dimensional system is considered. Three different types of sampling, the brute force method, the Metropolis-Hastings, and the Gibbs sampling, were tested for a given system. For each case the different learning rates were chosen to obtain the results. The best learning rate (among chosen) was then fixed while the number of hidden nodes becomes a new variable. In addition, for the case of Gibbs sampling, σ was also varied to reflect the learning progress dependence on it. Further complication of the system implies extention of the system to 2 electrons in 2 dimensions. All three types of sampling were compared and the brute force method was chosen to demonstrate, how the learning process is affected by the learning rate and number of hidden nodes chosen. All the results calculated demonstrate a strong dependence on initial guesses made, what is exactly expected from the gradient descent approach.

5.1. *The Brute Force Approach for Local Energy Estimation*

The first type of calculations performed exploits the brute force VMC for the local energy estimation. The series of these estimates calculated in the gradient descent loop are performed so, that the weights W and biases $\mathbf{a}$ and $\mathbf{b}$ are adjusted in each iteration in order to achieve the ground state, i.e. optimize the system. The standard deviation is held constant and equal to 1. One of the most representative ways to visualize, how the learning process is performed and might be performed in the most effective way is to vary the learning rate η. This variable directly affects the speed of learning by governing the speed of gradient updating after passing all MC cycles. The figure 5 demonstrates five different ways of energy evolution or learning for the chosen learning rates $\eta = 0.3, 0.2, 0.1, 0.05, 0.01$. The number of hidden nodes is 2 and the step size for movement proposals was chosen to be $step = 1$.

One observes quite good convergence for learning rates $\eta = 0.3, 0.2, 0.1, 0.05$, all the processes manage to reach a plateau in the vicinity of analytical solution $E_L = 2$ a.u. The larger learning rate is, the faster calculations converge to the analytical solution. For learning rate $\eta = 0.01$ the learning process turned out to be noticeably slower and larger number of iterations is required

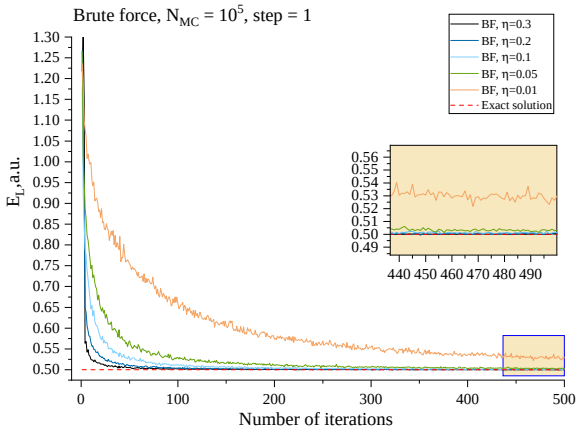

FIG. 5.— Dependence of local energy on number of iterations for brute force VMC. The step size chosen is $step = 1$,number of MC cycles is 10^5, number of hidden nodes $N = 2$, $\sigma = 1$, $\omega = 1$. Various learning rates are used $\eta = 0.3, 0.2, 0.1, 0.05, 0.01$.

if one aims to achieve convergence similar to that for $\eta = 0.3, 0.2, 0.1, 0.05$.

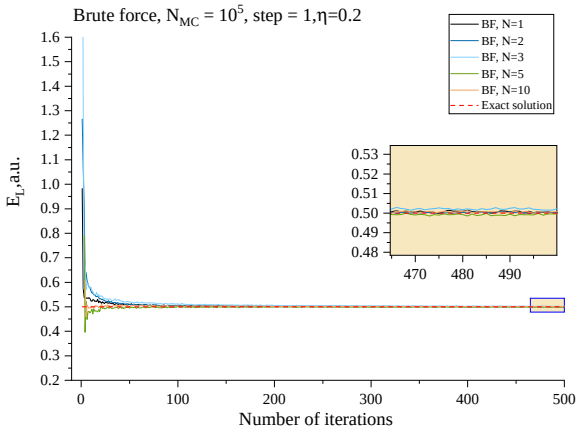

FIG. 6.— Dependence of local energy on number of iterations for brute force VMC. The step size chosen is $step = 1$,number of MC cycles is 10^5, learning rate $\eta = 0.2$, $\sigma = 1$, $\omega = 1$. Various hidden nodes are used $N = 1, 2, 3, 5, 10$.

The next step to be considered is variation of number of hidden nodes N. Local energies obtained in this case are presented in figure 6. For lower numbers of nodes $N = 1, 2, 3$ the initial energy increases, and for the larger number of nodes it takes noticeably larger number of iterations to reach a plateau. For $N = 5$ and 10 significant fluctuations are observed for lower number of iterations.

5.2. *The importance sampling approach for local energy estimation*

Similar calculations were performed for Metropolis-Hastings based VMC. First of all, different learning rates $\eta = 0.3, 0.2, 0.1, 0.05, 0.01$ are investigated. Here, the number of MC cycles is 10^5, size of the time step is $\Delta t = 0.01$, number of hidden nodes $N = 2$.

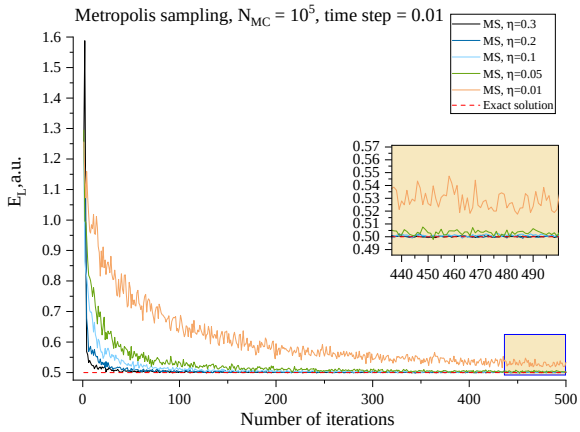

FIG. 7.— Dependence of local energy on number of iterations for importance sampling VMC. The time step size chosen is $s\Delta t = 0.01$,number of MC cycles is 10^5, number of hidden nodes $N = 2$, $\sigma = 1$, $\omega = 1$. Various learning rates are used $\eta = 0.3, 0.2, 0.1, 0.05, 0.01$.

As the figure 7 shows,for all the learning rates,except for

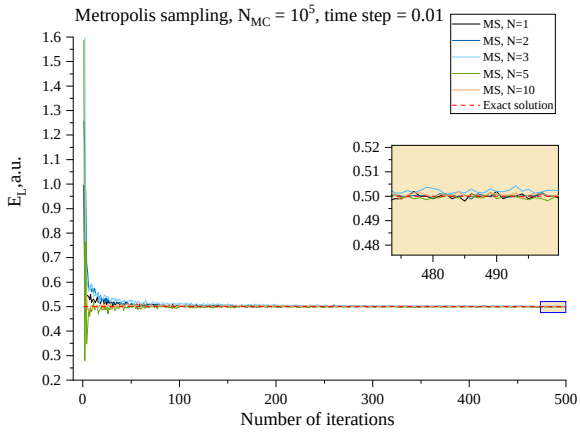

FIG. 8.— Dependence of local energy on number of iterations for importance sampling VMC. The time step size chosen is $\Delta t = 0.01$,number of MC cycles is 10^5, learning rate $\eta = 0.2$, $\sigma = 1$, $\omega = 1$. Various hidden nodes are used $N = 1, 2, 3, 5, 10$.

$\eta = 0.01$, the neural network is able to learn the correct values for the energy in 500 iterations, however for $\eta = 0.05, 0.01$ it takes longer time than the bigger values. Similarly to the case of the brute force, importance sampling experiments with different values for the hidden nodes were carried out as well. Here the time step was chosen to be constant $\Delta t = 0.01$ and the number of Monte Carlo cycles was set to be 10^5. When the number of hidden nodes was varied to be over $N = 5$, the results started fluctuating more, however, they still stabilize around the exact energy after considerably short amount of iterations. In the table 1 one can examine the results for the local energy for the different sampling methods and compare how accurate the different methods are with respect to different numbers of iterations.

5.3. *The Gibbs sampling approach for local energy estimation*

Again, similar calculations where performed for the Gibbs sampling. The same variations of the learning rates where used $\eta = 0.3, 0.2, 0.1, 0.05, 0.01$. With the number of Monte Carlo cycles being 10^5 and $\sigma = 1$.

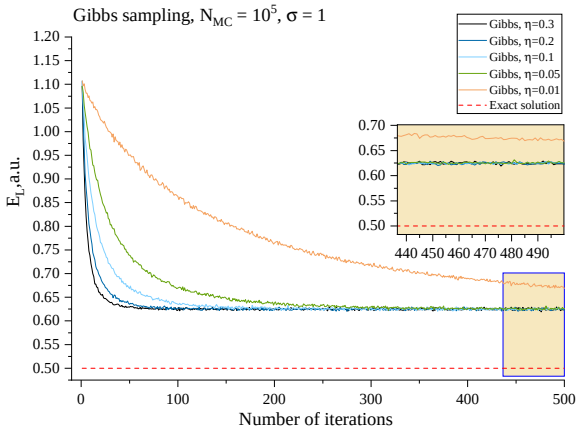

FIG. 9.— Dependence of local energy on number of iterations for the Gibbs VMC. The number of MC cycles is 10^5, learning rate $\eta = 0.2$, $\sigma = 1$, $\omega = 1$. Various learning rates are used $\eta = 0.3, 0.2, 0.1, 0.05, 0.01$.

Figure 9 shows the results for these calculations. As can be seen from this figure the sampling goes down to a value between 0.6 and 0.65 which is high above the exact value.

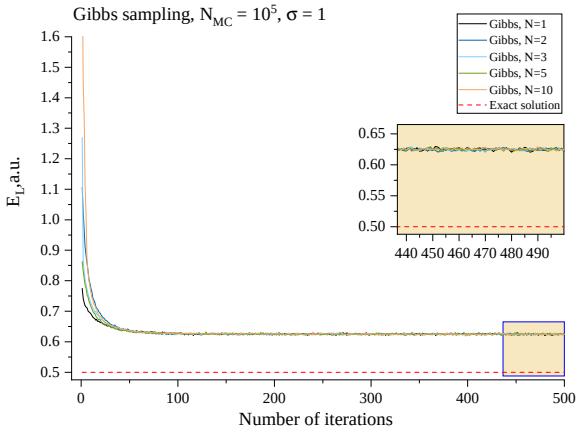

FIG. 10.— Dependence of local energy on number of iterations for the Gibbs VMC. The number of MC cycles is 10^5, learning rate $\eta = 0.2$, $\sigma = 1$, $\omega = 1$. Various hidden nodes are used $N = 1, 2, 3, 5, 10$.

For the variation of hidden nodes the same amount of Monte Carlo cycles and the $\sigma = 1$ were applied. The result can be seen in figure 10 and, just as in the case of variation of the learning rate, the results are high above the exact value of the local energy. For this type of sampling an additional test with the σ value was carried out. Figure 11 shows the variation of $\sigma = 0.50, 0.75, 1, 1.25, 1.5$ with learning rate $\eta = 0.2$ and 10^5 Monte Carlo cycles, two hidden nodes are considered. The figure shows especially good results for $\sigma = 0.75$, as the local energy calculated approaches the analytical solution.

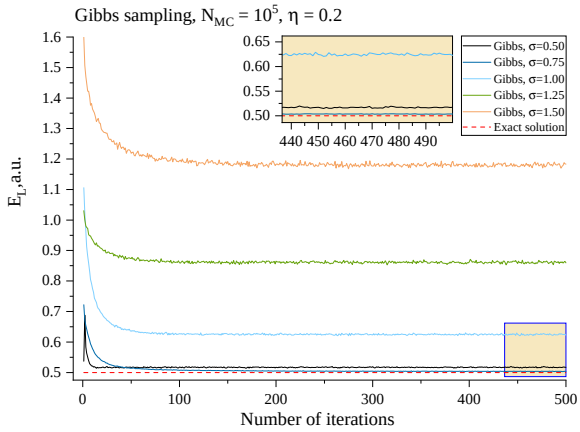

FIG. 11.— Dependence of local energy on number of iterations for the Gibbs VMC. The number of MC cycles is 10^5, learning rate $\eta = 0.2$, $N = 2$, $\omega = 1$. Various σ are used $\sigma = 0.5, 0.75, 1.0, 1.25, 1.5$.

5.4. *Joint analysis of the Brute Force, importance sampling, and the Gibbs sampling approach for local energy without interaction included*

The figure 12 demonstrates comparison of different types of sampling for the same parameters chosen: number of MC cycles is 10^5, number of hidden nodes is $N = 2$, step size $\Delta r = 1$, time step $\Delta t = 0.01$, and $\sigma = 1$.

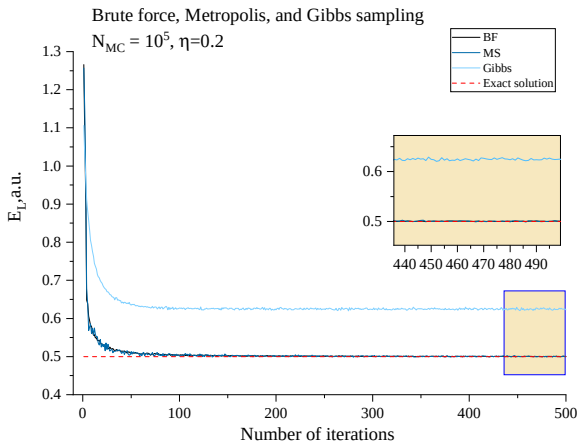

FIG. 12.— Comparison of local energy for the brute force, the Metropolis-Hastings and Gibbs VMC for non- interacting case. The number of MC cycles is 10^5, learning rate $\eta = 0.2$, $N = 2$, $\omega = 1$, $\sigma = 1$.

Intermediate local energy estimations for numbers of iterations 100,200,300,400,and 500 are shown in the table 1. One can notice similar behaviour of the brute force and importance sampling approach throughout all iter-

ations, so the presented by almost coinciding curves on the figure 12. The corresponding behaviour of the Gibbs sampling based VMC is different, since the calculations reach another plateau around $E_L \approx 0.62$, and the variance of calculations is still significant. However it can be corrected by including σ optimization, which is added to the program, but not used, since the default $\sigma = 1$ is required. An example if an appropriate local energy calculation is shown on the figure 10 for $\sigma = 0.75$.

TABLE 1
THE RESULTS OF THE CALCULATIONS FOR LOCAL ENERGY OBTAINED FOR DIFFERENT NUMBER OF ITERATIONS WITHOUT INTERACTION FOR THE BRUTE FORCE (BF) APPROACH, THE IMPORTANCE SAMPLING (IS), AND THE GIBBS SAMPLING. NUMBER OF MC CYCLES 10^5,STEP SIZE $\Delta r = 1$, TIME STEP $\Delta t = 0.01, sigma = 1$, $\omega = 1$.

$N_{iterations}$	$E_L^{BF}, a.u.$	$E_L^{IS}, a.u.$	$E_L^{Gibbs}, a.u.$	$E_L^{a}, a.u.$
100	0.503(61)	0.501(63)	0.63(54)	0.500
200	0.501(36)	0.501(37)	0.62(52)	0.500
300	0.500(25)	0.502(27).	0.62(53)	0.500
400	0.500(20)	0.500(21)	0.63(53)	0.500
500	0.500(17)	0.500(17)	0.62(52)	0.500

5.5. *Joint analysis of the brute force, importance sampling, and the Gibbs sampling approach for local energy with interaction included*

The calculations for interaction were performed for all the sampling methods and can be examined in figure 13. The number of particles are two $P = 2$ and so is the dimensions $D = 2$. The learning rate used for all the method was $\eta = 0.2$ and again the Monte Carlo cycles where set to be 10^5.

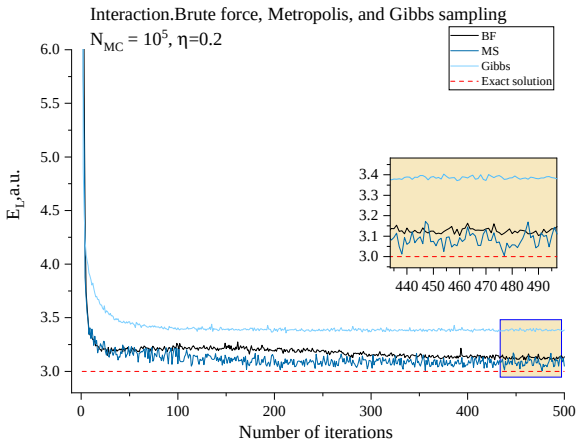

FIG. 13.— Comparison of local energy for the brute force, the Metropolis-Hastings and Gibbs VMC for interaction case. The number of MC cycles is 10^5, learning rate $\eta = 0.2$, $N = 2$, $\omega = 1$, $\sigma = 1$.

The table 2 shows the local energy for all the sampling methods with errors compared to number of iterations and the exact value. It might be noticed,that in all the calculations obtained up til this part, the factor, found in our code 9, which decides where the starting point for the program,was set to be is 2. In these calculations it was necessary to start higher since the interaction between two particles in two dimensions gives a higher exact value for the local energy, the factor for interaction was thus set to be 3.

TABLE 2
THE RESULTS OF THE CALCULATIONS FOR LOCAL ENERGY OBTAINED FOR DIFFERENT NUMBER OF ITERATIONS WITH INTERACTION FOR THE BRUTE FORCE APPROACH, IMPORTANCE SAMPLING, AND GIBBS SAMPLING. NUMBER OF MC CYCLES 10^5,STEP SIZE,TIME STEP $\Delta r = 1$, $\Delta t = 0.01$, $r_{step} = 1.0, sigma = 1$, $\omega = 1$.

$N_{iterations}$	$E_L^{BF}, a.u.$	$E_L^{IS}, a.u.$	$E_L^{Gibbs}, a.u.$	$E_L^{a}, a.u.$
100	3.2(17)	3.1(13)	3.4(17)	3.0
200	3.2(19)	3.1(12)	3.4(16)	3.0
300	3.1(16)	3.1(10).	3.4(14)	3.0
400	3.1(14)	3.1(12)	3.3(15)	3.0
500	3.1(13)	3.0(10)	3.3(16)	3.0

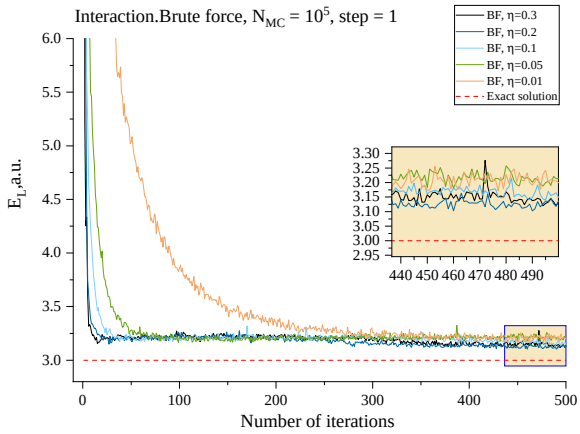

FIG. 14.— Dependence of local energy on number of iterations for brute force VMC for interacting case. The step size chosen is $step = 1$,number of MC cycles is 10^5, number of hidden nodes $N = 2$, $\sigma = 1$, $\omega = 1$. Various learning rates are used $\eta = 0.3, 0.2, 0.1, 0.05, 0.01$.

Additionally, from this point on the only calculations done are with the brute force sampling method with two particles $P = 2$ and two dimensions $D = 2$ with the step size $\delta t = 1$ and 10^5 Monte Carlo cycles to demonstrate all the tendencies. The first figure 14 shows the brute force method with interaction whilst varying the learning rate $\eta = 0.3, 0.2, 0.1, 0.05, 0.01$. The fluctuations are significantly greater then for one particle in one dimension, and the value that is fluctuated around is greater than the exact value 3 a.u.. Figure 15 demonstrates the brute force method with interaction with the learning rate $\eta = 0.2$ and this time varying the number of hidden nodes $N = 1, 2, 3, 5$.

5.6. *The Blocking Method and improved statistical errors*

In the calculations with the different sampling methods the Blocking Method was used to acquire the improved statistical errors for the data. In the table 3 the variance

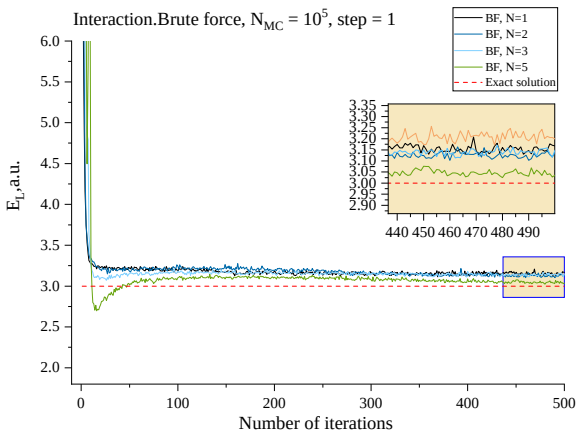

FIG. 15.— Dependence of local energy on number of iterations for brute force VMC for interacting case. The step size chosen is $step = 1$,number of MC cycles is 10^5, learning rate $\eta = 0.2$, $\sigma = 1$, $\omega = 1$. Various hidden nodes are used $N = 1, 2, 3, 5$.

σ^2 for the brute force, importance sampling, and Gibbs sampling can be found for the different learning rates $\eta = 0.3, 0.2, 0.01$. The same was also done with the variation of number of hidden nodes N and in the table 4 one can see the variance σ^2 for the three different methods for $N = 1, 2, 3$. In addition, for the fixed learning rate $\eta = 0.2$ and number of hidden nodes, similar analysis was carried out for the interacting case, results of improved variances are presented in the table 5.

TABLE 3
THE RESULTS OF THE CALCULATIONS FOR LOCAL ENERGY OBTAINED FOR 500 ITERATIONS WITHOUT INTERACTION FOR THE BRUTE FORCE APPROACH, METROPOLIS-HASTINGS, AND GIBBS SAMPLING. THE RESULTS ARE PERFORMED WITH BLOCKING BASED AND ORDINARY STATISTICAL VARIANCES $\sigma^2(E_L)$=< E_L^2 >-< E_L >^{2}. NUMBER OF MC CYCLES 2^18,STEP SIZE,TIME STEP $\Delta r = 1$, $\Delta t = 0.01$, $r_{step} = 1.0$,$sigma = 1$, $\omega = 1$,$N = 2$. DIFFERENT LEARNING RATES ARE CONSIDERED.

η	< E_L >,a.u.	$\sigma^2_{block}(E_L)$,(a.u)^2	$\sigma^2_0(E_L)$,(a.u.)^2
Brute force			
0.3	0.5004	$2.176{\cdot}10^{-4}$	$1.507{\cdot}10^{-4}$
0.2	0.5002	$3.245{\cdot}10^{-4}$	$2.908{\cdot}10^{-4}$
0.01	0.5245	$1.446{\cdot}10^{-3}$	$9.258{\cdot}10^{-4}$
Importance sampling			
0.3	0.5002	$2.524{\cdot}10^{-4}$	$1.855{\cdot}10^{-4}$
0.2	0.5002	$3.372{\cdot}10^{-4}$	$2.851{\cdot}10^{-4}$
0.01	0.5245	$3.324{\cdot}10^{-2}$	$3.347{\cdot}10^{-2}$
Gibbs sampling			
0.3	0.6242	$2.793{\cdot}10^{-1}$	$2.796{\cdot}10^{-1}$
0.2	0.6213	$2.762{\cdot}10^{-1}$	$2.767{\cdot}10^{-1}$
0.01	0.6720	$3.782{\cdot}10^{-1}$	$3.787{\cdot}10^{-1}$

5.7. *Variation of learning rate*

Throughout the whole project, for all the calculations performed, the learning rate was a fixed number. However, the learning process should not necessarily carried out with a certain learning rate, it can be adjusted with number of iterations N_{it}. One of such dependencies was

TABLE 4
THE RESULTS OF THE CALCULATIONS FOR LOCAL ENERGY OBTAINED FOR 500 ITERATIONS WITHOUT INTERACTION FOR THE BRUTE FORCE APPROACH, METROPOLIS-HASTINGS, AND GIBBS SAMPLING. THE RESULTS ARE PERFORMED WITH BLOCKING BASED AND ORDINARY STATISTICAL VARIANCES $\sigma^2(E_L)$=< E_L^2 >-< E_L >^{2}. NUMBER OF MC CYCLES 2^18,STEP SIZE,TIME STEP $\Delta r = 1$, $\Delta t = 0.01$, $r_{step} = 1.0$,$sigma = 1$, $\omega = 1$,$\eta = 0.2$. DIFFERENT NUMBERS OF HIDDEN NODES ARE CONSIDERED.

N	< E_L >,a.u.	$\sigma^2_{block}(E_L)$,(a.u)^2	$\sigma^2_0(E_L)$,(a.u.)^2
Brute force			
1	0.5006	$5.198{\cdot}10^{-4}$	$4.547{\cdot}10^{-4}$
2	0.5002	$3.372{\cdot}10^{-4}$	$2.851{\cdot}10^{-4}$
3	0.5006	$3.715{\cdot}10^{-4}$	$4.164{\cdot}10^{-4}$
Importance sampling			
1	0.4999	$6.605{\cdot}10^{-4}$	$7.539{\cdot}10^{-4}$
2	0.5002	$3.372{\cdot}10^{-4}$	$2.851{\cdot}10^{-4}$
3	0.5013	$6.046{\cdot}10^{-4}$	$7.386{\cdot}10^{-4}$
Gibbs sampling			
1	0.6264	$2.765{\cdot}10^{-1}$	$2.764{\cdot}10^{-1}$
2	0.6213	$2.762{\cdot}10^{-1}$	$2.767{\cdot}10^{-1}$
3	0.6307	$2.808{\cdot}10^{-1}$	$2.815{\cdot}10^{-1}$

TABLE 5
THE RESULTS OF THE CALCULATIONS FOR LOCAL ENERGY OBTAINED FOR 500 ITERATIONS WITHOUT INTERACTION FOR THE BRUTE FORCE APPROACH, METROPOLIS-HASTINGS, AND GIBBS SAMPLING. THE RESULTS ARE PERFORMED WITH BLOCKING BASED AND ORDINARY STATISTICAL VARIANCES $\sigma^2(E_L)$=< E_L^2 >-< E_L >^{2}. NUMBER OF MC CYCLES 2^18,STEP SIZE,TIME STEP $\Delta r = 1$, $\Delta t = 0.01$, $r_{step} = 1.0$,$sigma = 1$, $\omega = 1$, $\eta = 0.2$, $N = 2$

Sampling	< E_L >,a.u.	$\sigma^2_{block}(E_L)$,(a.u)^2	$\sigma^2_0(E_L)$,(a.u.)^2
GB	3.140	1.642	1.640
IS	3.396	1.086	1.084
GS	3.060	2.271	2.267

tested:

$$\eta = \frac{0.5}{N_{it}}, \tag{57}$$

meaning that the learning process speed is maximum for the lower number of iterations, and as the machine learns, it drops down. An assuption was, that the smaller learning rate closer to the large numbers of iterations might reduce fluctuations. However, the results presented on the figure 16, show that for the large constant learning rate the better result is achieved and accompanied with small fluctuations, as compared to the varied learning rate. Nevertheless, different dependencies can be chosen for future studies.

6. DISCUSSION

In the following section the results presented in the section above will be discussed. First, the estimation of the local energy done with the brute force approach for one electron in one dimension, for so to move on for the same case with the importance and the Gibbs sampling. Furthermore, the interaction with all the three sampling methods will be examined and the statistical error, found with the Blocking Method section, will also be discussed.

Beginning with the simplest calculation preformed in

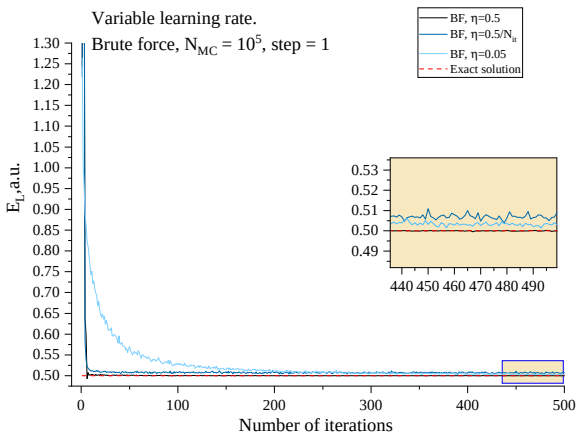

FIG. 16.— Dependence of local energy on number of iterations for brute force VMC for non-interacting case. The step size chosen is $step = 1$,number of MC cycles is 10^5, learning rate $\eta = 0.5, 0.05$, and $\eta = \frac{0.5}{N_{iter}}$ are chosen.

this study, the brute force for one particle in one dimension, the learning step was varied, and the corresponding results are shown in figure 5. By looking at the figure one can see, that the higher learning rate ($\eta = 0.3$) the faster the program reaches the exact value and fluctuates less around the exact value for the remaining iterations. This is to be expected, in the figure 16, there is shown the learning rate $\eta = 0.5$ together with a varying learning rate (from 0.5 to 0.001), and the learning rate $\eta = 0.05$. This is a good illustration of how well the program learns for higher learning rates. For the values lower $\eta = 0.2, 0.1, 0.05, 0.01$ the program fluctuates slightly more, especially for $\eta = 0.01$, and for this case the program never reaches the exact value for a chosen number of iterations. For all other values of the learning rate they reach the exact value within $200-300$ iterations and then fluctuate around the exact value for the rest of the run. In this study the learning rate $\eta = 0.2$ was chosen as the default value for the learning rate when not varying it, this because would correspond to the middle of to high and to low performance of the program, thus giving good values for the purpose of this study.

Mentioning again figure 16, this illustrates the learning rate being high and low and also varying the learning rate from 0.5 to 0.001 like mentioned before. This shows that in the beginning the varying learning rate follows the 0.5-rate quite well, but when the varying learning rate drops it also results in the program never reach the exact value, but it fluctuates right above it for the remaining iterations. It is thus concluded that having the learning rate high in the beginning for then learning slow in the end is not a good solution for the particular problem.

Figure 6 shows the brute force sampling with different hidden nodes with $\eta = 0.2$ and 10^5 Monte Carlo cycles. For $N = 1, 2, 3$ the results look similar, however, when performing the calculations for $N = 5, 10$, such large number of hidden nodes turned out to affect the fluctuation in the learning in the beginning of the process. Nevertheless, all the solutions end up fluctuating around the exact value between 0 and 100 iterations, so the chosen amount of iterations was sufficient. Since the values might fluctuate significantly for low number of iterations, comparatively low number of hidden nodes can be the answer for this study, however for problem of more complex systems this the reverse (larger numbers of nodes) might be more applicable.

Moving on to the importance sampling and the estimation of the local energy for this method, one find results of experiments with the learning rate parameter on fugure 7. This shows the varying of η, the learning rate of the program. The main difference from the Brute Force is the fluctuation which is much more abrupt in the case of Metropolis-Hastings algorithm, than for the brute force.This might be presumably explained by the introduction of quantum force and the Green function and the way the calculations are performed in this case. Nevertheless, both methods reach the exact values for the higher learning rates within 300 iterations, but again $\eta = 0.01$ proves to be too slow for the learning rate, and will thus use more than 500 iterations to achieve the exact value of the local energy.

Figure 8 shows the Metropolis-Hasting with different numbers of hidden nodes $N = 1, 2, 3, 5, 10$ for $\eta = 0.2$. This also similar results to that of the brute force. When using low numbers for the hidden nodes the results turn out to be quite good, and the program learns within approximately 100 iterations. However, as for the brute force, the nodes higher than $N = 5$ provide fluctuations reaching non-physical results in the beginning of learning process.

For the Gibbs sampling the sigma can be varied along with the learning rate and the number of hidden nodes. In the first figure 9 the sigma is kept constant $\sigma = 1$ and only the learning rate was varied as for the brute force and the importance sampling methods. Gibbs appears to be even more dependent on high learning rate compared to the other methods, it uses more time for $\eta = 0.1, 0.05$ to reach a plateau of a stable value. The more important fact:the Gibbs sampling based calculations never reach the exact value for any of the learning rates for the fixed $\sigma = 1$. This might be explained by specific choice of the wave function (see 3.2.3). With the chosen wave function $\Psi(x) = \sqrt{(F_{rbm})}$, the Gibbs sampling does not alow to reproduce the same results as for the brute force and importance sampling, given the same value of $\sigma = 1$, which requires further adjusting. Figure 10 shows the Gibbs method when varying the number of hidden nodes for $\eta = 0.2$, the learning process seems to be stable for all the different numbers of hidden nodes, and it fluctuates less around the value it settles around, the different values for the hidden node does not require significant number of iterations either. One thing to mention here is the factor which is chosen for the program as a starting point. In the case of Gibbs the factor was set higher than for the other methods, the program thus initially starts at a higher point than before. This might have made the fluctuation for higher number of hidden nodes more stable. However, the Gibbs sampling is not reaching the exact value rather it settles around 0.6 and not 0.5. Returning back to the σ dependence,

of the Gibbs sampling method, the results are shown in figure 11.Here the σ parameter is varied and this gives different outcome for what values the program ends up fluctuating around.With $\sigma = 0.75$ giving the best result as well as $\sigma = 0.5$. This indicates that Gibbs sampling is a method which can be trusted to find the correct value only if the optimization of σ is included.

In table 3,4, and 5 the improved statistical errors for this study is represented for three of the different learning rates $eta = 0.3, 0.2, 0.01$ and for three of the values of the hidden nodes $N = 1, 2, 3$ as well as for interacting case.

From examining the table one can see that the Blocking Method gives slightly higher values than the ordinary variance calculated in the program. The variance is thus greater than first indicated by examining only the ordinary variance. However, the ordinary variance demonstrates an appropriate magnitude for all the methods used. The greatest statistical errors is for the Gibbs method where the magnitude of the variance is 10^{-1} whilst the magnitude for the brute force and importance sampling is approximately 104. This indicates that the Gibbs sampling method may be to incorrect to use at least for the case in question compared to the other sampling methods, unless an appropriate σ is found. Even for one of σ examined, $\sigma = 0.75$, the variance is of the same magnitude, as for other methods.

The rest of the calculations for this study were performed with interaction included in the Hamiltonian and thus extending the problem to two particles in two dimension at least. Figure 13 shows the three methods with the learning rate $\eta = 0.2$ and 10^5 Monte Carlo cycles. None of the methods are able to reach the exact value 3 a.u., for 500 iterations, and they all fluctuate more than for the case without interaction as can be seen in figure 12. This illustrates that as the system gets more complex, the program need more iterations to learn the exact values and has comparatively low speed of stabilization, even though the same learning rates were chosen. Again, the Gibbs sampling settles at a different value than the exact, the reason being that the parameter $\sigma = 1$.The same occurs for the case without interaction in figure 12.

Figure 14 and 15 shows the Brute Force with interaction when varying the learning rate and the number of hidden nodes N, as was done for the case without interaction. It might be noticed that for the case with interaction included,the brute force never reaches the exact value for the local energy when varying the learning rate. All the calculations settle around 3.25 a.u. and they all fluctuate more than for the case without interaction. When varying the number of hidden nodes one can notice that the case for $N = 5$ is quite close to the exact value. This indicates that the larger number of hidden nodes is indeed better for a more complex system and may therefore fluctuate closer to the exact value than less number of hidden nodes will. However, $N = 5$ provides more fluctuations in the beginning of the learning process, and early on approaches the exact value. Then the program learns, reaching closer to the exact value, after about 400 iterations.

In the end one can examine 5 to see the statistical errors for all the sampling methods with interaction. Again the blocking method gives higher variance than the original variance made in the program, this is to be expected. The variance is again the greatest for the Gibbs sampling method, but for the interaction case the variance is quite high for all the methods examined. The reason is mainly hidden in the fact, that none of the methods reaches the exact value within 500 iterations,as in the case for no interaction. The increasing complexity of the system affects the results greatly, and it will require more iterations than in the case for no interaction and may also require different learning rate, more hidden nodes to get more exact values.

7. PERSPECTIVE FOR FUTURE IMPROVEMENTS

In contrast to the previously delivered project 1, the current project was mainly built on the base of knowledge on how to use classes as the base for calculations. The project 1 facilitated our understanding of what classes might be useful for. However, in the considered case only two classes *system* and *rbm* were introduced. Since only fermions were investigated, the *particles* class is still missing. Here, one might also include symmetry or asymmetry of a wave function, depending on type of particles we are working with. In this framework both project 1 and project 2 can be coupled. However, both authors experienced considerable lack of time, bit still considering the idea of creating a single program, based on the work with two separate classes of bosons and fermions. Another improvement of the project 2 in contrast to the project 1 is parallelization of the code with MPI, which significantly improved the timescales authors were working with. However, only two simultaneous processes were used in the presented calculations. It leaves a large space for experiments with larger number of processes, exploiting more computational power than that provided with ordinary PCs. The authors also wish they had less examinations during the past month to test all the hypotheses they obtained during the analysis process, especially experiment with significantly larger number of iterations.

8. CONCLUSION

Under the present study it was shown that the machine learning approach with a neural network introduced is an efficient way to reproduce ground state energy of a given system. Efficiency of this approach is also based on the introduction of the Boltzmann restricted machine being combined with the variational Monte Carlo. The gradient descent method was applied in order to carry out effective optimization of the system. A neural network might be described by a large number of parameters one have to adjust to minimize the energy of a system. Thus, a graphical search of a minimum is no longer applicable, and the gradient descent provides us with effective optimization. The system of one electron in one dimension and two interacting and non-interacting electrons in two dimensional harmonic oscillator potential was considered. For all cases three different sampling types, the brute force, the Metropolis-Hastings, the

Gibbs sampling were tested. Both the brute force and the Metropolis-Hastings algorithms demonstrate similar behaviour with the latter being more noisy for low number of iterations. The Gibbs sampling demonstrates an overall result deviating from an analytical solution, which might be explained with specific choice of the wave function. The present project also aimed at the demonstration of how the learning process depends on such parameters as learning rate η, number of hidden nodes N, and distribution standard deviation σ. All the results obtained point to the improved convergence rate for larger values of learning rate and lower number of hidden nodes, i.e. the results converge to an analytical solutions faster. For large number of hidden nodes the learning process becomes to oscillate significantly, as for low values of learning rate considerably large number of iterations is required to achieve an appropriate result reproducing the benchmarks. In addition, an improved error analysis based on the blocking method was included into the present research.

9. APPENDIX A: LINK TO ALL PROGRAMS

Link to the project in Github

10. APPENDIX B: ANALYTICAL CALCULATIONS OF LOCAL ENERGY AND QUANTUM FORCE FOR THE METROPOLIS ALGORITHM

The motivation of restricted Boltzmann Machine (RBM) for quantum many body problems is to examine if it can reduce some of the difficulties that these problems incline, like the degrees of freedom for instance. Among the most successful technique to attack these challenges are artificial neural networks. For this method to work it is necessary to choose the right RBM and therefore also a wavefunction which should be a probability amplitude depending on x, this since the RBM model is given by the joint distribution of x and h:

$$\Psi(\mathbf{x}) = F_{rbm}(\mathbf{x},\mathbf{h}) = \sum_h F_{rbm}(\hat{x},\hat{h}) = \frac{1}{Z} e^{-\frac{1}{T_0}E(\mathbf{x},\mathbf{h})}, \tag{58}$$

in this study the constant $T_0 = 1$ and the same applies to the partition function since in this case it does not affect the results in any way. Since the Gaussian-Binary RBM is used the visible nodes should be continuous and the energy equation in the exponential is:

$$E(\mathbf{x},\mathbf{h}) = \sum_i^M \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_j^N b_j h_j - \sum_{i,j}^{M,N} \frac{X_i w_{ij} h_j}{\sigma_i^2}, \tag{59}$$

thus, the trial wavefunction becomes:

$$\begin{aligned}
\Psi_T(\mathbf{x}) = F_{rbm}(\mathbf{x},\mathbf{h}) = \\
&= \sum_h e^{-\sum_i^M \frac{(x_i-a_i)^2}{2\sigma^2}} \dot{e}^{\sum_j^N b_j h_j + \sum_{ij}^{N,M} \frac{x_i w_{ij} h_j}{\sigma^2}} \\
&= e^{-\sum_i^M \frac{(x_i-a_i)^2}{2\sigma^2}} \sum_{h_1}\sum_{h_2}\dots\sum_{h_N} e^{\sum_j^N b_1 h_1 + b_2 h_2 + \dots + b_N h_N} \times \\
&\times e^{\sum_i^M \frac{x_i w_{i1} h_1}{\sigma^2} + \sum_i^M \frac{x_i w_{i2} h_2}{\sigma^2} + \dots + \sum_i^M \frac{x_i w_i h_N}{\sigma^2}}.
\end{aligned} \tag{60}$$

Since h is binary in this case either 0 or 1, summing over h simply gives:

$$\Psi(\mathbf{X}) = e^{-\sum_i^M \frac{(x_i-a_i)^2}{2\sigma^2}} \times \prod_j^N \left(1 + e^{b_j + \sum_i^M \frac{x_i w_{ij}}{\sigma^2}}\right). \tag{61}$$

As given in the project description the local energy or the cost function is given by

$$E_L = \frac{1}{\Psi}\hat{H}\Psi. \tag{62}$$

In this project the Hamiltonian is given as:

$$\hat{H} = \sum_{i=1}^N \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right) + \sum_{i<j}\frac{1}{r_{ij}}, \tag{63}$$

The Hamiltonian multiplied with the trial wavefunction gives the energy:

$$\hat{H}\psi_T = \psi_T(E_{kin} + E_{pot} + E_{int}), \tag{64}$$

the kinetic energy part being:

$$\frac{1}{\psi_T}\nabla^2\psi_T = \nabla^2 \ln\psi_T + (\nabla \ln\psi_T)^2. \tag{65}$$

this results in:

$$\ln\psi = -\sum_i^M \frac{(x_i - a_i)^2}{2\sigma^2} + \sum_j^N \ln\left(1 + e^{b_j + \sum_{i=1}^M \frac{x_i w_{ij}}{\sigma^2}}\right). \tag{66}$$

In the vector form it might be rewritten as:

$$\nabla \ln\psi_T = \frac{\partial}{\partial x_1}\ln\psi_T \vec{e}_{x_1} + \frac{\partial}{\partial x_2}\ln\psi_T \vec{e}_{x_2} + \dots + \frac{\partial}{\partial x_M}\ln\psi_T \vec{e}_{x_M}. \tag{67}$$

Taking the derivative with respect to x_k:

$$\begin{aligned}
&\frac{\partial}{\partial x_k}\ln\psi_T = \\
&= -\frac{(x_k - a_k)}{\sigma^2} + \sum_j^N \frac{e^{b_j + \sum_{i=1}^M \frac{x_i w_{ij}}{\sigma^2}}}{1 + e^{b_j + \sum_{i=1}^M \frac{x_i w_{ij}}{\sigma^2}}} \times \frac{w_{kj}}{\sigma^2} = \\
&= -\frac{(x_k - a_k)}{\sigma^2} + \sum_j^N \frac{1}{1 + e^{-b_j - \sum_{i=1}^M \frac{x_i w_{ij}}{\sigma^2}}} \times \frac{w_{kj}}{\sigma^2},
\end{aligned} \tag{68}$$

from this point onwards:

$$sigmoid(u(j)) = \frac{1}{1 + e^{-b_j - \sum_{i=1}^M \frac{x_i w_{ij}}{\sigma^2}}}, \tag{69}$$

with $u(j) = -b_j - \sum_{i=1}^M \frac{x_i w_{ij}}{\sigma^2}$. A function that have this form is called the logistic sigmoid function, thus the name given to the function. The second derivative is then:

$$\frac{\partial^2}{\partial x_k^2}\ln\psi_T = -\frac{1}{\sigma^2} + \sum_j^N sigmoid(-u(j)) \times sigmoid(u(j)) \times \frac{w_{kj}^2}{\sigma^4}. \tag{70}$$

The kinetic part of the local energy function then is:

$$\begin{aligned}
&\frac{1}{\psi_T}\nabla^2\psi_T = \\
&= \nabla^2 \ln\psi_T + (\nabla \ln\psi_T)^2 = \\
&= -\frac{M}{\sigma^2} + \sum_i^M \sum_j^N sigmoid(-u(j)) \times sigmoid(u(j)) \times \frac{w_{ij}^2}{\sigma^4} + \\
&+ \sum_i^M \frac{(x_i - a_i)}{\sigma^2} - 2\sum_i^M \frac{(x_i - a_i)}{\sigma^2} \sum_j^N sigmoid(u(j)) \frac{w_{ij}^2}{\sigma^2} + \\
&+ \sum_i^M \sum_j^N (sigmoid(u(j)))^2 \frac{w_{ij}^2}{\sigma^4},
\end{aligned} \tag{71}$$

this can be written in matrix form:

$$\begin{aligned}
&\frac{1}{\psi_T}\nabla^2\psi_T = \\
&= -\frac{M}{\sigma^2} + \sum_i^N \frac{\mathbf{W_{*i}}\mathbf{W_{i*}^T}}{\sigma^4} sigmoid(-u(i)) \times sigmoid(u(i)) + \\
&+ \frac{(\mathbf{x}-\mathbf{a})^\mathbf{T}(\mathbf{x}-\mathbf{a})}{\sigma^4} - \\
&- 2\sum_i^N \frac{(\mathbf{x}-\mathbf{a})\mathbf{W_{*i}}\mathbf{W_{i*}^T}}{\sigma^4} sigmoid(u(j)) + \\
&+ \sum_i^N (sigmoid(u(j)))^2 \frac{\mathbf{W_{*i}}\mathbf{W_{i*}^T}}{\sigma^4}.
\end{aligned} \tag{72}$$

The local energy thus takes the form:

$$\begin{aligned}
&E_L = \\
&= -\frac{M}{\sigma^2} + \sum_i^N \frac{\mathbf{W_{*i}}\mathbf{W_{i*}^T}}{\sigma^4} sigmoid(-u(i)) \times sigmoid(u(i)) + \\
&+ \frac{(\mathbf{x}-\mathbf{a})^\mathbf{T}(\mathbf{x}-\mathbf{a})}{\sigma^4} - \\
&- 2\sum_i^N \frac{(\mathbf{x}-\mathbf{a})\mathbf{W_{*i}}\mathbf{W_{i*}^T}}{\sigma^4} sigmoid(u(j)) + \\
&+ \sum_i^N (sigmoid(u(j)))^2 \frac{\mathbf{W_{*i}}\mathbf{W_{i*}^T}}{\sigma^4} + \\
&+ E_{pot} + E_{int}.
\end{aligned} \tag{73}$$

Writing it in matrix/vector form illustrates better how to think of this problem computationally.

11. APPENDIX C: ANALYTICAL CALCULATIONS OF GIBBS LOCAL ENERGY

Gibbs is well described in the Method 3 section. The only difference for the calculations of the local energy with Gibbs sampling is that the trial wavefunction $\psi_T = \sqrt{F_{rbm}}$. The only difference is in the kinetic part of the local energy which is shown below:

$$\begin{aligned}
&\frac{1}{\tilde{\psi_T}}\nabla^2\tilde{\psi_T} = \\
&= \nabla^2 \ln\tilde{\psi_T} + (\nabla \ln\tilde{\psi_T})^2 = \\
&= \nabla^2(\frac{1}{2}\ln\psi_T) + (\nabla\frac{1}{2}\ln\psi_T = \\
&= \frac{1}{2}\nabla^2(\ln\psi_T) + \frac{1}{4}(\nabla\ln\psi_T)^2 = \\
&= -\frac{M}{2\sigma^2} + \frac{1}{2}\sum_i^N \frac{\mathbf{W_{*i}}\mathbf{W_{i*}^T}}{\sigma^4} sigmoid(-u(i)) \times \\
&\times sigmoid(u(i)) + \frac{1}{4}\frac{(\mathbf{x}-\mathbf{a})^\mathbf{T}(\mathbf{x}-\mathbf{a})}{\sigma^4} - \\
&- \frac{1}{2}\sum_i^N \frac{(\mathbf{x}-\mathbf{a})\mathbf{W_{*i}}\mathbf{W_{i*}^T}}{\sigma^4} sigmoid(u(j)) + \\
&+ \frac{1}{4}\sum_i^N (sigmoid(u(j)))^2 \frac{\mathbf{W_{*i}}\mathbf{W_{i*}^T}}{\sigma^4}
\end{aligned} \tag{74}$$

.

12. APPENDIX D : THE GRADIENTS FOR DIFFERENT CHOICES OF THE WAVE FUNCTION

First a reminder for the trial wavefunction that has been chosen for this study:

$$\psi_T = e^{-\sum_i^M \frac{(x_i-a_i)^2}{2\sigma^2}} \times \prod_j^N \left(1 + e^{b_j + \sum_i^M \frac{x_i w_{ij}}{\sigma^2}}\right). \tag{75}$$

For the importance sampling the gradient descent has the equations below for the different parameters a_j, b_k, W_{lm}:

$$\frac{\partial\psi_T}{\partial a_j} = \frac{(x_j - a_j)}{\sigma^2}\psi_T, \tag{76}$$

$$\frac{\partial\psi_T}{\partial b_k} = \psi_T sigmoid(u(k)), \tag{77}$$

$$\frac{\partial\psi_T}{\partial W_{lm}} = \psi_T sigmoid(u(m)) \times \frac{x_l}{\sigma^2}. \tag{78}$$

In case of alternative wave function chosen (this case is not used in the program) and therefore using the trial wavefunction $\tilde{\psi_T} = \sqrt{F_{rbm}}$, the derivatives for the parameters becomes:

$$\frac{\partial\tilde{\psi_T}}{\partial a_j} = \frac{(x_j - a_j)}{2\sigma^2}\psi_T, \tag{79}$$

$$\frac{\partial\tilde{\psi_T}}{\partial b_k} = \frac{1}{2}\psi_T sigmoid(u(k)), \tag{80}$$

$$\frac{\partial\tilde{\psi_T}}{\partial W_{lm}} = \frac{1}{2}\psi_T sigmoid(u(m)) \times \frac{x_l}{\sigma^2}. \tag{81}$$

13. APPENDIX E: THE IMPORTANT SAMPLING FOR METROPOLIS AND GIBBS

For the Important Sampling there is made use of the quantum force to make the sampling more efficient thus getting to the global minimum faster than when using the brute Metropolis sampling, this is called Metropolis-Hastings and is described in methods 3. The quantum force is defined as:

$$F = 2\frac{\nabla\psi_T}{\psi_T}, \tag{82}$$

it can be simplified further to obtain the following form:

$$\frac{\nabla\psi_T}{\psi_T} = \nabla_k \ln\psi_T \tag{83}$$

The quantum force is a vector which points in a direction with a certain size:

$$\vec{F} = 2\frac{\nabla\psi_T}{\psi_T} = 2\sum_k^M \frac{\partial}{\partial x_k}\ln\psi_T\vec{e}_{x_k}, \tag{84}$$

further calculations give:

$$F_k = -2\frac{(x_k - a_k)}{\sigma^2} + 2\sum_j^N sigmoid(u(j)) \times \frac{w_{kj}}{\sigma^2} \tag{85}$$

For the different choice of the trial wavefunction being $\tilde{\psi_T} = \sqrt{F_{rbm}}$, the quantum force will be given as:

$$\begin{aligned} F = & \\ & = 2\frac{\nabla\tilde{\psi_T}}{\tilde{\psi_T}} = 2\nabla_k \ln\tilde{\psi_T} = \nabla_k \ln\psi_T = \\ & = \frac{(x_k - a_k)}{\sigma^2} + \sum_j^N sigmoid(u(j)) \times \frac{w_{kj}}{\sigma^2} \end{aligned} \tag{86}$$

.

This case was not used in the calculations, since the quantum force is required only for the importance sampling. However, it was presented as an illustration of how the choice of the wave function might affect the form of the quantum force.

REFERENCES

[1]H.J. Morten. Computational physics - lecture notes fall 2015,(2015).
[2]M. Taut // Phys. Rev. A 48, 3561 - 3566, (1993).
[3]D. Ren, B. Wang et al // The Royal Society of Chemistry, Analytical methods, Critical review, 18,(2017).
[4]X. Wang, M. J. Ruedas-Rama et al // Analytical Letters, 40: 14971520, (2007).
[5]M. Markova, V. M. Valsdottor. Computational Physics II: Project 1, (2019) (see appendix 9)
[6]P. Teng, Phys. Rev. E 98, 033305 (2018).
[7]S. Marsland, "Machine Learning. An Algorithmic Perspective",Second edition,CRC Press, New-York, (2015).
[8]P. Broecker, J. Carrasquilla et al// Machine learning quantum phases of matter beyond the fermion sign problem, Scientific Reports 7, 8823 (2017).
[9]G. Carleo and M. Troyer, Science 355, Issue 6325, pp. 602-606 (2017)
[10]W. S. McCulloch, W. H. Pitts // Bulletin of Mathematical Byophisics, Vol. 5, p.115-133, (1943).
[11]H.J. Morten. Restricted Boltzmann Machine applied to Quantum Mechanical Problems,lectures,(2019)
[12]N. Metropolis, A.W. Rosenbluth et al.// The Journal of ChemicalPhysics 21(6), 1087. URLhttp://link.aip.org/link/?JCP/21/1087/1, (1953).