FYS4411 - Project 2

Two Electrons in a Harmonic Oscillator Trap

Alexander Sexton – Simen Håpnes – Gabriel Cabrera

(Dated: June 15, 2020)

In the research presented, we study the ground state energies of electrons in harmonic oscillator traps. We consider systems consisting of one and two particles in one, two and three dimensions. We also study the effect of a correlating force in the two-particle case, and benchmark our results with analytical expectations for the cases with known solutions. The numerical approach is to apply the restricted Boltzmann machine along with variational Monte-Carlo sampling. We have applied brute-force sampling from the Metropolis algorithm, importance sampling from Metropolis-Hastings algorithm and finally Gibbs sampling. For the one electron one-dimension system, the Metropolis algorithm and Metropolis-Hastings algorithm converged to ground state energies of $0.50000000 \pm 1.4 \cdot 10^{-8}$ a.u. and $0.5000000000 \pm 2.2 \cdot 10^{-10}$ a.u, respectively, which is in excellent agreement with theoretical values. We also find that Gibbs sampling is not well suited for studying the one-particle case. For the two electron, two-dimensional system with inter-particle interaction, we find the best method to be Gibbs sampling, which converged to a ground state energy of 3.0570 ± 0.0014 a.u., which is higher than the theoretical expectation of 3 a.u.

I. INTRODUCTION

A modern approach to solving quantum mechanical problems is by the use of numerical methods. These type of physics problems are often only analytically solvable in the most simple of cases, due to the rapid increase of mathematical complexity in multi-particle systems. In many areas of physics when the problem complexity exceeds the analytical limitations, numerical methods can provide surprisingly good models. It is important to ensure that the models are able to reproduce the theoretical expectations of the solvable problems, before extending the numerical approaches.

In the study presented, we adopt a reinforcement learning method, namely the *Restricted Boltzmann Machine* (or *RBM*). This method, while reminiscent of logistic regression (or a single-hidden-layer neural network), differs in two important ways: firstly, a direct neural network (or *DNN*) performs a series of alternating forward and back propagations during training, while the weights in our RBM are calculated via a simple gradient descent method on our system's energy function.

The second main difference is that a trained DNN is only meant to perform a single forward propagation, while an RBM will re-use its hidden-layer outputs as inputs for its visible nodes. This means that an RBM is inherently cyclical, unlike a standard DNN.

In the next section, we will present the theory upon which the project is built. We will then go through the specific methods and algorithms used in the methods section. Thereafter the findings will be presented in the results section, and interpreted in the discussion section. We will summarize our work with the most important findings in the conclusion section. For the interested reader, the full mathematical derivations are included in the appendix.

The code that was used in this research can by found in the following GitHub repository `https://github.com/alexahs/FYS4411/tree/master/project2`.

II. THEORY

A. The System

The system that will be simulated is a quantum mechanical system, since we are dealing with subatomic particles. We will simulate two electrons trapped in a harmonic oscillator (HO) trap. We will simply disregard the Pauli exclusion principle because we only consider two fermions – this is because the first two electron states only vary in terms of *spin*, and our calculations do not depend on electron spin. Future work on similar problems with *more than* two electron will have to account for Pauli's exclusion principle.

The kinetic and potential energy of the HO trap, are governed by the Hamiltonian

$$\hat{H}_0 = \frac{1}{2}\sum_{i}^{N} \nabla_i^2 + \omega^2 r_i^2. \tag{1}$$

where ω is the oscillator frequency. We have also set

$$\hbar = \omega = e = m_e = c = 1. \tag{2}$$

The repulsive interactions between the two electrons are described by

$$\hat{H}_1 = \sum_{i=1}^{P-1}\sum_{j=i+1}^{P}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{3}$$

$$= \sum_{i<j}\frac{1}{r_{ij}}. \tag{4}$$

The shorthand index notation $i < j$ simply indicates all possible pairs of $i \neq j$ exactly once. r_{ij} is defined as the absolute distance between particle i and j. The total Hamiltonian of the system is

$$\hat{H} = \hat{H}_0 + \hat{H}_1. \tag{5}$$

When adopting natural units, the total Hamiltonian simplifies to

$$\hat{H} = \frac{1}{2}\left(\nabla_1^2 + \nabla_2^2 + r_1^2 + r_2^2\right) + \frac{1}{r_{12}}. \tag{6}$$

The gradients are with respect to the particles 1 and 2. r_1, r_2 are the particles' distances to the center of the trap. It is important to remember that the scaled Hamiltonian yields energy in Hartree atomic units (a.u.). This energy unit is defined as

$$a.u. \equiv \frac{\hbar^2}{m_e a_0^2}, \tag{7}$$

where m_e is the electron mass and a_0 is the natural length scale of the system.

B. The Restricted Boltzmann Machine

We will apply a machine learning method known as the *Restricted Boltzmann Machine* (RBM). This is a reinforcement learning method. It is a two-layer network, in which one layer is visible and the other is hidden. The nodes in the visible layer are referred to as $\mathbf{X} \in \mathbb{R}^M$, where M is the product of the number of particles, P, and the number of dimensions, D. The hidden layer is referred to as $\mathbf{H} \in \mathbb{R}^N$, where N is the number of hidden nodes.

There are several types of RBMs, and we will adopt the so called Gaussian-binary version. This means that the visible nodes contains continuous values and the hidden nodes take binary values (0 or 1). There is a weight matrix, $\mathbf{W} \in \mathbb{R}^{M\times N}$ which connects the two layers. There are also two sets of biases: the visible biases, $\mathbf{a} \in \mathbb{R}^M$, and the hidden biases, $\mathbf{b} \in \mathbb{R}^N$.

In the RBM, the *joint probability distribution function* (PDF) is given as

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

Here, Z is a normalization constant, and T_0 can be ignored by setting it to equal 1. The energy function, E, of this particular Gaussian-Binary RBM, is given as

$$E(\mathbf{X}, \mathbf{H}) = \sum_i^M \frac{(X_i - a_i)^2}{2\sigma_i^2} - \sum_j^N b_j H_j - \sum_{i,j}^{M,N}\frac{X_i w_{i,j} H_j}{\sigma_i^2}. \tag{9}$$

The idea is to let the marginal PDF, $F_{RBM}(\mathbf{X})$ represent the wave function. Note that the marginal PDF is different from the joint PDF $F_{RBM}(\mathbf{X}, \mathbf{H})$, where the relationship is given by

$$F_{RBM}(\mathbf{X}) = \frac{1}{Z}\sum_{\mathbf{h}} F_{RBM}(\mathbf{X}, \mathbf{h}). \tag{10}$$

The wave function is then given by

$$\Psi(\mathbf{X}) = F_{RBM}(\mathbf{X}) \tag{11}$$

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

which we also refer to as the *neural quantum state* of the system.

When running the RBM, the idea is to apply gradient descent in order to minimize a cost function. The aim is to try to converge towards the *ground state energy*, and hence the cost function is the energy. This convergence will work by iteratively tweaking the parameters of the model, so that the wave function will converge towards lower energy states. These parameters consists of the *hidden biases*, the *visible biases* and the *weights*. We will let α_i denote any of the parameters of the model. The gradient of the local energy is

$$\frac{\partial\langle E_L\rangle}{\partial\alpha_i} = 2\left(E_L\frac{1}{\Psi}\langle\frac{\partial\Psi}{\partial\alpha_i}\rangle - \langle E_L\rangle\langle\frac{1}{\Psi}\frac{\partial\Psi}{\partial\alpha_i}\rangle\right). \tag{13}$$

Furthermore, we will apply the following derivation rule,

$$\frac{1}{\Psi}\frac{\partial\Psi}{\partial\alpha_i} = \frac{\partial\ln\Psi}{\partial\alpha_i}, \tag{14}$$

and therefore directly find the respective gradients w.r.t. the logarithm of the wave function, which is

$$\ln \Psi(\mathbf{X}) = \ln Z - \sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_j^N \ln\left(1 + e^{b_j + \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}\right). \tag{15}$$

Lastly, we will find the derivatives of equation 15 w.r.t. the hidden and visible biases and the weights of the network. These are

$$\frac{\partial \ln \Psi}{\partial a_i} = (X_i - a_i)\,\sigma^{-2}, \tag{16}$$

$$\frac{\partial \ln \Psi}{\partial b_j} = \left(e^{-b_j - \sigma^{-2}\sum_i^M X_i w_{ij}} + 1\right)^{-1}, \tag{17}$$

$$\frac{\partial \ln \Psi}{\partial w_{ij}} = X_i \left(e^{-b_j - \sigma^{-2}\sum_i^M X_i w_{ij}} + 1\right)^{-1} \sigma^{-2}. \tag{18}$$

C. Sampling Approaches

There are several available sampling approaches to apply in the Monte-Carlo procedure. The idea is to apply statistics to determine whether or not a new step is accepted. We will adopt three sampling rules, namely brute-force sampling from the Metropolis algorithm, importance sampling from the Metropolis-Hastings algorithm, and Gibbs sampling.

1. Brute-Force Sampling

The Metropolis algorithm exhibits a brute-force sampling rule, where a proposed step is chosen randomly. The old and the new state are named state i and j, respectively. Each state has a corresponding probability of the system being in that state. These probabilities are denoted as p_i and p_j. Additionally there are the transitional probabilities $T_{i\to j}$ and $T_{j\to i}$, which represent the probability of the transition illustrated in the indices. An important assumption in the regular Metropolis sampling is that the transitional probabilities are assumed to be equal, and therefore cancel out in the acceptance probability,

$$A_{i\to j} = min\left\{1, \frac{p_i T_{i\to j}}{p_j T_{j\to i}}\right\}. \tag{19}$$

$A_{i\to j}$ is the probability that the transition from state i to j is *accepted*. In the quantum mechanical system, the probabilities are represented by the absolute values of the wave functions.

Numerically, the acceptance ratio will be calculated, and then there is a binary choice: we have to either accept or reject the proposed step. This is done by comparing $A_{i\to j}$ to a randomly drawn number $r \in [0,1]$,

```
if (A > r):
    Accept move
else:
    Reject move
```

With this approach, the evolution of the system exhibits stochasticity.

2. Importance Sampling

One of the problems with Metropolis sampling, is that a significant fraction of the proposed moves are rejected. This means that the system did not evolve at that proposed step, and the code has wasted a certain number of CPU cycles. This is where the extended algorithm, namely the *Metropolis-Hastings* algorithm, serves a useful purpose. The idea is to still propose random steps, but the directions of these steps are biased in the direction of the drift force. The drift force is a vector which points in the direction where the wave function increases the most,

$$\mathbf{F} = \frac{2\nabla\Psi}{\Psi}. \tag{20}$$

In order to implement the concept of drift force in the algorithm, we introduce the Fokker-Planck equation,

$$\frac{\partial P(x,t)}{\partial t} = D\frac{\partial}{\partial x}\left(\frac{\partial}{\partial x} - F\right)P(x,t), \tag{21}$$

which is a partial differential equation from statistical mechanics. D is a diffusion coefficient, F is the drift term, P is the probability density and x denotes the position of a particle at time t. The entire equation describes the time evolution of the PDF (analogous to the wave function) of a particle which is influenced by both random motion and drift force. We also have the Langevin equation, which can be derived from the Fokker-Planck equation,

$$\frac{\partial x(t)}{\partial t} = DF(x(t)) + \eta, \tag{22}$$

where η is a random variable. By using the Langevin equation, the ratio of accepted moves will typically be significantly higher than for the brute-force Metropolis sampling scheme. The discretized Langevin equation,

$$y = x + DF(x)\Delta t + \eta\sqrt{\Delta t}, \tag{23}$$

describes a proposed move from position x to position y. Δt is the specified time-step, which is a new variable in the Metropolis-Hastings scheme. Notice how the direction will be affected by the drift force F, but there will still be some stochasticity regulated by the variable η. This sampling rule is also known as *importance sampling*, and it is a method which provides a higher acceptance ratio, which in turn will waste fewer CPU cycles.

A solution to the Fokker-Planck equation yields the transition probability: this solution is known as the *Green's function*,

$$G(y,x,\Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp\left(-\frac{(y-x-D\Delta t F(x))^2}{4D\Delta t}\right). \tag{24}$$

This will now be a part of the sampling rule. In the Metropolis algorithm, we assumed that $T_{i\to j} = T_{j\to i}$. In the Metropolis-Hastings algorithm, the Greens function will substitute the transition probabilities, yielding an acceptance probability of

$$A_{i\to j} = \min\left(1, \frac{G(x,y,\Delta t)|\Psi_T(y)|^2}{G(y,x,\Delta t)|\Psi_T(x)|^2}\right). \tag{25}$$

Notice that the scaling factor in the Green's functions cancel out.

3. *Gibbs Sampling*

A final sampling scheme which will be applied is the *Gibbs* sampling method, which can be used for our system since the wave function is *positive definite.* In this case we write our wave function as the square root of the marginal probability

$$|\Psi(\mathbf{X})|^2 = F_{rbm}(\mathbf{X}), \tag{26}$$
$$\Psi(\mathbf{X}) = \sqrt{F_{rbm}(\mathbf{X})} \tag{27}$$
$$= \frac{1}{\sqrt{Z}} e^{-\sum_i^M \frac{(X_i-a_i)^2}{4\sigma^2}} \prod_j^N \sqrt{1 + e^{b_j + \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}}. \tag{28}$$

This approach uses a two-step sampling process, where the updates to the new positions of the particles are based on the states of the hidden nodes, effectively making it a *recurrent neural network.* This is accomplished by sampling the joint probability of both the input nodes $\boldsymbol{x}$ and hidden nodes $\boldsymbol{h}$

$$P(\boldsymbol{x}|\boldsymbol{h}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{a} + \boldsymbol{w}\boldsymbol{h}, \sigma^2), \tag{29}$$

and the updated samples are then generated according to the conditional probabilities

$$P(x_i|\boldsymbol{h}) = \mathcal{N}(x_i; a_i + \boldsymbol{w}_{i*}\boldsymbol{h}, \sigma^2) \tag{30}$$
$$P(H_j|\boldsymbol{x}) = \frac{\exp\left(b_j + \frac{\boldsymbol{x}^T \boldsymbol{w_{*j}}}{\sigma^2}\right) H_j}{1 + \exp\left(b_j + \frac{\boldsymbol{x}^T \boldsymbol{w_{*j}}}{\sigma^2}\right)}. \tag{31}$$

The new samples are automatically accepted as the new configuration of the system. The logarithm of this wave function is given by

$$\ln\Psi(\mathbf{X}) = -\frac{\ln Z}{2} - \sum_i^M \frac{(X_i - a_i)^2}{4\sigma^2} + \frac{1}{2}\sum_j^N \ln\left(1 + e^{b_j + \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}\right), \tag{32}$$

and we need the expressions for its derivatives w.r.t. the hidden and visible biases and the weights of the network. These are

$$\frac{\partial \ln\Psi}{\partial a_i} = \frac{1}{2\sigma^2}(X_i - a_i), \tag{33}$$
$$\frac{\partial \ln\Psi}{\partial b_j} = \frac{1}{2}\left(e^{-b_j - \sigma^{-2}\sum_i^M X_i w_{ij}} + 1\right)^{-1}, \tag{34}$$
$$\frac{\partial \ln\Psi}{\partial w_{ij}} = \frac{1}{2\sigma^2} X_i \left(e^{-b_j - \sigma^{-2}\sum_i^M X_i w_{ij}} + 1\right)^{-1}. \tag{35}$$

D. Statistical Analysis

A statistical procedure which will be applied is *blocking*, which was popularized by Flyvbjerg and Pedersen[1], is a suitable method for large data sets. In this method we must use a sample which of size 2^d, where $d \geq 1$ is an integer. The procedure is as follows:

1. Initial sample vector $\mathbf{x}_0$.
2. Compute the next sample $\mathbf{x}_1$ where each element is the average of each subsequent neighbor-pairs from $\mathbf{x}_0$. This vector is therefore half the length of the previous.
3. Repeat until we have $\mathbf{x}_{d-1}$.

For each of the obtained vectors, we can compute the variance by

$$V(\mathbf{x}_k) = \frac{\sigma_k^2}{n_k} + \frac{2}{n_k}\sum_{h=1}^{n_k-1}\left(1-\frac{h}{n_k}\right)\gamma_k(h) \tag{36}$$

$$= \frac{\sigma_k^2}{n_k} + e_k, \tag{37}$$

where n_k is the size of vector $\mathbf{x}_k$, γ_k is the *autocovariance*, and e_k is called the *truncation error*. It is possible to show that

$$V(\mathbf{x}_k) = V(\mathbf{x}_j), \tag{38}$$

for all $k, j \in [0, d-1]$. Therefore, we have that

$$V(\mathbf{x}_0) = \frac{\sigma_k^2}{n_k} + e_k. \tag{39}$$

It was also demonstrated in ref. [1] that the sequence $\{e_k\}_{k=0}^{d-1}$ is decreasing.

III. METHOD

The code for the simulations was written in C++, and the code for the post-analysis was written in Python 3.

A. The Restricted Boltzmann Machine

The aim of the algorithms is to minimize the energy of various quantum mechanical systems. We achieve this by applying Variational Monte-Carlo along with the RBM. The RBM is constructed such that the input nodes represent the positions of the particles (one node represents the position of one particle along one dimension), and the hidden nodes are used as variable parameters. Increasing the number of hidden nodes then corresponds to increasing the complexity of the system which the RBM can capture. The wave function of the system can be evaluated from the state of the RBM, as shown in equation 12. In order to let the algorithm converge towards a ground state energy, the parameters of the RBM were updated in each outer RBM iteration. The gradient of the energy in equation 13, which was considered to be our *cost function*, required us to sample equations 33, 34 and 35 as well as the local energy itself. These *samples* were obtained by running a high number of Monte-Carlo cycles, and this gave us an expectation value of the energy and parameter gradients, which we used for optimizing the parameters w.r.t. the local energy. We have used *steepest descent* as our scheme for finding the optimal parameters. The main outline of the algorithm for performing this process for Metropolis and Importance Sampling is

```
initialize positions, weights and biases

for gradient_descent_iterations:
    equilibrate system

    for montecarlo_cycles:
        pick particle at random
        propose a move by adjusting position
        if move is accepted:
            calculate new energy
            calculate gradients

    calculate averages of energy and gradients
    calculate cost function
    update NQS parameters with gradient descent
```

The adjustment of the steps and deciding whether it is accepted or not follows the rules which are governed by the respective sampling method, as is explained in sections II C 1 and II C 2. The pseudocode above, outlines the general algorithm for the brute-force sampling and importance sampling.

After the search for the optimal parameters, we did a final run with a larger number of Monte-Carlo cycles for the purpose of collecting a large amount of data which we used in estimating the errors in the calculation.

In addition to running this scheme with the standard *Metropolis* sampling rule and *Importance* sampling, we used *Gibbs* sampling. This method automatically accepts the new configuration of the nodes in the network, and the algorithm of this approach is slightly different to the one outlined above. The updates to the network nodes in a single step are calculated by equation 31 and implemented in the following algorithm:

```
for node in hidden layer:
    calculate conditional probability for \
        activating node
    if probability < uniform(0,1):
        node = True

for node in input layer:
    calculate mean position of node
    draw value from normal distribution with \
        this mean value and sigma variance
    update node position to the drawn value
```

The RBM procedure contains two additional parameters which must be considered, the *learning rate* in the gradient descent scheme and the *number of hidden nodes* in the RBM. These two parameters were varied in order to study the properties of the various combinations, which include the numerical stability and the rate of convergence. The best parameter values were determined by implementing a grid-search.

Another important aspect of the RBM, is the initialization of the weights and biases. These are always initialized from a Gaussian distribution, $\mathcal{N}(0, \sigma_{init}^2)$, where $\sigma_{init} = 0.001$.

B. The Quantum Mechanical Systems

We wanted to study numerous quantum systems. Since we are studying fermions, the Pauli exclusion principle restricted us to an upper bound of two particles. Therefore, both one- and two-particle systems were studied. Another aspect that was varied was the number of dimensions, ranging from one-, two-, and three-dimensions. The last physical property that was varied was for two-particle systems, was to both include and exclude the correlating Hamiltonian term from equation 4. This is the term that represents the Coulomb repulsion between two electrons. In summary, the various systems were combinations of

- One and two particles,
- One-, two-, and three-dimensions,
- With and without the correlating force (only for two-particle systems).

The initial positions of the particles were drawn from a uniform distribution $r_i \in [-0.5, 0.5]$, for $i \in \{x, y, z\}$.

For brute-force sampling and importance sampling, the wave function of the system is given by the marginal PDF, $F_{RBM}(\mathbf{X})$ as shown in equation 12. As is stated in the theory section, the T_0 parameter was set to 1, so it can be ignored. In the wave function, $\sigma_i = \sigma = 1$.

For Gibbs sampling as an exception, the wave function is $\sqrt{F_{RBM}(\mathbf{X})}$. In addition, the $\sigma_i = \sigma$ was varied in order to see how it affected the energy of the system.

For all three sampling methods, the numerical results were benchmarked with analytical values. There were two reasons for these benchmarks: (1) to verify that the algorithms were consistent and able to reproduce theoretical results, and (2) to tune and study the dependencies of model parameters. The analytical values included the one-dimensional one-particle system. For the non-interacting systems, the theory predicts ground-state energies of $\frac{P \cdot D}{2}$ a.u., where P is the number of particles and D is the number of dimensions. For the systems *with* correlation, only the two-dimensional two-particle system with correlation were benchmarked. The theory predicts a ground state energy of 3 a.u. for this particular configuration.

For all of the sampling methods, natural units were applied in the Hamiltonian when evaluating the local energy $\hbar = e = c = 1$. Lastly, the harmonic oscillator frequency was set to one, $\omega = 1$.

All of the iterations resulted in large samples over energies, in which we applied the blocking method to provide statistical estimates of the means and standard errors. The blocking code we applied was provided by [2].

IV. RESULTS

In this section, various results from all experiments are listed with limited context. An in-depth analysis is instead presented in the following **Discussion** section.

A. One-Particle Systems

The experiments in which a single-particle system is initiated are listed in this subsection. Since there are only isolated electrons in these cases, there are no inter-particle forces at work.

Before storing the values used in the post-analysis, an optimization process is used to allow the energy outputs to converge towards the desired values. In FIG. 1, FIG. 2, and FIG. 3, this process is shown for the brute-force Metropolis, Metropolis-Hastings, and Gibbs sampling methods, respectively. The standard error is also visualized (in red) in these plots, though it is only visible in FIG. 3 for the Gibbs sampler. This process is always run for exactly two-hundred iterations.

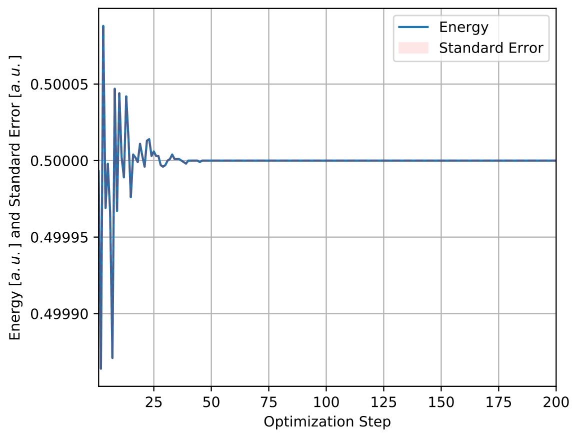

FIG. 1: The energy and variance of the 1-dimensional 1-particle system during the convergence phase, using Metropolis sampling.

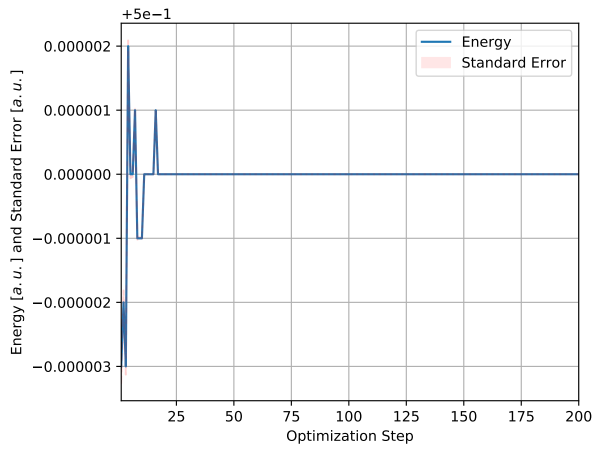

FIG. 2: The energy and variance of the 1-dimensional 1-particle system during the convergence phase, using Metropolis-Hastings sampling.

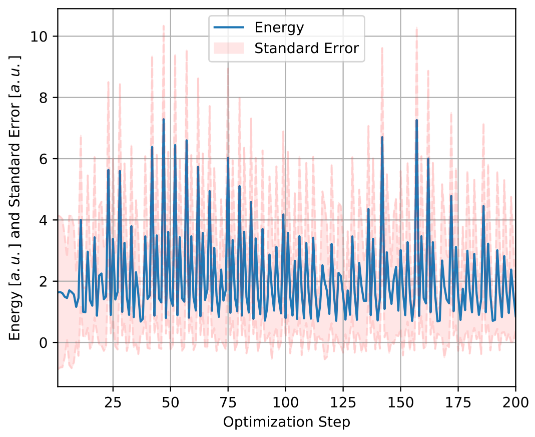

FIG. 3: The energy and variance of the 1-dimensional 1-particle system during the convergence phase, using Gibbs sampling.

Once the convergence process is complete, the calculations continue for another 1048576 cycles, though these new energy outputs are now stored in their own separate binary file. These values are the ones which we are most interested in – as such, it is the mean of these values that are considered the model's final outputs. Additionally, they are processed using the *blocking* method in order to determine the standard error of the model.

To identify which σ yields the best results for the one-dimensional, one-particle *Gibbs* sampler, a linear search is run at the beginning of the experiment; this search was run with the knowledge that we expected an analytical energy of 0.5 a u , so the best σ was the one that yielded the smallest deviation from this expected energy. These results are shown in FIG. 4, where we see that our optimum is $\sigma = 0.707$.

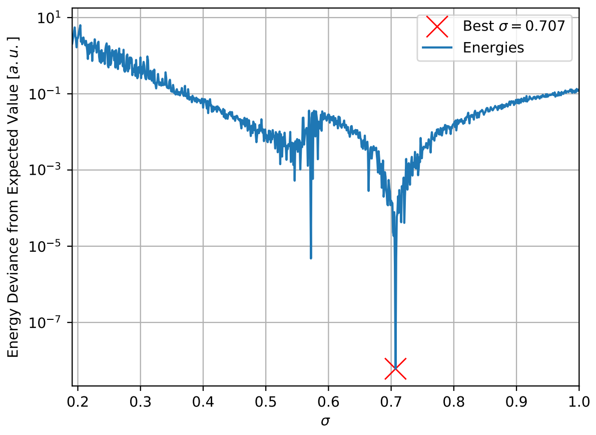

FIG. 4: The results of the linear search for an optimal σ in the normal distribution $\mathcal{N}(0, \sigma)$ for particle position initialization for a 1-dimensional 1-particle system using Gibbs sampling.

In order to determine how the number of hidden units and learning rate η affect the model's accuracy, a grid-search is performed for one to fifteen hidden nodes, and learning rates $\eta \in [0.045, 0.2]$. The resulting means and standard errors for these grid-searches are presented in FIG. 5, FIG. 6, and FIG. 7.

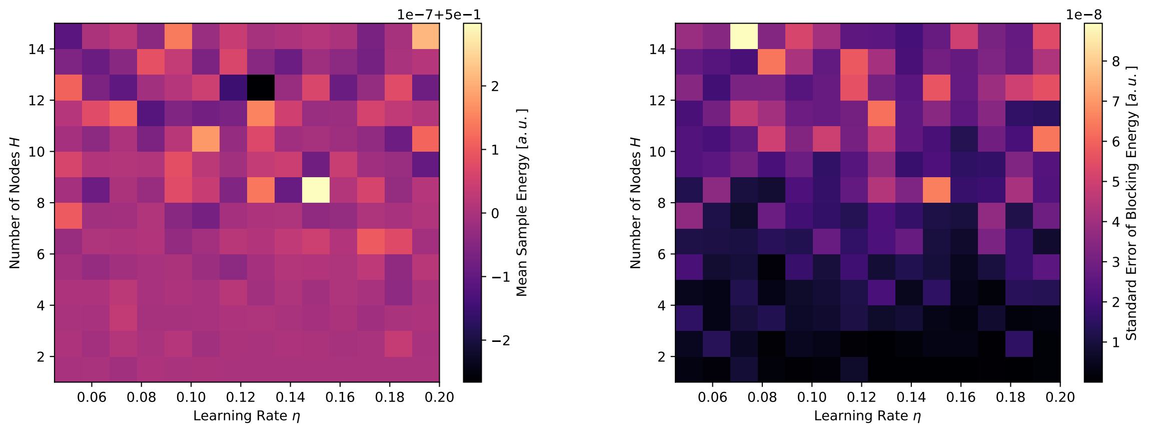

FIG. 5: A grid-search over a selection of hidden units and learning rates, for a 1-dimensional 1-particle system using Metropolis sampling.

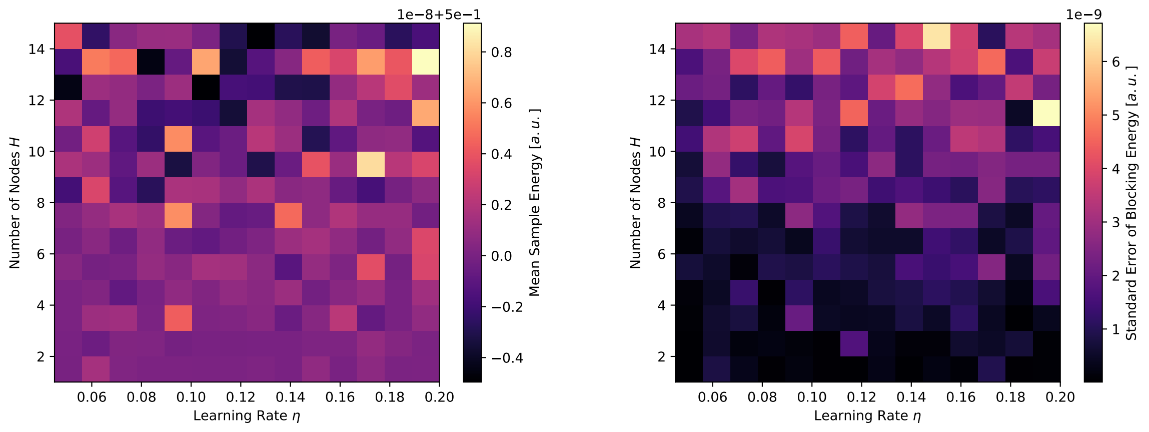

FIG. 6: A grid-search over a selection of hidden units and learning rates, for a 1-dimensional 1-particle system using Metropolis-Hastings sampling.

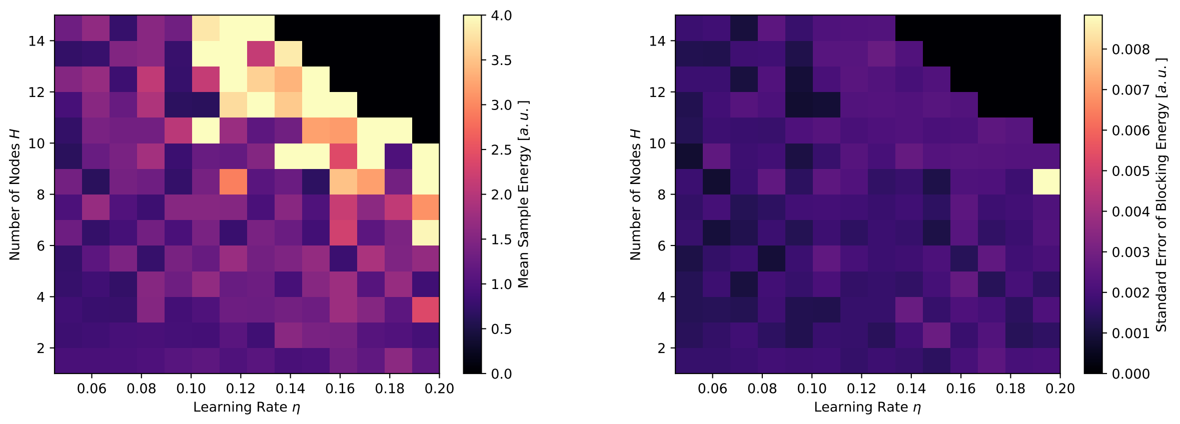

FIG. 7: A grid-search over a selection of hidden units and learning rates, for a 1-dimensional 1-particle system using Gibbs sampling.

Finally, the positions of the electron over the course of the sampling process are stored in their own binary files. These are reminiscent of a Boltzmann or Gaussian-like distribution when presented as a histogram, which can be seen for all three sampling methods in FIG. 8, FIG. 9, and FIG. 10. Note that each plot is shown only for one set of hyperparameters (no. of hidden units and η), and that these are listed in their respective captions.

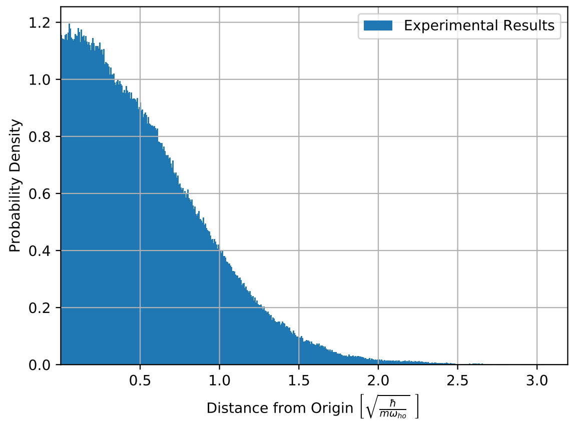

FIG. 8: The probability of a particle being at a particular distance from the origin over the course of an experiment, for a 1-dimensional 1-particle system using Metropolis sampling, with 1.0 hidden units and a learning rate of $\eta = 0.177857$.

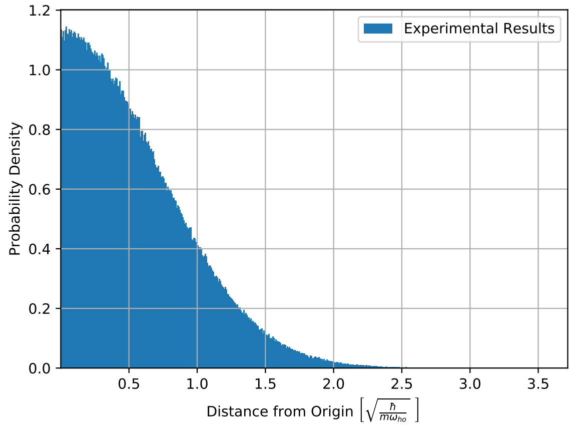

FIG. 9: The probability of a particle being at a particular distance from the origin over the course of an experiment, for a 1-dimensional 1-particle system using Metropolis-Hastings sampling, with 4.0 hidden units and a learning rate of $\eta = 0.177857$.

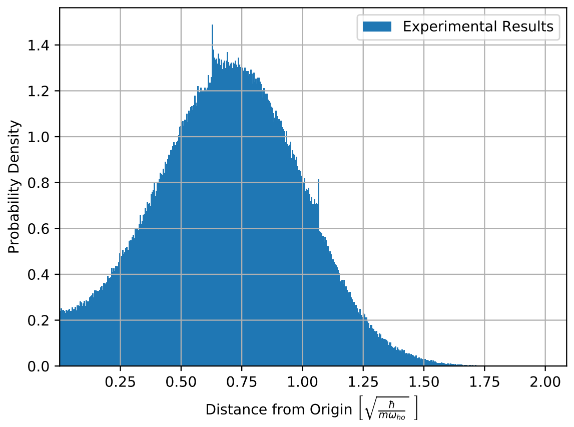

FIG. 10: The probability of a particle being at a particular distance from the origin over the course of an experiment, for a 1-dimensional 1-particle system using Gibbs sampling, with 15.0 hidden units and a learning rate of $\eta = 0.045$.

B. Two-Particle Systems

The next results are those taken from experiments involving two electrons, and includes the interacting force between them. These particles' positions are also initialized by a normal distribution, and are therefore optimized for the Gibbs sampler using a linear search (just like previously in FIG. 4) – this can be seen in FIG. 11, where the optimal value was shown to be $\sigma = 0.667$

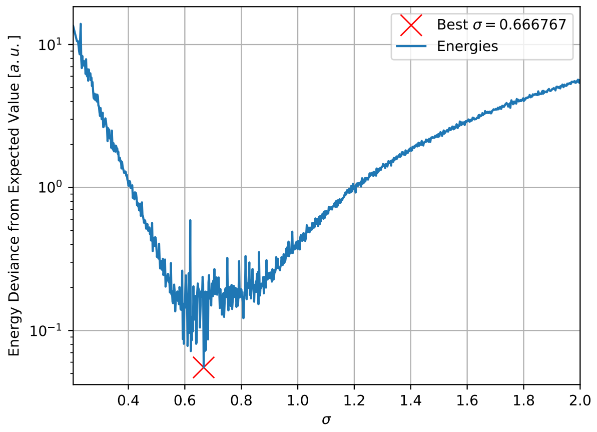

FIG. 11: The results of the linear search for an optimal σ in the normal distribution $\mathcal{N}(0, \sigma)$ for particle position initialization for a 2-dimensional 2-particle system using Gibbs sampling.

As in the one-particle case, a grid-search was performed following the optimization phase over the same hyperparameter values (hidden units in range one to fifteen, and $\eta \in [0.045, 0.2]$) and number of cycles (1048576). The resulting means and standard errors are visualized in FIG. 12 and FIG. 13.

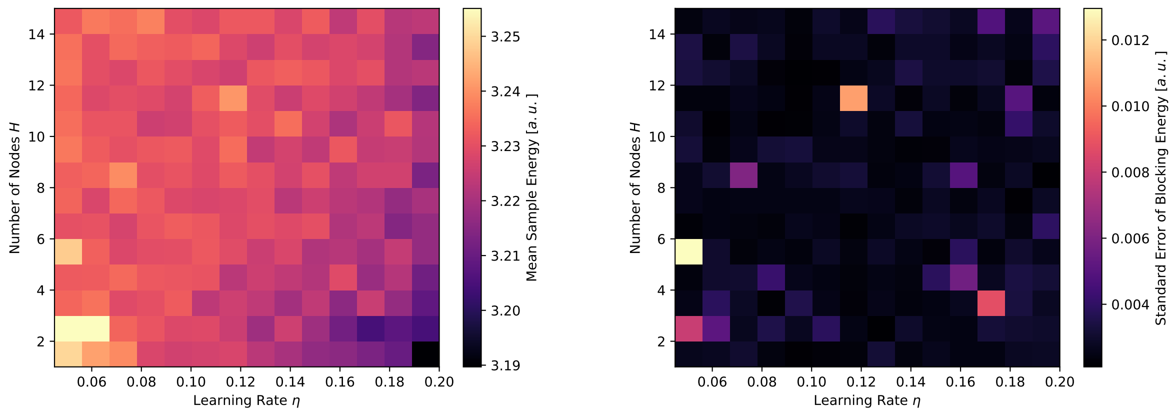

FIG. 12: A grid-search over a selection of hidden units and learning rates, for a 2-dimensional 2-particle system using Metropolis-Hastings sampling.

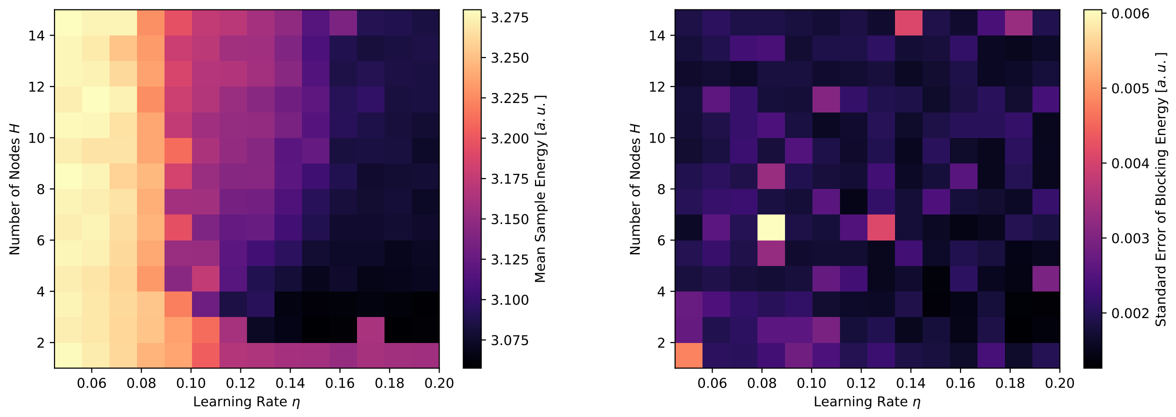

FIG. 13: A grid-search over a selection of hidden units and learning rates, for a 2-dimensional 2-particle system using Gibbs sampling.

Finally, the particles' positions are stored just as in the one-particle case, and their positions are once more visualized in the form of a histogram (see FIG. 14) – one should however note that the distribution represents the position of just *one* of the particles in question, rather than both.

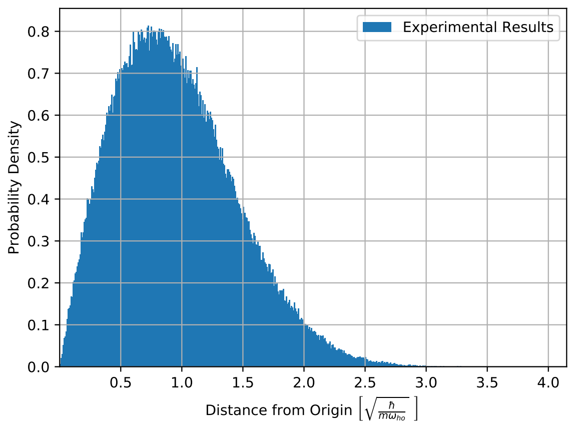

FIG. 14: The probability of a particle being at a particular distance from the origin over the course of an experiment, for a 2-dimensional 2-particle system using Metropolis-Hastings sampling, with 2.0 hidden units and a learning rate of $\eta = 0.2$.

C. Summary of Experiments

In order to compare the models in an effective manner, TABLE I contains the most important and best-performing results from all six grid-searches, and includes all the hyperparameter values, simulation settings, and post-processed simulation outputs. Though it is not explicitly shown, the one-particle, one-dimensional case has an analytical solution of $E = \frac{1}{2}\mathrm{a\,u}$, and the two-particle, two-dimensional case has an analytical solution of $E = 3\mathrm{a\,u}$.

TABLE I: Comparison of results for 6 experiments, each with 1048576 cycles.

Particles	Dimensions	Sampling Method	Hidden Units	Learning Rate η	Mean [a u]	Std. Error [a u]
1	1	Metropolis	4	0.178	0.500	1.40491×10^{-8}
1	1	Metropolis-Hastings	4	0.178	0.500	2.20138×10^{-10}
1	1	Gibbs	4	0.178	1.691	0.00202036
2	2	Metropolis-Hastings	2	0.200	3.185	0.00264646
2	2	Gibbs	2	0.200	3.057	0.00137064
2	3	Metropolis-Hastings	2	0.200	3.798	0.000914501

V. DISCUSSION

A. One-Particle Systems

As seen in the previous section, there were many unsurprising results, as well as a few that were somewhat unexpected. In FIG. 1 and FIG. 2, we see that the convergence of the energy in the optimization process for Metropolis and Metropolis-Hastings sampling converged quickly and correctly, giving us the expected analytical solution of $\frac{1}{2}\mathrm{a\,u}$, which has served as an effective benchmark for our code. We see that Metropolis-Hastings sampling converged quicker than Metropolis sampling due to the added drift-force, which caused the electron to be pushed in the direction of higher expected positional probability density. On the other hand, we see that Gibbs sampling in FIG. 3 did not yield good results, given that it does not manage to converge to the expected energy value.

When using Metropolis and Metropolis-Hastings sampling, the value of σ in the wave function was fixed to $\sigma = 1$; in the case of Gibbs sampling, we needed to tune this parameter. This was accomplished via a linear search, visualized in FIG. 4, where it can be seen that the the accuracy of the model has a direct dependency on the selected value for σ. Although we found an optimal value for σ, this sampling method performed weakly for the one-particle, one-dimensional case, which is later shown in TABLE I, where $E = 1.691\mathrm{a\,u}$.

In FIG. 5 and FIG. 6, the vast majority of the configurations give values that are extremely close to the expected analytical solution, with a very low standard error. Once again, we note that Metropolis-Hastings (FIG. 6) performed slightly better than Metropolis sampling (FIG. 5), which is likely due to the individual cycles' higher effectiveness in Metropolis-Hastings.

On the other hand, FIG. 7 once again demonstrates that Gibbs sampling is outright ineffective at properly modeling a one-particle, one-dimensional system, as the energy values are overall quite incorrect, and their respective standard errors are each very large in comparison to the other sampling methods.

Intuitively speaking, the one-particle system for the given potential should eventually result in the particle being close to the origin, rather than at some arbitrary point in the distance (due to the potential being symmetric). We see in FIG. 8 and FIG. 9, that the preferred position of the electron is clearly at the origin, as seen by the Boltzmann-distributed density. Once again, we see that Gibbs under-performs in FIG. 10, due to the distribution being centered around an arbitrary point (approximately $0.7 \cdot \sqrt{\hbar/m\omega_{ho}}$).

B. Two-Particle Systems

In the two-particle systems, we wanted to explore the effect of the variable σ in the expression for the wave function. In FIG. 11, we illustrated the converged energy after applying the Gibbs sampling rule on a two-dimensional system with correlation. For this sampling rule, the wave function is defined as the square root of the marginal PDF. The expected ground state energy for this system is 3 a.u. (where 2 a.u. comes from the regular Hamiltonian, and 1 a.u. is due to the correlated Hamiltonian). FIG. 11 shows the deviance from this expectation value, and we can clearly see the trend of a single valley with a minimum. The best results are achieved at $\sigma = 0.667$, and the local region about this value is noisy, and the figure strongly indicates that σ both smaller and larger than the optimal value achieves worse results.

FIG. 12 shows a grid search over a number of hidden nodes N, and learning rate η, when applying the Metropolis-Hastings algorithm on the two-particle two-dimensional system with correlation. The left plot shows the achieved ground state energy and the right plot shows the corresponding standard error. Since the theoretical expected value is 3 a.u., the figure indicates that higher learning rates and fewer hidden nodes provided the best results. The standard errors are mostly below 0.004 a.u. for most of the configurations, except for a few. There is presumably no obvious explanation for these few configurations resulted in significantly higher standard errors.

FIG. 13 shows a grid search over number of hidden nodes N, and learning rate η, when applying the Gibbs sampling rule on the two-particle two dimensional system with correlation. The left plot shows the ground state energies of which the algorithm converged towards, and there are some interesting patterns in the plot. The darker regions are closest to the theoretical expectation. In Metropolis-Hastings algorithm for the same system, two hidden nodes provided the best results, but for Gibbs sampling, two hidden nodes are too few; there was even one case with two hidden nodes which were too few. The two sampling rules both show the same general trend, but the Gibbs sampling rule is stronger affected and governed by the model parameters. The right plot in FIG. 13 shows the corresponding standard errors to the ground state energies, and they are low for the entire grid, and presumably only governed by stochasticity.

FIG. 14 shows the position distribution as a histogram of distance from origin for one of the particles in the two-particle two dimensional system with correlation. The position distribution closely resembles a Maxwell-Boltzmann distribution. An important aspect of this figure, is that we cannot see any indications of irregularity, as the sampled positions produced a smooth known probability distribution. A possible situation which could have occurred was to initialize the two particles very close to each other, in which could have resulted in a numerically large and unstable correlation term. If this had happened, we would expect to see a significant number of positions sampled at larger distances from the origin. We also assumed that given the few number of particles, there is a very low probability that the two particles are initialized close to each other.

C. Summary of Experiments and Future Improvements

In TABLE I we showed the configurations of the models which produced the best results for each of the three sampling methods. When comparing these, we notice that the configurations of the RBM have more hidden units for the system of interacting particles than for the system with only one particle, which is quite surprising. Intuitively, we would expect that bigger systems would require a more complex model, but this is not the case here. We also notice that the estimated errors in the one-particle case are very low when using Metropolis and Metropolis-Hastings compared to the two-particle system, which may be explained by the fact that the one-particle system is more stable. In the two-particle system we have errors which are orders of magnitudes larger for Metropolis and Metropolis-Hastings compared to the one-particle system, but these errors are likely underestimated, as the expected theoretical value lies outside our error estimate. When using Gibbs sampling, however, the expected theoretical value lies within our error estimate, and therefore determine this sampling method the superior one for correlated systems. Another aspect of the Gibbs sampler is that it is superior in terms of speed of the algorithm as well. One of the big advantages it has compared to Metropolis and Metropolis-Hastings in this regard, is that it automatically accepts the new configuration of the system, rather than having to evaluate the ratio of the wave functions of the new and old configuration.

When optimizing the parameters, we used a fixed number of iterations in gradient descent, and we can see from FIG. 1 and FIG. 2 that many of the iterations are effectively wasted, since the energy converges so quickly. We could have made the code much more efficient by breaking out of the gradient descent loop if the energies of previous iterations have stayed constant. This would have made the grid search of the learning rate and number of hidden nodes much quicker. With the Gibbs sampling method in the one-particle system, this was not the case, and we were unable to minimize the energy within the number of gradient descent iterations. We could have attempted to increase the number of iterations, but considering the oscillating behaviour of the energy minimization in FIG. 3, we find it unlikely that the energy would converge.

VI. CONCLUSION

Overall, the results of this project and the simulated experiments yielded satisfactory results. It was determined that while some methods were very effective overall (namely the Metropolis and Metropolis-Hastings algorithms), others only performed well in certain cases – for instance, the Gibbs sampler only performed well for the two-particle case, and in fact superseded the other methods.

In order to calculate these energy results, a grid-search was performed for each particle-dimensionality-sampler combination over the hyperparameters N and η (the hidden nodes and learning rate, respectively) in order to find the best values for that particular configuration – a clear trend was found for Gibbs sampling, which was especially in the two-particle case. Additionally, a linear-search was performed in order to calculate σ in the normal distribution governing particle position initialization for the Gibbs sampling method with success, as a clear trend was found showing that selecting σ is a vital step in developing an effective model.

In the one-particle, one-dimensional case, we expected to find energies $E = 1/2\,\mathrm{a\,u}$; while both the Metropolis and Metropolis-Hastings algorithms yielded this value to a very high numerical precision, the standard errors were 1.405×10^{-8} and 2.201×10^{-8}, respectively. As for the Gibbs sampler, the best result was $E_{gibbs} = 1.691$ with a standard error of 2.024×10^{-3}; this was shown to be due to a lack of proper convergence.

On the other hand, the two-particle, two-dimensional case (where the expected energy was $E = 3\,\mathrm{a\,u}$) yielded the best results when using Gibbs sampling, where $E_{gibbs} = 3.057$, and $E_{metropolis-hastings} = 3.185\,\mathrm{a\,u}$. Additionally, their respective standard errors were 1.371×10^{-3} and 2.646×10^{-3}, implying that the Gibbs method is overall superior when it comes to two-particle, two-dimensional systems.

In the one-particle case, it is expected that the electron should mostly reside at the origin, due to the harmonic oscillator potential and lack of interacting forces. We were therefore able to confirm the soundness of our interpretations by observing the particles' positional probability distributions – from these, we once more managed to confirm the conclusions made about the effectiveness of each algorithm; we observed that the one-particle, one-dimensional displacements were centered about the origin for the Metropolis and Metropolis-Hastings algorithms, while the Gibbs sampler was centered at $R \approx 0.7 \cdot \sqrt{\hbar/m\omega_{ho}}$, once more pointing to its ineffectiveness in the one-particle case.

[1] H. Flyvbjerg and H. G. Petersen. Error estimates on averages of correlated data. *The Journal of Chemical Physics*, 91(1):461–466, 1989.

[2] Morten Hjorth-Jensen. Compphysics/computationalphysics2, Feb 2019.

VII. APPENDIX

A. Finding the Local Energy Analytically

For a system of P particles of dimension D, we are given a *Gaussian* wave function $\Psi(\mathbf{X})$ of the form

$$\Psi(\mathbf{X}) = \frac{1}{Z}\exp\left(-\sum_{i}^{M}\frac{(X_i - a_i)^2}{2\sigma^2}\right) \prod_{j}^{N}\left(1+\exp\left[b_j + \sum_{i}^{M}\frac{X_i w_{ij}}{\sigma^2}\right]\right), \tag{40}$$

with a standard deviation σ, and such that $M \equiv P\cdot D$. Here, $\mathbf{X}\in\mathbb{R}^M$ is a matrix containing the particles' positions, and consists of individual particles $X_i \in \mathbb{R}$. We also have the matrix of *visible biases* $\mathbf{a}\in\mathbb{R}^M$ with elements $a_i \in \mathbb{R}$.

Additionally, we must also account for the hidden layers of our *Boltzmann machine*, thereby defining N as its number of *hidden nodes*. As such, this gives us our *hidden weights* $\mathbf{w}\in\mathbb{R}^{N\times M}$ and our *hidden biases* $\mathbf{b}\in\mathbb{R}^N$.

To keep things simple, we will separate our wave function into two factors Ψ_A and Ψ_B such that

$$\Psi_A(\mathbf{X}) = \exp\left(-\sum_{i}^{M}\frac{(X_i - a_i)^2}{2\sigma^2}\right) \tag{41}$$

and

$$\Psi_B(\mathbf{X}) = \prod_{j}^{N}\left(1+\exp\left[b_j + \sum_{i}^{M}\frac{X_i w_{ij}}{\sigma^2}\right]\right), \tag{42}$$

giving us $\Psi(\mathbf{X}) = \frac{1}{Z}\Psi_A(\mathbf{X})\Psi_B(\mathbf{X})$.

To find the *local energy* of this wave function, we also require a *Hamiltonian operator*, given by

$$\hat{H} = \hat{H}_0 + \hat{H}_1,$$

which consists of two terms – the *unperturbed* part $\hat{H}_0$,

$$\hat{H}_0 = \sum_{p=1}^{P}\left(-\frac{1}{2}\nabla_p^2 + \frac{1}{2}\omega^2 r_i^2\right), \tag{43}$$

and the *interacting* part $\hat{H}_1$,

$$\hat{H}_1 = \sum_{i<j}\frac{1}{r_{ij}}. \tag{44}$$

Here, r_i is a scalar value denoting the absolute distance from the origin of the i^{th} particle in our system.

To be more precise, assuming we have a position $\mathbf{R}_i \in \mathbb{R}^D$ (not to be confused with r_i) This can be defined as $r_i = \sqrt{\sum_{j=1}^{D} R_{ij}^2}$, where $R_{ij}\in\mathbb{R}$ is the dimension-wise position of particle $\mathbf{R}_i$.

Additionally, we have $r_{ij}\in\mathbb{R}$, which is defined to be the absolute distance between the i^{th} and j^{th} particles in our system, such that $r_{ij} = |\mathbf{R}_i - \mathbf{R}_j|$.

The local energy we wish to calculate is the *energy eigenvalue* of our Hamiltonian with respect to the given wave function, so we wish to solve

$$\hat{H}\Psi(\mathbf{X}) = E\Psi(\mathbf{X})$$

for the local energy E. We will begin by applying the Laplace operator $\nabla_p^2 = \nabla_p\cdot\nabla_p$ to Ψ; to keep things simple, we will first calculate the gradient

$$\nabla_p\Psi(\mathbf{X}) = \Psi_B(\mathbf{X})\nabla_p\Psi_A(\mathbf{X}) + \Psi_A(\mathbf{X})\nabla_p\Psi_B(\mathbf{X}). \tag{45}$$

In order to ensure compatibility between our gradients and our positions, we will rewrite our wave function factors in terms of our positions $\mathbf{R}$ such that

$$\Psi_A(\mathbf{X}) = \exp\left(-\sum_{p}^{P}\sum_{d}^{D}\frac{(R_{pd} - A_{pd})^2}{2\sigma^2}\right) \tag{46}$$

and

$$\Psi_B(\mathbf{X}) = \prod_{j}^{N}\left(1+\exp\left[b_j + \sum_{p}^{P}\sum_{d}^{D}\frac{R_{pd}W_{pdj}}{\sigma^2}\right]\right), \tag{47}$$

where $\mathbf{A}\in\mathbb{R}^{P\times D}$ is the reshaped vector a, and $\mathbf{W}\in\mathbb{R}^{P\times D\times N}$ is the reshaped matrix $\mathbf{w}$.

Eq. (46) can be written in vector form:

$$\Psi_A(\mathbf{X}) = \exp\left(-\frac{1}{2\sigma_2}\sum_{p}^{P}(\mathbf{R}_p - \mathbf{A}_p)^2\right),$$

and eq. (47) can likewise be written in vector form, using a dot product:

$$\Psi_B(\mathbf{X}) = \prod_{k}^{N}\left(1+\exp\left[b_k + \frac{1}{\sigma^2}\sum_{p}^{P}\mathbf{R}_p\cdot\mathbf{W}_{p*k}\right]\right).$$

It is also worth noting that we define $\nabla_p^T \equiv \begin{pmatrix}\frac{\partial}{\partial R_{p1}} & \frac{\partial}{\partial R_{p2}} & \cdots & \frac{\partial}{\partial R_{pD}}\end{pmatrix}$.

B. The Gradient of Ψ_A

We begin with our first factor (note that the gradient index has changed in order to account for the varying indices of the wave function):

$$
\begin{aligned}
\nabla_i \Psi_A &= \nabla_i \exp\left(-\sum_p^P \sum_d^D \frac{(R_{pd}-A_{pd})^2}{2\sigma^2}\right) \\
&= -\Psi_A \nabla_i \sum_p^P \sum_d^D \frac{(R_{pd}-A_{pd})^2}{2\sigma^2} \\
&= -\frac{\Psi_A}{2\sigma^2} \nabla_i \sum_d^D (R_{id}-A_{id})^2 \\
&= -\frac{\Psi_A}{2\sigma^2} \sum_d^D \frac{\partial}{\partial R_{id}} (R_{id}-A_{id})^2 \hat{\mathbf{u}}_{id}
\end{aligned}
$$

which gives us

$$
\nabla_p \Psi_A = -\frac{\Psi_A}{\sigma^2}(\mathbf{R}_p - \mathbf{A}_p) \tag{48}
$$

C. The Gradient of Ψ_B

Next, we need to calculate our second gradient

$$
\nabla_i \Psi_B(\mathbf{X}) = \nabla_i \prod_j^N \left(1 + \exp\left[b_j + \sum_p^P \sum_d^D \frac{R_{pd}W_{pdj}}{\sigma^2}\right]\right).
$$

Clearly, implementing the product rule in this case will be messy, so we will temporarily redefine our terms such that

$$
\psi_j \equiv 1 + \exp\left(b_j + \sum_p^P \sum_d^D \frac{R_{pd}W_{pdj}}{\sigma^2}\right), \tag{49}
$$

giving us

$$
\begin{aligned}
\nabla_i \Psi_B(\mathbf{X}) &= \nabla_i \prod_j^N \psi_j = \sum_k^N \left(\nabla_i \psi_k \prod_{j\neq k}^N \psi_j\right) \\
&= \sum_k^N \left(\frac{\nabla_i \psi_k}{\psi_k} \prod_j^N \psi_j\right) = \left(\prod_j^N \psi_j\right)\left(\sum_k^N \frac{\nabla_i \psi_k}{\psi_k}\right) \\
&= \Psi_B \sum_k^N \frac{\nabla_i \psi_k}{\psi_k}.
\end{aligned}
$$

We may now calculate our gradient in a straightforward manner:

$$
\begin{aligned}
\nabla_i \psi_k &= \nabla_i \left(1 + \exp\left[b_k + \sum_p^P \sum_d^D \frac{R_{pd}W_{pdk}}{\sigma^2}\right]\right) \\
&= \nabla_i \exp\left(b_k + \frac{1}{\sigma^2}\sum_p^P \sum_d^D R_{pd}W_{pdk}\right) \\
&= (\psi_k - 1)\nabla_i \left(b_k + \frac{1}{\sigma^2}\sum_p^P \sum_d^D R_{pd}W_{pdk}\right) \\
&= \left(\frac{\psi_k - 1}{\sigma^2}\right)\left(\nabla_i \sum_p^P \sum_d^D R_{pd}W_{pdk}\right) \\
&= \left(\frac{\psi_k - 1}{\sigma^2}\right)\left(\nabla_i \sum_d^D R_{id}W_{idk}\right) \\
&= \left(\frac{\psi_k - 1}{\sigma^2}\right)\sum_d^D \left(W_{idk}\frac{\partial}{\partial R_{id}} R_{id}\hat{\mathbf{u}}_{id}\right) \\
&= \left(\frac{\psi_k - 1}{\sigma^2}\right)\sum_d^D (W_{idk}\hat{\mathbf{u}}_{id}) \\
&= \left(\frac{\psi_k - 1}{\sigma^2}\right)\begin{pmatrix} W_{i1k} \\ W_{i2k} \\ \vdots \\ W_{iDk}\end{pmatrix}
\end{aligned}
$$

Plugging this into our previous expression gives us our gradient

$$
\nabla_p \Psi_B(\mathbf{X}) = \frac{\Psi_B}{\sigma^2}\sum_k^N \left[\left(\frac{\psi_k - 1}{\psi_k}\right)\begin{pmatrix} W_{p1k} \\ W_{p2k} \\ \vdots \\ W_{pDk}\end{pmatrix}\right].
$$

With the use of sub-indexing, we can condense the above:

$$
\nabla_p \Psi_B(\mathbf{X}) = \frac{\Psi_B}{\sigma^2}\sum_k^N \left[\left(\frac{\psi_k - 1}{\psi_k}\right)\mathbf{W}_{p*k}\right]. \tag{50}
$$

D. The Second Derivative

Before we calculate our second derivatives, we must combine our previous results, giving us

$$\nabla_p \Psi(\mathbf{X}) = \frac{\Psi_A \Psi_B}{\sigma^2 Z} \left(\sum_k^N \left[\frac{\psi_k - 1}{\psi_k} \mathbf{W}_{p*k} \right] + \mathbf{A}_p - \mathbf{R}_p \right).$$

To find ∇_p^2, we must calculate the divergence (or gradient dot-product) of the above expression, which requires one more instance of the product rule. To make this more concise, we will define two more expressions

$$\Psi_C(p) \equiv \sum_k^N \left(\frac{\psi_k - 1}{\psi_k} \mathbf{W}_{p*k} \right),$$

and

$$\Psi_D(p) \equiv \mathbf{R}_p - \mathbf{A}_p.$$

Using these re-definitions, we can compress our previous expression such that

$$\nabla_p \Psi(\mathbf{X}) = \frac{\Psi}{\sigma^2} \left(\Psi_C - \Psi_D \right). \tag{51}$$

Now we will differentiate again, using the fact that Laplace operator is equivalent to a gradient dot-product; we will need to use the product rule yet again:

$$\begin{aligned} \nabla_p \cdot \nabla_p \Psi(\mathbf{X}) &= \frac{1}{\sigma^2} \nabla_p \cdot \left(\Psi \left[\Psi_C - \Psi_D \right] \right) && (52) \\ &= \frac{1}{\sigma^2} \left(\nabla_p \Psi \cdot \left[\Psi_C - \Psi_D \right] \right. \\ &\quad \left. + \Psi \left[\nabla_p \cdot \Psi_C - \nabla_p \cdot \Psi_D \right] \right) && (53) \end{aligned}$$

This means that we must differentiate two more expressions: $\nabla_p \cdot \Psi_C$ and $\nabla_p \cdot \Psi_D$.

1. *The Gradient of* Ψ_C

We begin by applying an initial product rule such that

$$\begin{aligned} \nabla_p \cdot \Psi_C &= \nabla_p \cdot \sum_k^N \left(\frac{\psi_k - 1}{\psi_k} \mathbf{W}_{p*k} \right) \\ &= \sum_k^N \nabla_p \cdot \left(\left[1 - \frac{1}{\psi_k} \right] \mathbf{W}_{p*k} \right) \\ &= \sum_k^N \left(\nabla_p \cdot \mathbf{W}_{p*k} - \nabla_p \cdot \frac{\mathbf{W}_{p*k}}{\psi_k} \right). \end{aligned}$$

Next, we need to apply the quotient rule to our fraction:

$$\nabla_p \cdot \Psi_C = \sum_k^N \left(\nabla_p \cdot \mathbf{W}_{p*k} - \frac{\psi_k \nabla_p \cdot \mathbf{W}_{p*k} - \mathbf{W}_{p*k} \cdot \nabla_p \psi_k}{\psi_k} \right).$$

Since $\mathbf{W}_{p*k}$ is constant with respect to our gradient, we can simplify our expression:

$$\nabla_p \cdot \Psi_C = \sum_k^N \left(\frac{\mathbf{W}_{p*k} \cdot \nabla_p \psi_k}{\psi_k} \right).$$

We already have the expression for $\nabla_p \psi_k$, so we can plug it into the above,

$$\begin{aligned} \nabla_p \cdot \Psi_C &= \frac{1}{\sigma^2} \sum_k^N \left(\frac{\psi_k - 1}{\psi_k} \mathbf{W}_{p*k} \cdot \mathbf{W}_{p*k} \right) && (54) \\ &= \frac{1}{\sigma^2} \sum_k^N \sum_d^D \left(\frac{\psi_k - 1}{\psi_k} W_{pdk}^2 \right). && (55) \end{aligned}$$

2. *The Gradient of* Ψ_D

The last gradient we need to calculate is as follows:

$$\begin{aligned} \nabla_p \cdot \Psi_D &= \nabla_p \cdot (\mathbf{R}_p - \mathbf{A}_p) = \nabla_p \cdot \mathbf{R}_p \\ &= \begin{pmatrix} \frac{\partial}{\partial R_{p1}} \\ \frac{\partial}{\partial R_{p2}} \\ \vdots \\ \frac{\partial}{\partial R_{pD}} \end{pmatrix} \cdot \begin{pmatrix} R_{p1} \\ R_{p2} \\ \vdots \\ R_{pD} \end{pmatrix} \\ &= \frac{\partial R_{p1}}{\partial R_{p1}} + \frac{\partial R_{p2}}{\partial R_{p2}} + \cdots + \frac{\partial R_{pD}}{\partial R_{pD}} \\ &= D \end{aligned}$$

E. Combining the Parts

We must now combine the terms given in eq. (53):

$$\begin{aligned}
&\nabla_p^2 \Psi = \\
&\frac{1}{\sigma^2}\left(\nabla_p \Psi \cdot [\Psi_C - \Psi_D] + \Psi\left[\nabla_p \cdot \Psi_C - \nabla_p \cdot \Psi_D\right]\right) \\
&= \frac{\Psi}{\sigma^4}\left(\left[\sum_k^N \left(\left[1 - \frac{1}{\psi_k}\right] \mathbf{W}_{p*k}\right) - \mathbf{R}_p - \mathbf{A}_p\right]^2\right. \\
&\left. + \sum_k^N \sum_d^D \left(\left[1 - \frac{1}{\psi_k}\right] W_{pdk}^2\right) - D\right).
\end{aligned}$$

Although the result is somewhat messy, the energy eigenvalue can thereby be written as follows:

$$\begin{aligned}
E = \frac{1}{2\sigma^4}\sum_p^P &\left(- \left[\sum_k^N \left(\left[1 - \frac{1}{\psi_k}\right] \mathbf{W}_{p*k}\right) - \mathbf{R}_p - \mathbf{A}_p\right]^2 \right. \\
&\left. - \sum_k^N \sum_d^D \left(\left[1 - \frac{1}{\psi_k}\right] W_{pdk}^2\right) + D + \omega^2 |\mathbf{R}_p| \right) + \sum_{i<j} \frac{1}{r_{ij}}.
\end{aligned} \tag{56}$$