FYS4411 - Project 2

Geir Tore Ulvik, Kari Eriksen, Robert Solli

June 16, 2018

Abstract

In this work we explore two fermions in a harmonic oscillator trap both with and without a repulsive coloumb interaction. In place of the traditional ansatz for the wavefunction we optimize a Restricted Boltzmann Machine (RBM) which is in the family of generative machine learning models. We find that the Metropolis algorith has both better stability and performance over alternative sampling models explored. In additon to the Metropolis algorithm versions with Importance and Gibbs sampling were implemented. For systems in two and three dimensions the importance sampling algorithm was quite unstable under variations of the hyperparameters like learning rate and the initial spread of the sampling space. The Gibbs sampler we were unable to imlpement in a satisfactory manner. Opting to use the Metropolis implementation we also explored adding a simple Coloumb interaction to the hamiltonian. For the $N = D = 2$ case we observe that the model is able to come within one decimal place of the analytical value of 3 a.u. at approximately 3.1 ± 0.2 a.u.

1 Introduction

The aim of this project is to apply Restricted Boltzmann Machine (RBM) on the quantum many-body-problem. We seek the ground state energy of a system containing N particles, however we will not extend the case of two particles. The reason for this is that for two electrons in a quantum dot with frequency $\hbar\omega = 1$ we have a closed form solutions to the ground state energy. We also have analytic answers for the non-interacting case. This is convenient in that we can compare results from our numerical methods.

Another reason to limit the amount of particles is that we operate with three optimization parameters. This requires a lot of CPU time.

Our wave function is represented by the energy of the RBM, eq. (2). We produce input to feed the RBM using Monte Carlo method with Metropolis Hasting or Importance sampling algorithm for selecting states. With stochastic gradient descent we are able to optimize the parameters, the weights W_{ij} and bias functions $a_i(v_i)$ and $b_j(h_j)$.

We also want to experiment with Gibbs sampling. For this sampling method we can use an easier version of the wave function since we know it to be positive definite. The expressions of both energy and the derivatives for the gradient descent can be found in the Appendix C and B.

2 Theory

2.1 Ansatz for the wavefunction and the hamiltonian

We are studying a system of two electrons confined in a harmonic oscillator trap described by the Hamiltonian

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

where the first sum is the standard harmonic oscillator part and the last is the interacting part between the electrons and N represent the number of particles. ω is the oscillator frequency of the trap and r_i is the position of particle i, whereas r_{ij} is the distance between the particles and given as $r_{ij} = |\mathbf{r_i} - \mathbf{r_j}|$.
Our wave function is given from the energy of the restricted Boltzmann machine, see (4), which is the joint energy functional between the visible and hidden nodes. From the marginal probability of the joint probability distribution, see (7), we get our wave equation.

$$\begin{aligned}\Psi(X) &= F_{rbm}(X) &(2)\\ &= \frac{1}{Z}\exp\left(-\sum_{i}^{M}\frac{(X_i - a_i)^2}{2\sigma^2}\right)\prod_{j}^{N}\left(1 + \exp\left(b_j + \sum_{i}^{M}\frac{X_i\omega_{ij}}{\sigma^2}\right)\right) &(3)\end{aligned}$$

Here Z is the partition function, X_i represents the visible nodes running up to M, and a_i and b_j are the biases described in the section below, (2.3), where number of hidden nodes j runs up to N. ω_{ij} is an $M \times N$ matrix holding the weights connecting the visible nodes with the hidden and σ is the standard deviation of the noise in our model.

2.2 Variational Monte Carlo

Monte Carlo simulations are widely used methods in numerical science that employs random walkers. In this project we are taking a closer look at trapped bosons. We are given a trail wave function we assume is as close to the real case as possible, $\Psi_T(\mathbf{R};\alpha)$ where $\mathbf{R} = (\mathbf{R}_1, ..., \mathbf{R}_N)$ is the position of the different particles. From quantum mechanics we know the probability distribution is given by the wave function.

$$P(\mathbf{R};\alpha) = \frac{|\Psi_T(\mathbf{R};\alpha)|^2}{\int |\Psi_T(\mathbf{R};\alpha)|^2 d\mathbf{R}}$$

Monte Carlo integration allow us to evaluate the integral at hand. The expectation value of the Hamiltonian is given as follows.

$$\langle\widehat{\mathbf{H}}\rangle = \frac{\int d\mathbf{R}\Psi^*(\mathbf{R})H(\mathbf{R})\Psi(\mathbf{R})}{\int d\mathbf{R}\Psi^*(\mathbf{R})\Psi(\mathbf{R})}$$

The variational principle states that the expectation value of the Hamiltonian is an upper-bound for the ground state energy of the Hamiltonian.

$$E_0 \leq \langle H\rangle$$

This is what the Variational Monte Carlo method bases itself on. Given a probability distribution we can evaluate the wave function and look for a local minimum. We define the local energy by

$$\widehat{\mathbf{E}}_L(\mathbf{R};\alpha) = \frac{1}{\Psi_T(\mathbf{R};\alpha)}\widehat{\mathbf{H}}\Psi_T(\mathbf{R};\alpha).$$

Then the expectation value of the local energy is given by

$$\langle\widehat{\mathbf{E}}_L\rangle = \int P(\mathbf{R})\widehat{\mathbf{E}}_L d\mathbf{R} \approx \frac{1}{N}\sum_{i=1}^{N}\mathbf{E_L}(x_i)$$

where N is the number of Monte Carlo cycles. Now we can calculate the probability distribution and the local energy. And for each cycle we propose a new configuration $\mathbf{R}_p$ for the system and hopefully we come closer to the ground state.

$$\mathbf{R}_p = \mathbf{R} + r * \Delta\mathbf{R}$$

2.3 Restricted Boltzmann Machine

Restricted Boltzmann machine is a energy-based generative model which include hidden and visible variables [3]. It consists of a two-layer network with a two dimensional matrix W_{ij} telling how strong the connections between the hidden and visible nodes are. The energy related to a configuration of the nodes is what forms the basis of our model. It is given in (4).

$$E(\mathbf{v},\mathbf{h}) = -\sum_i a_i(v_i) - \sum_\mu b_j(h_j) - \sum_{ij} W_{ij}v_ih_j \tag{4}$$

Here $a_i(v_i)$ and $b_j(h_j)$ are bias functions of the visible and hidden layers which we are allowed to choose our self. In our case we want the visible layer to take a continuous form and the hidden layer to be binary, (Gaussian-binary), meaning a_i and b_j takes the following from

$$a_i(v_i) = \frac{v_i^2}{2\sigma_i^2}, \quad b_j(h_j) = b_jh_j.$$

We are working with restricted Boltzmann machine meaning there is no connection between nodes within layers, only between layers. Also this network is a generative one so we want our network to learn a probability distribution. We begin with the joint probability distribution of the visible and hidden nodes is given in (5) where Z is the partition function, see (6).

$$F_{rbm}(\mathbf{X}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{X},\mathbf{h})} \tag{5}$$

$$Z = \int \int \frac{1}{Z} e^{-E(\mathbf{X},\mathbf{h})} d\mathbf{x} d\mathbf{h} \tag{6}$$

From (5) we can marginalize over all the hidden units and get the distribution over the visible units. And as mentioned above this is what we use to represent our wave function.

$$F_{rbm}(\mathbf{X}) = \sum_{\mathbf{h}} F_{rbm}(\mathbf{X}, \mathbf{h}) \tag{7}$$

$$= \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{X},\mathbf{h})} \tag{8}$$

Since we have no data to train we use the technique of reinforcement learning. We feed the network with input using Monte Carlo method and based on the variational principle we find the ground state of the system by seeking the configuration that gives the lowest quantum mechanical energy. According to this principle we change the weights and biases by gradient descent method and hopefully the network will converge towards the correct state.

2.4 The Metropolis Algorithm

How we select the new configurations during the simulation is given by the Metropolis Algorithm. Ideally we would have a transition probability matrix which told us how likely it would be for each particle to move in the configuration space. Since we don't we try and model it. We define the following entities:

$P_i^{(n)} \rightarrow$ probability of finding the system in state i at the n'th step.
$j \rightarrow$ possible new step
$A_{i\rightarrow j} \rightarrow$ probability of acceptance.
$T_{j\rightarrow i} \rightarrow$ probability of making the transition

Now we can say that a transition probability matrix can be constructed by $T_{j\rightarrow i} A_{j\rightarrow i}$. The probability of finding the system in state i at step n is

$$P_i^{(n)} = \sum_j \left[P_j^{(n-1)} T_{j\rightarrow i} A_{j\rightarrow i} + P_j^{(n-1)} T_{i\rightarrow j} (1 - A_{i\rightarrow j}) \right]. \tag{9}$$

We want to push the system towards high density regional space of $P_i^{(n)}$. /We want to select states according to the probability distribution. In that way to converge to the desired stationary distribution p_i.

$$P_i^{(n\to\infty)} \to p_i$$

Using this statement and the fact that the sum over all possible transitional probabilities is one we can rewrite the above equation 9.

$$\sum_j [p_j T_{j\to i} A_{j\to i} - p_i T_{i\to j} A_{i\to j}] = 0$$

In order to stop at the configuration where we have reached equilibrium we us the condition of detailed balance, and it gives us

```
Data: matrix $\mathbf{R}$
Result: float $\mathbf{E}_L$
number of MC-cycles $N$;
initialize $\mathbf{R}$;
calculate $|\Psi_T(\mathbf{R})|^2$;
for $i$ in $[0, N]$ do
    $\mathbf{R}_p = \mathbf{R} + r * \Delta\mathbf{R}$;
    calculate $\omega = \frac{|\Psi_T(\mathbf{R}_p)|^2}{|\Psi_T(\mathbf{R})|^2}$;
    if $q \leq \omega$ then
        accept new move;
    else
        reject new move;
    end
    update energy, $\mathbf{E}_L$;
end
```

Algorithm 1: Monte Carlo with Metropolis-Hastings

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

We choose to accept when the acceptance is bigger than 1.

$$A_{j\to i} = \min\left(1, \frac{p_i T_{i\to j}}{p_j T_{j\to i}}\right)$$

What we must calculate is equation 10. And an example of the code is given further down, in Algorithm 1

$$\frac{p_i T_{i\to j}}{p_j T_{j\to i}} = \frac{|\Psi_T(\mathbf{R}_p)|^2}{|\Psi_T(\mathbf{R})|^2} \tag{10}$$

2.5 Importance Sampling

Now we move to an extension of the Metropolis algorithm, the Importance sampling. This method provides us with a better way of suggesting new moves. The expression for the new move, y, is given by the solution of the Langevin equation. It reads

$$y = x + DF(x)\Delta t + \xi\sqrt{\Delta t},$$

where ξ is a gaussian random variable, D is the diffusion coefficient and is $\frac{1}{2}$ and Δt is the time step which take values between $[0.001, 0.01]$. $F(x)$ is the drift force and is given by the gradient of the wave function.

$$F = 2\frac{1}{\Psi_T}\nabla\Psi_T$$

It is the drift force that ensures us we move particles towards regions of configuration space where the trail wave function is large. This method increases the efficiency of our program, since the standard Metropolis can suggest new moves in every direction with same probability. From the solution of the Fokker-Planck equation we get a new transition probability, the Green's function.

$$G(y, x, \Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}}\exp(-(y - x - DF(x)\Delta t)^2/4D\Delta t)$$

Now equation 10 from Metropolis algorithm changes to

$$\frac{p_i T_{i\rightarrow j}}{p_j T_{j\rightarrow i}} = \frac{G(x, y, \Delta t)|\Psi_T(y)|^2}{G(y, x, \Delta t)|\Psi_T(x)|^2},$$

and we must calculate the drift force in order to find new moves and the Green's function to accept/reject these moves.

2.6 Gibbs Sampling

A special case of the Metropolis sampling is when we know the wave function to be positive definite. We can then express the wave function as in (**??**) and since we do not need to worry about a complex wave function we can model the probability density as we wish. Now we sample new configurations from the conditional probabilities directly. This we can do since all nodes in each layer are independent from one another.

$$P(H_j = 1|\mathbf{x}) = \frac{1}{1 + e^{-b_j - \sum_i \frac{x_i w_{ij}}{\sigma^2}}} \tag{11}$$

We first calculate values for the hidden nodes from the current configuration, which in the beginning is initialized with random numbers. Here we recognize the Sigmoid function. From this we begin generating new samples for the visible values. These however are picked from a normal distribution with mean equal $a_i + \sum_j \mathbf{w}_{ij}\mathbf{h}$ and variance equal σ^2.

$$P(X_i|\mathbf{h}) = \mathcal{N}(X_i; a_i + \sum_j \mathbf{w}_{ij}\mathbf{h}, \sigma^2) \tag{12}$$

The acceptance rate is 1, so the configuration is change every time. We then update the energy as before. Accept now we operate with a different wave function so the expression for both the energy and the derivatives in the gradient descent method are different, with a factor 0.5. So this must be accounted for. When evaluating the wave function we take a ratio and this factor vanishes, so changing the wave function itself is unnecessary.

2.7 Gradient Descent

Like most problems in physics the variational quantum problem is one of optimization. In the last project we optimizied the expectation value of the energy over the single parameter α. Though in general interesting problems in many-body problems will not have one, but very many parameters to optimize. This is part of the motivation for the RBM implementation. That while the number of parameters are significantly larger than one, it may still be preferable to a more traditional ansatz for the wave function. In this project we then seek to optimize over the RBM parameters $\alpha = \{\mathbf{a}, \mathbf{b}, \mathbf{w}\}$. Put more formally we seek to minimize the expecattion of the energy over the aforementioned variational parameters by performing gradient descent. More sophisticated methods for optimization are available but will not be explored here. We're then left with the simple GD algorithm as before.

$$\alpha_{i+1} = \alpha_i - \gamma \frac{\partial \langle E_L(\alpha_i) \rangle}{\partial \alpha_i} \tag{13}$$

The differentiation to find the expecation values are a bit more laborious now than for the previous project , but they are listed in appendix D

2.8 Blocking Method

For the statistical analysis and error estimates we have used the technique of blocking. We are looking for the expectation value of the ground state energy of our system. In order to say something about how accurate these results are we want to look at the standard deviation.

If our samples where uncorrelated we could calculate the standard deviation through the following equation

$$\sigma = \sqrt{\frac{1}{n-1}\left(\langle E_L^2 \rangle - \langle E_L \rangle^2\right)}.$$

It can be showed that for correlated samples the equation for the standard deviation is

$$\sigma = \sqrt{\frac{1 + 2\tau/\Delta t}{n-1}\left(\langle E_L^2 \rangle - \langle E_L \rangle^2\right)}$$

where τ is the time between one sample and the next uncorrelated sample and Δt is the time between each sample. If we knew what τ was we could simply find our σ, but since we don't we use blocking to find it. The method is simple enough, we divide our samples of data into blocks and calculating the mean of each block. So if we have an amount of samples of $\langle E_L \rangle$, we divide these into M blocks and calculate the mean. By plotting the standard deviation as function of the block size we can keep blocking and blocking until we see that the std. dev. stops increasing, and this is where the blocks are uncorrelated. Now we have an estimate for τ and thus the standard deviation.

3 Method

3.1 Importance sampling

The importance sampling implementation uses a Greens function and drift force (quantum force) calculations to imporove the metropolis algorithm such that the suggested positions are better thus increasing the acceptance rate of the sampling process.

```
// Generate distributions
uniform_int_distribution<int> dis_r(0, N_p - 1);
uniform_real_distribution<double> dis_p(0.0, 1.0);
normal_distribution<double> dis_zeta(0, 1);

// Calculate Greens' function
for(int i = 0; i < M; i++) {
        Greens += pow((R(i) - R_p(i)-0.5*dt*F_drift_proposed(i))
           ,2) + pow((R_p(i) - R(i) - 0.5*dt*F_drift(i)),2);
}
Greens = exp(Greens/(4*0.5*dt));

// Metropolis
double eps = dis_p(*gen);
if(eps < P*Greens){
        R = R_p;
        return psi_t.E_l(R);
}
else{
        return prev_E_l;
}
```

3.2 Gibbs sampling

Our Gibbs sampler works quite simple updating the hidden nodes according to (11), a listing of the code can be seen below. For each number of hidden unit we calculate the new value hj(j) given the conditional probability.

```
    for(int j = 0; j < N; j++){

        double sum_xi_wij = 0;
        for(int i = 0; i < M; i++){
            sum_xi_wij += R(i)*psi_t.W(i, j);
        }

        double Hj = psi_t.b(j) + (sum_xi_wij/sigma_sq);
        double z = 1/(1 + exp(-Hj));
        hj(j) = z;
    }
```

As for the visible units they are drawn from a normal distribution according to (12) and an illustration is seen below. Here the mean is the variable my and standard deviation is σ.

```
    for(int i = 0; i < M; i++){

        double sum_hj_wij = 0;
        for(int j = 0; j < N; j++){
            sum_hj_wij += psi_t.W(i, j)*hj(j);
        }
        double my = psi_t.a(i) + sum_hj_wij;
        //Creating a normal dist
        uniform_real_distribution<double> dis(my, sigma_sq);
        R(i) = dis(*gen);
    }
```

4 Results

4.1 Metropolis Sampling

For the simulations a highest learning-rate value of $\gamma = 0.4$ was chosen based on recommendations in [1]. Figure 2 along with table 1 shows that the neural network tends to converge faster when the learning rate is increased, and stays stable at the ground state, when using brute force metropolis sampling. $\gamma = 0.4$ is clearly the best learning rate of the tested ones.

Mean time per mc-cycle		$5.88 \cdot 10^{-5}$ s
γ	Mean $\langle E_L \rangle$ a.u	Mean Variance: σ^2
$1 \cdot 10^{-3}$	4.36	$3.07 \cdot 10^{-2}$
$1 \cdot 10^{-2}$	3.57	$1.97 \cdot 10^{-2}$
$1 \cdot 10^{-1}$	3.005	$1.97 \cdot 10^{-3}$
$4 \cdot 10^{-1}$	3.002	$1.45 \cdot 10^{-3}$

Table 1: Table of learning rates γ and mean expectation values for the local energy E_L with mean variance, for brute force metropolis sampling of 2 particles in 3 dimensions, with 4 hidden nodes. Mean CPU-time per monte-carlo cycle across all iterations at the top. For corresponding plot, see figure 2.

4.2 Importance Sampling

For importance sampling (figure 3, table 2) the data follows the same trend. With importance sampling the precision is lower than that of brute force metropolis, but the mean variance is lower. For the best converging value of gamma, a simulation with the same parameters, but varying the timestep dt was done (figure 4, table 3). From the figure and data in the table, the dt parameter influences mainly the variance, lowering it by $\frac{1}{3}$ per order of magnitude. Mean variance and data analysis was produced using the blocking method ([2]).

4.3 Gibbs sampling

Figure 5, 6 and 7 show simulations for the system using Gibbs sampling. In 5 we see the results for one particle for 1, 2 and 3 dimensions. For all cases we see that the higher γ the faster the gradient descent method moves towards an the expectation energy, however they are not the right values. In the case of one particle in one dimension the expectation value keeps dropping to infinity.

For the two particle case we still have a factorization problem, the energy stabilizes at wrong energies.

In figure 7 we see how the simulations evolve using different values of standard deviation σ. For small σ we draw new values for the visible units from a smaller range in position

Mean time per mc-cycle		$5.80 \cdot 10^{-4}$ s
γ	$\langle E_L \rangle$ a.u	Mean Variance: σ^2
$1 \cdot 10^{-3}$	3.62	$2.16 \cdot 10^{-2}$
$1 \cdot 10^{-2}$	3.149	$6.09 \cdot 10^{-3}$
$1 \cdot 10^{-1}$	3.1011	$7.51 \cdot 10^{-4}$
$2 \cdot 10^{-1}$	3.0959	$6.91 \cdot 10^{-4}$

Table 2: Table of learning rates γ and corresponding expectation values obtained for the local energy E_L with variance, for importance sampling of 2 particles in 3 dimensions, with 4 hidden nodes. Mean CPU-time per monte-carlo cycle across all iterations at the top. 4 hidden nodes were used. For corresponding plot, see figure 3.

Mean time per mc-cycle		$5.24 \cdot 10^{-4}$ s
dt	$\langle E_L \rangle$ a.u	Mean Variance: σ^2
$1 \cdot 10^{-3}$	3.099	$9.18 \cdot 10^{-3}$
$1 \cdot 10^{-2}$	3.103	$3.43 \cdot 10^{-3}$
$1 \cdot 10^{-1}$	3.106	$1.13 \cdot 10^{-3}$

Table 3: Table of timesteps dt and corresponding expectation values obtained for the local energy E_L with variance, for importance sampling of 2 particles in 3 dimensions, with 4 hidden nodes. Mean CPU-time per monte-carlo cycle across all iterations at the top. For corresponding plot, see figure 4.

space and for bigger σ we draw from a broader range. Calculating new moves with a higher standard deviation could cause the method to moves faster. And we see that the slope for high σ is steeper than for smaller σ. However the energies are again wrong.

4.4 Interaction

We add the simple coloumb interaction to the program and opt to use the brute force metropolis algorithm as it was the most stable over the model hyperparameters, e.g. γ. Simulatig for the $N = D = 2$ case granted results as can be seen in figure 1 and table 4. We observe an instability around the reported analytic mean of 3 a.u. And also no seeming great impact of variations of the paramete γ on the scale $1e-1$. More rigorous analysis of the significance of these changes were however not performed.

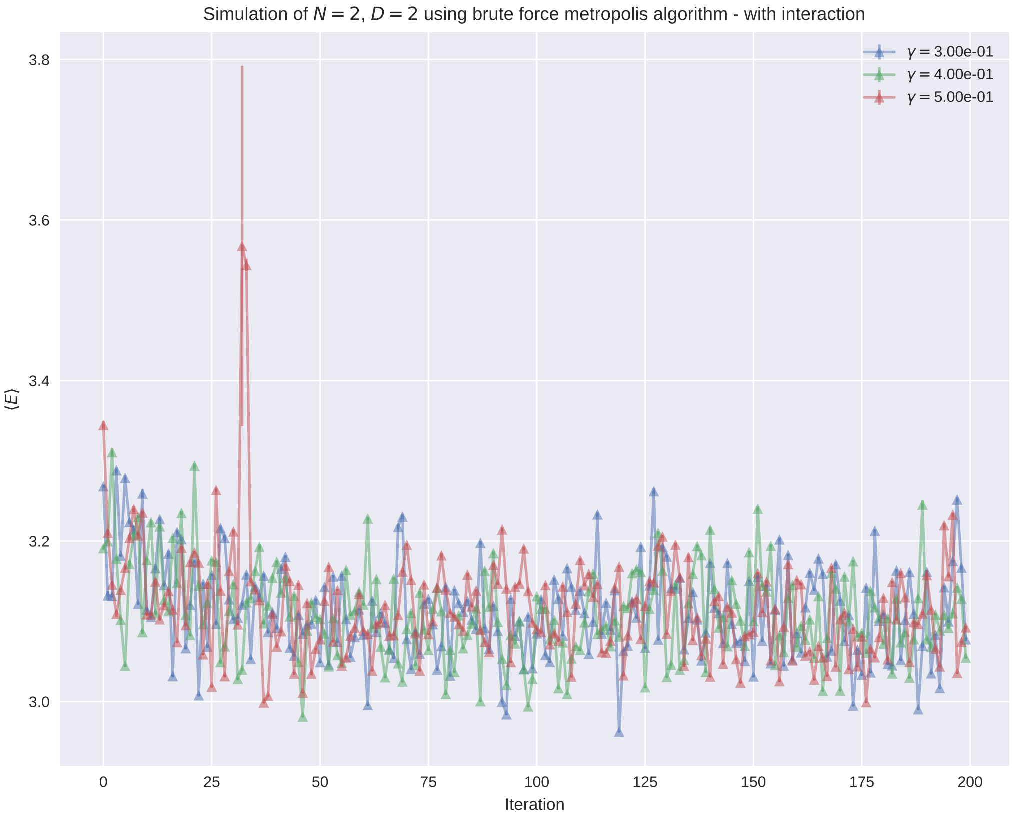

Figure 1: Plotting the expectation value of the energy on the vertical axis versus the gradient descent step on the horizontal shows an oscillating behaviour. In this case the hamiltonian for the system has been modified to include a interactive term based on the distance between the two particles. As noted in the project description this system has an analytic minimum of 3 a.u. that the RBM touches, but does not stabilize to. See table 4 for details on the stability for the different values of γ

γ	$mean(\langle E\rangle)$	$mean(\sigma^2)$
$3.00e-01$	3.10432	$4.11e-02$
$4.00e-01$	3.10004	$4.07e-02$
$5.00e-01$	3.10643	$4.15e-02$

Table 4: Simulating the $N = D = 2$ system with an added coloumb interaction led to a more oscillating behaviour than for the non-interactive case. This can be seen in figure 1. The simulation was run for 200 gradient descent iterations, the means were taken over the last 160 of them to capture only the oscillation and not the convergence. We observe that gamma does not seeem to have a dramatic impact on the stability in this case. The mean time for the simulations was $2.97e-04$ seconds

5 Discussion

5.1 Brute force metropolis

Using the simplest version of the metropolis algorithm we've shown good results for the convergence towards analytical values for two non-interactive fermions. In comparison to a more traditional VMC method we encounter more typical machine learning problems. Working with hyperparameters like γ and the magnitude of the sampling step in the metropolis algorithm exposes (especially with importance sampling) problems with glassy energy landscapes. Wherein the optimum is fairly localized and close to zero outside leading to gradient descent breaking down. We did not explore methods to remedy this problemm, though it should be noted that this exists in the literature. While the tuning of the variational parameters are fairly expensive for two particles how the method scales is the true test of it's efficacy in the field.

5.2 Importance sampling

Modifying the brute force metropolis algorithm to use a quantum force (drift force) to sample more efficiently shows sub-optimal results for the ground-state energy, but is more stable than the brute force method when it converges. Importance sampling has shown itself to be much more sensitive to parameter tuning than the brute force method, resulting in different values of gamma used when comparing. The method often does not converge at all - rather, it diverges and then stabilizes several orders of magnitute too high (or low) - when using learning rates $\gamma > 0.2$. This seems to indicate that the increased complexity of the sampling algorithm doesn't necessarily provide better results, keeping in mind that the problem it is applied to is of a simpler character.

5.3 Gibbs sampling

When it comes to the Gibbs sampling we have had some difficulties along the way and can not say we got the sampler to work. First we've had big problems with initializing the system, with starting value of the energy sometimes getting below the expectation value. This is a problem since the energy then only keeps sinking without end, never reaching the true expectation value.

Another problem is the calculation of the hidden and visible nodes. We are essentially accepting all new configurations compared to standard Metropolis. Making it a less reliable method than Metropolis.

5.4 The interactive case

We observe that the importance sampling algorithm was very sensitive to the tuning of hyperparameters and we were not able to get the gibbs-sampler to perform even for the non-interactive case. We opted to use the brute force method to investigate the interactive case of two fermions. The brute force method achieves great, but spurious accuracy as it touches the analytic minimum of 3 a.u. However, it does not hold this solution and averages at about 3.1 a.u. with a standard deviation of about 0.2. This estimate does not seem wholly unreasonable and we can conclude that using a generative method as a replacement to a more traditional ansatz of the wavefunction may not be optimal but is at least feasible.

6 Conclusion

In this project we've shown the possible use cases of a generative machine learning algorithm applied to a simple many-body phyisics problem. For two non-interactive fermions we've shown convergence to analytical values when simulatig with a VMC + RBM model with a simple metropolis algorithm. Further we implemented two different sampling schemes to try and improve the model. From our results it would seem importance sampling is very sensitive to the hyper-parameters of the model while we were unable to produce satisfactory results with the gibbs sampler at all.

A Figures for brute force metropolis and importance sampling

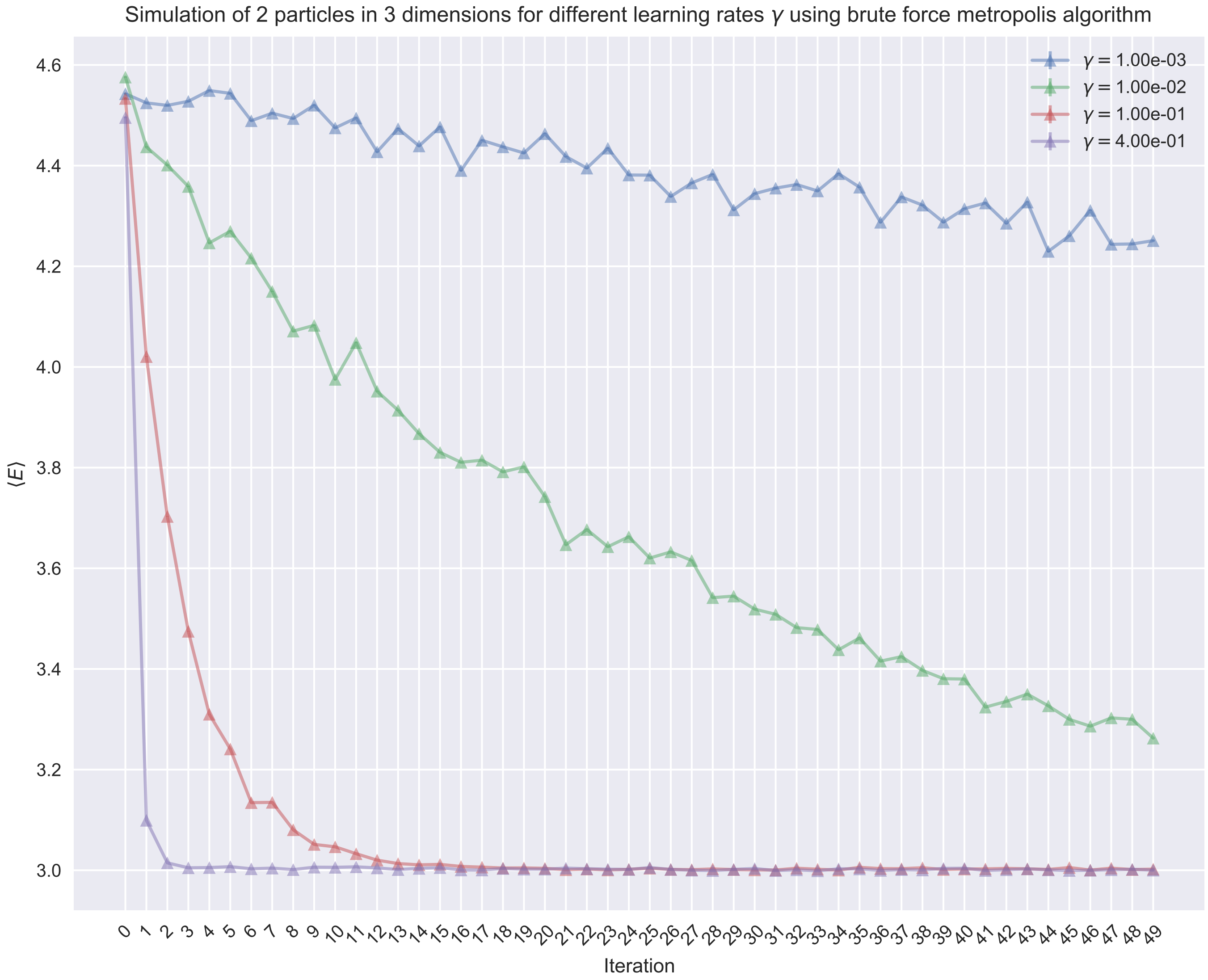

Figure 2: Plot of simulation with 2 particles in 3 dimensions with varied learning rates γ, using brute force metropolis sampling. 2 hidden nodes were used. For detailed values on energy, variance, and CPU-time per mc-cycle, see table 1

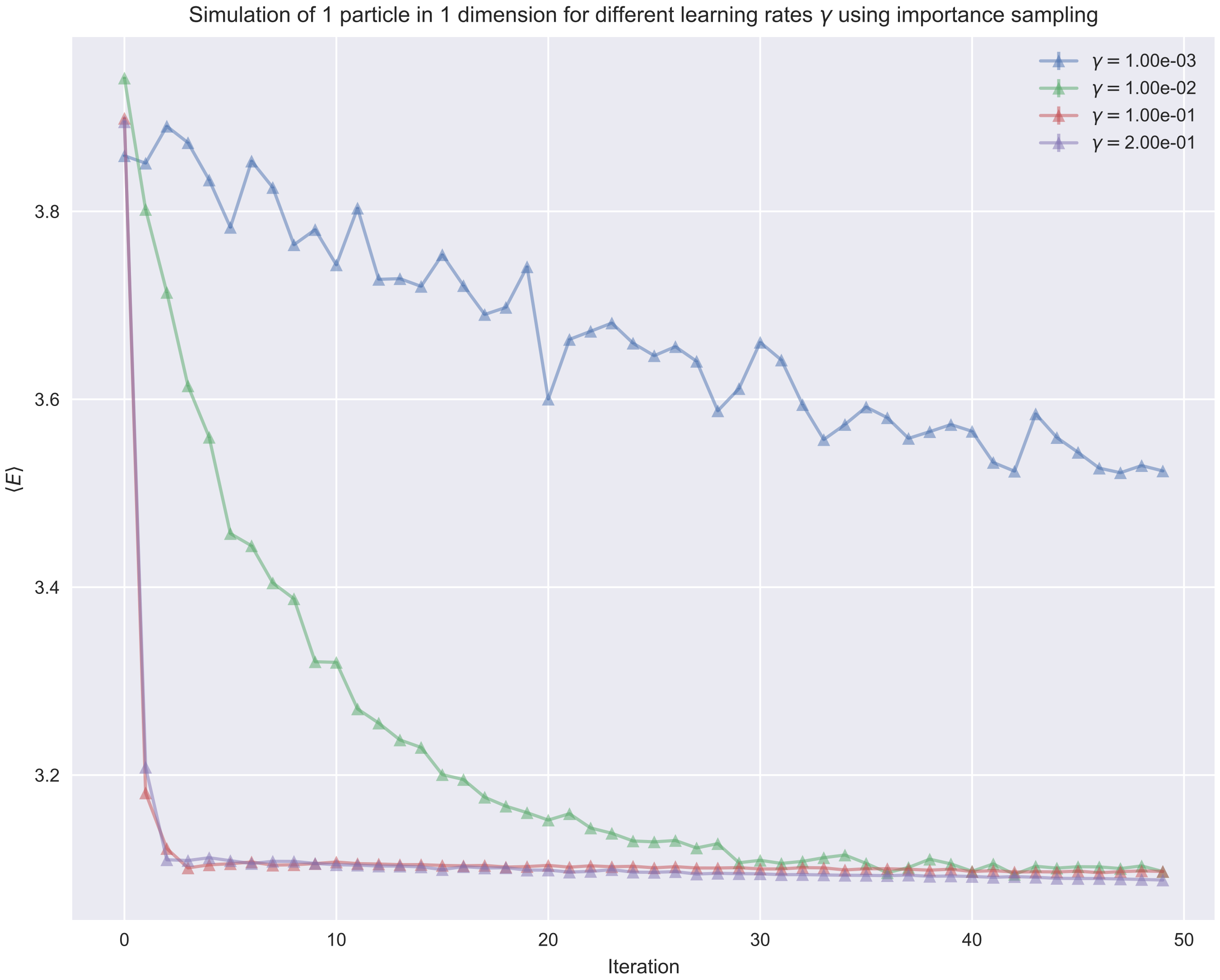

Figure 3: Plot of simulation with 2 particles in 3 dimensions with varied learning rates γ, using brute force metropolis sampling. For detailed values on energy, variance, and CPU-time per mc-cycle, see table 2

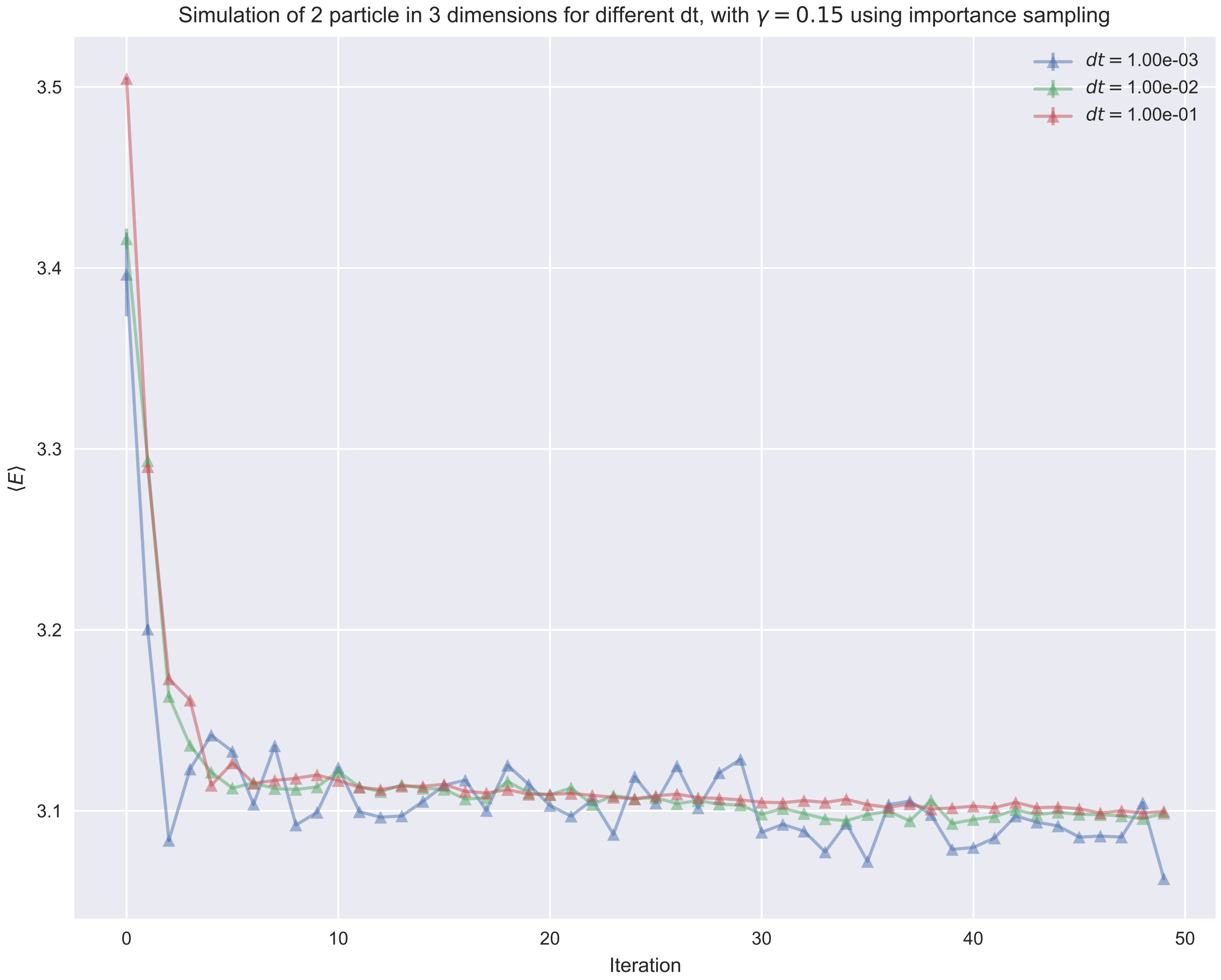

Figure 4: Plot of simulation with 2 particles in 3 dimensions with varied learning rates γ, using brute force metropolis sampling. For detailed values on energy, variance, and CPU-time per mc-cycle, see table 3

B Figures

B.1 1f)

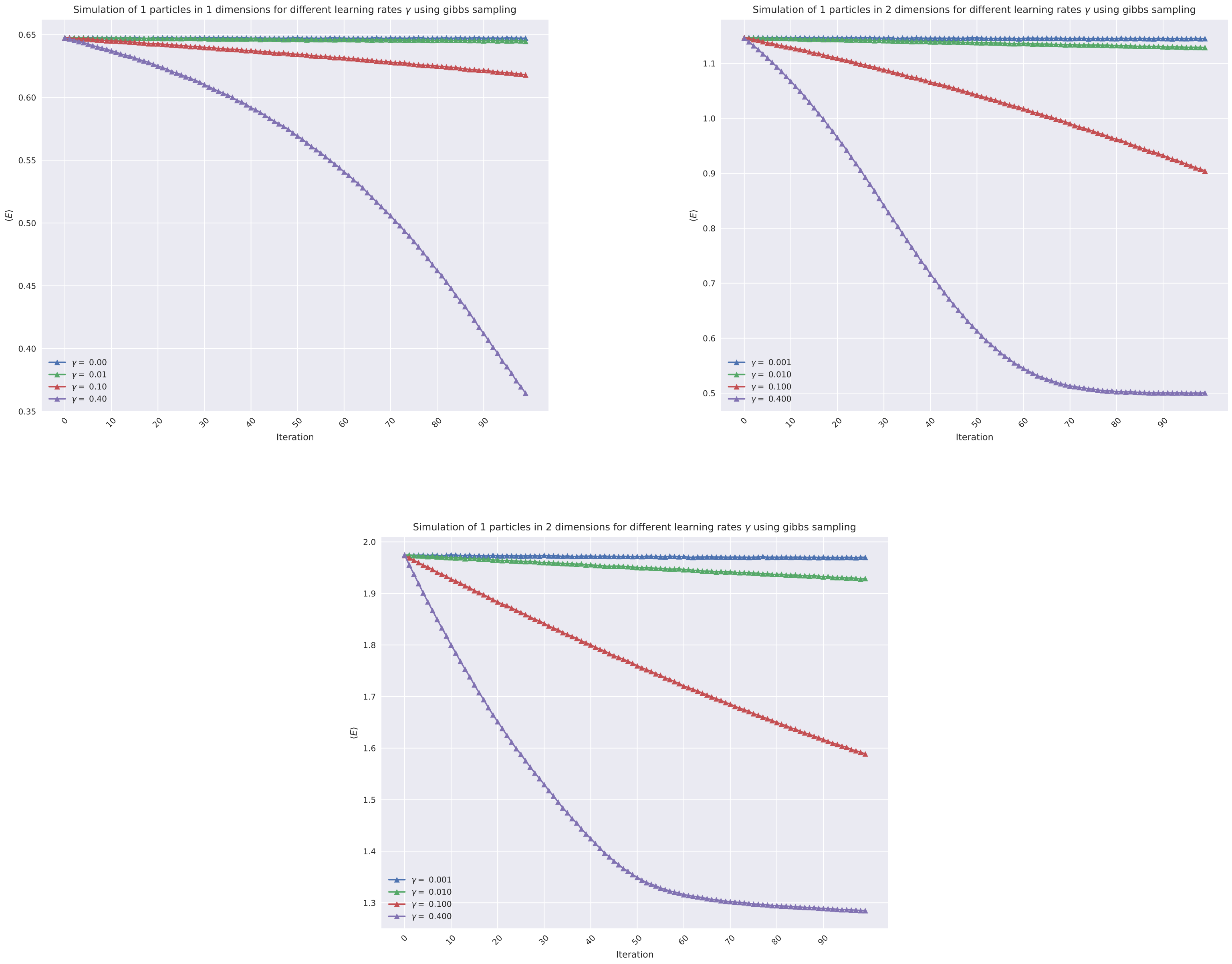

Figure 5: Plots for simulations with 1 particle and 1-3 dimensions using Gibbs sampling with different training rate values. Number of simulations for optimizing parameters are showed in x-axis.(The lowest figure is for 1 particle in 3 dimensions, wrong title.)

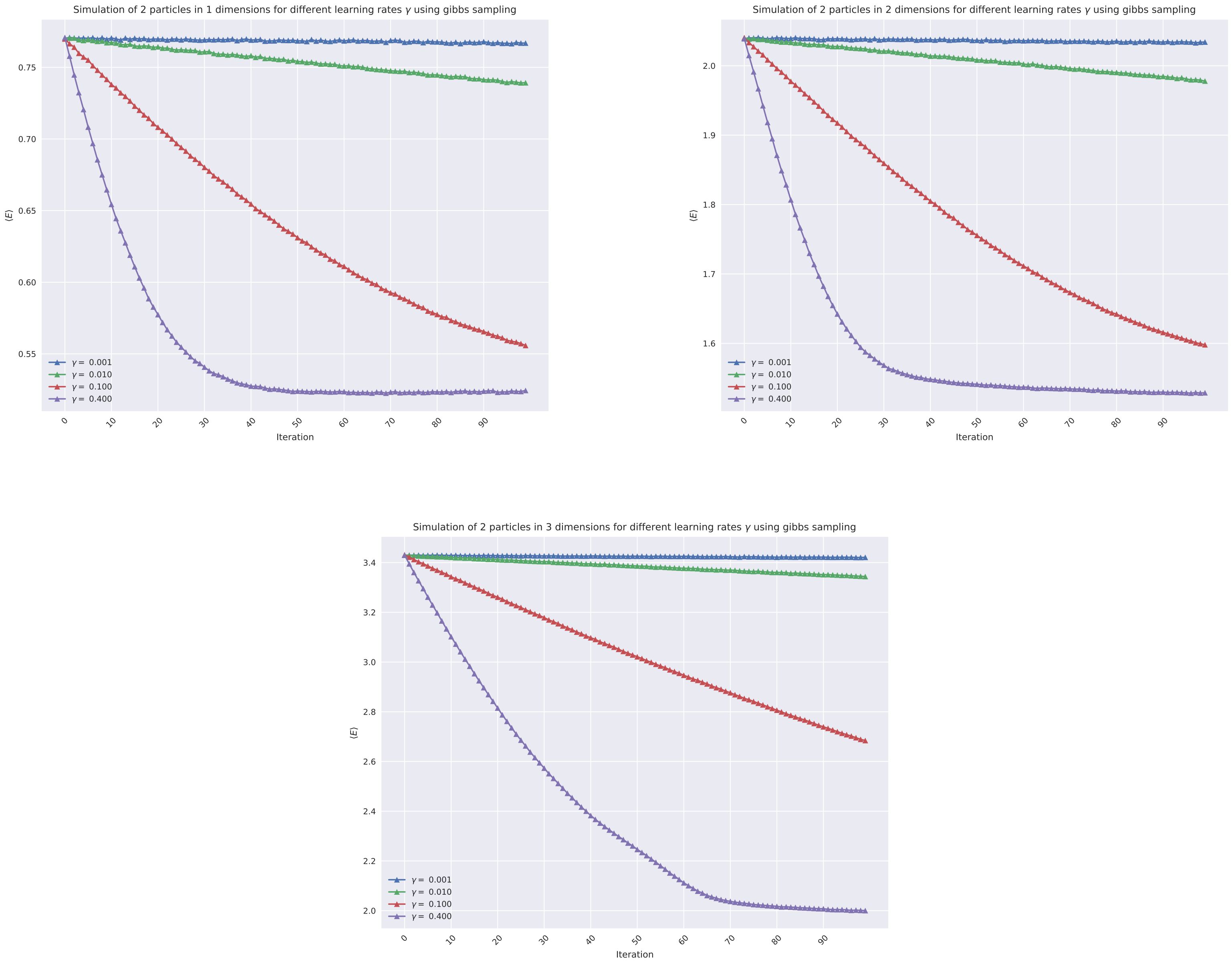

Figure 6: Plots for simulations with 2 particle and 1-3 dimensions using Gibbs sampling with different training rate values. Number of simulations for optimizing parameters are showed in x-axis.

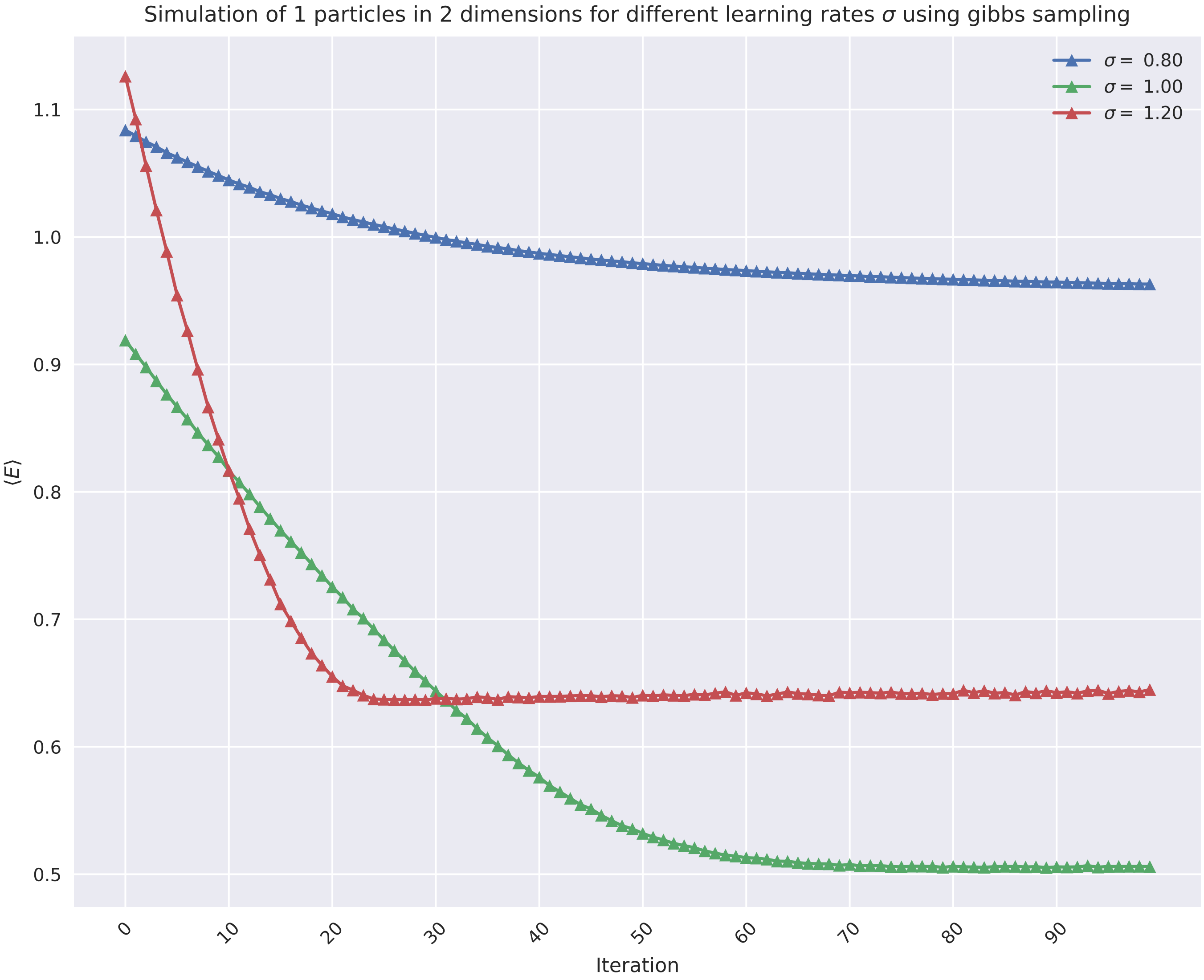

Figure 7: Plot of simulation with 1 particle in 2 dimensions using Gibbs sampling with different standard deviation σ.

C Local energy with $\psi(x) = F_{rbm}$

Given the Hamiltonian in eq. 1 the expression for the local energy is given by the following.

$$
\begin{aligned}
E_L &= \frac{1}{\Psi}\hat{H}\psi \\
&= \sum_k^P \frac{1}{2}\left(-\frac{1}{\Psi}\nabla_k^2\Psi + \omega^2 r_k^2\right) + \sum_{k<l}\frac{1}{r_{kl}}
\end{aligned}
$$

We wish to find the Laplacian of the NQS wave function. Since our visible nodes represents the particle positions in Cartesian coordinates the Laplace operator takes the second partial derivative with respect to each independent variable in the vector space. It can therefore be written as

$$\nabla_k^2\Psi = \sum_l^D \frac{\partial^2}{\partial x_{kl}^2}\Psi.$$

$$E_L = \sum_k^P \sum_l^D \frac{1}{2}\left(-\frac{1}{\Psi}\frac{\partial^2}{\partial x_{kl}^2}\Psi + \omega^2 r_k^2\right) + \sum_{k<l}\frac{1}{r_{kl}}$$

$$E_L = \sum_i^M \frac{1}{2}\left(-\frac{1}{\Psi}\frac{\partial^2}{\partial X_i^2}\Psi + \omega^2 X_i^2\right) + \sum_{k<l}\frac{1}{r_{kl}}$$

In the above equation we have used the fact that the number of visible nodes are $M = P \cdot D$. This reduces the number of sums in the first term to one. Also we see that it would be an advantage to solve the derivatives of $\ln\Psi$ since our wave function consist of sums in the exponential and also removes the Ψ in the denominator. Thus the identity below is helpful.

$$\frac{1}{\Psi}\frac{\partial^2}{\partial X_i^2}\Psi = \left(\frac{\partial}{\partial X_i}\ln\Psi\right)^2 + \frac{\partial^2}{\partial X_i^2}\ln\Psi$$

$$E_L = \sum_i^M \frac{1}{2}\left(-\left(\frac{\partial}{\partial X_i}\ln\Psi\right)^2 + \frac{\partial^2}{\partial X_i^2}\ln\Psi + \omega^2 X_i^2\right) + \sum_{k<l}\frac{1}{r_{kl}}$$

Our local energy can be rewritten and we must now solve both the first and second derivative of the wave function. Our wave function is represented by the marginal probability F_{rbm}.

$$
\begin{aligned}
\Psi(X) &= F_{rbm}(X) \\
&= \frac{1}{Z}\exp\left(-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2}\right)\prod_j^N\left(1 + \exp\left(b_j + \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\right)\right)
\end{aligned}
$$

$$\ln \Psi = -\ln Z - \sum_{i}^{M} \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_{j}^{N} \ln \left(1 + \exp\left(b_j + \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right) \right)$$

Since Z is the normalization constant its derivative is zero and can be removed. We end up with the following

$$\begin{aligned}
\nabla_i \ln \Psi &= \nabla_i \left(-\sum_{i}^{M} \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_{j}^{N} \ln \left(1 + \exp\left(b_j + \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right) \right) \right) \\
&= \frac{\partial}{\partial X_i} \left(-\sum_{i}^{M} \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_{j}^{N} \ln \left(1 + \exp\left(b_j + \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right) \right) \right) \\
&= \frac{-2(X_i - a_i)2\sigma^2}{(2\sigma^2)^2} + \sum_{j}^{N} \frac{1}{1 + \exp\left(b_j + \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right)} \left(\frac{\omega_{ij}}{\sigma^2} \right) \exp\left(b_j + \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right) \\
&= \frac{-(X_i - a_i)}{\sigma^2} + \frac{1}{\sigma^2} \sum_{j}^{N} \frac{\omega_{ij}}{1 + \exp\left(-b_j - \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right)}
\end{aligned}$$

Given the first derivative we now find the second derivative. And we have solved the analytic case of the local energy.

$$\begin{aligned}
\nabla_i^2 \ln \Psi &= \nabla_i^2 \left(-\sum_{i}^{M} \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_{j}^{N} \ln \left(1 + \exp\left(b_j + \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right) \right) \right) \\
&= \frac{\partial}{\partial X_i} \left(\frac{-(X_i - a_i)}{\sigma^2} + \frac{1}{\sigma^2} \sum_{j}^{N} \frac{\omega_{ij}}{1 + \exp\left(-b_j - \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right)} \right) \\
&= -\frac{1}{\sigma^2} + \frac{1}{\sigma^2} \sum_{j}^{N} \frac{-\omega_{ij}}{\left(1 + \exp\left(-b_j - \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right) \right)^2} \left(\frac{-\omega_{ij}}{\sigma^2} \right) \exp\left(-b_j - \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right) \\
&= -\frac{1}{\sigma^2} + \frac{1}{\sigma^4} \sum_{j}^{N} \frac{\left(\omega_{ij}\right)^2}{\left(1 + \exp\left(-b_j - \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right) \right)^2} \exp\left(-b_j - \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right)
\end{aligned}$$

D Gradients with respect to RMB parameters, $\psi(x) = F_{rbm}$

To find the local energy minimum of the NQS wavefunction the gradient of the local energy with respect to the variational parameter $\alpha = \{\mathbf{a}, \mathbf{b}, \mathbf{W}\}$ needs to be computed. For each variational parameter $\alpha_i \in \alpha$ the gradient is defined as in equation 14

$$G_i = \frac{\partial \langle E_L \rangle}{\partial \alpha_i} = 2 \left(\langle E_L \frac{1}{\Psi} \frac{\partial \Psi}{\partial \alpha_i} \rangle - \langle E_L \rangle \langle \frac{1}{\Psi} \frac{\partial \Psi}{\partial \alpha_i} \rangle \right) \tag{14}$$

While the NQS is defined for the Gaussian Binary RBM as (REPLACE WITH REF TO EQ)

$$\Psi(\mathbf{X}) = \frac{1}{Z} \exp \left(- \sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2} \right) \prod_j^N \left(1 + \exp \left(b_j + \sum_i^M \frac{X_i w_{ij}^{\mathbf{T}}}{\sigma^2} \right) \right)$$

We now find it useful to use the identity

$$\frac{d}{dx} \ln f(x) = \frac{1}{f(x)} \frac{d}{dx} f(x) \tag{15}$$

Taking the logarithm of the NQS wavefunction then gives

$$\ln(\Psi(\mathbf{X})) = \ln \frac{1}{Z} - \sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_j^N \ln \left[1 + \exp \left(b_j + \sum_i^M \frac{X_i w_{ij}^{\mathbf{T}}}{\sigma^2} \right) \right] \tag{16}$$

The final result for the gradient of the components of α_i is then

$$\frac{\partial \ln(\Psi(\mathbf{X}))}{\partial a_k} = \frac{1}{\sigma^2}(X_k - a_k) \tag{17}$$

$$\frac{\partial \ln(\Psi(\mathbf{X}))}{\partial b_n} = \left(1 + \exp \left(-b_n - \sum_i^M \frac{X_i w_{in}^{\mathbf{T}}}{\sigma^2} \right) \right)^{-1} \tag{18}$$

$$\frac{\partial \ln(\Psi(\mathbf{X}))}{\partial w_{kn}} = \frac{X_k}{\sigma^2} \left(1 + \exp \left(-b_n - \sum_i^M \frac{X_i w_{in}^{\mathbf{T}}}{\sigma^2} \right) \right)^{-1} \tag{19}$$

E Local energy with $\psi(x) = \sqrt{F_{rbm}}$

In the case of the Gibbs sampling we represent the wave function as

$$\begin{aligned}\Psi(X) &= \sqrt{F_{rbm}(X)}\\ &= \sqrt{\frac{1}{Z}\exp\left(-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2}\right)\prod_j^N \left(1 + \exp\left(b_j + \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\right)\right)}.\end{aligned}$$

Which is simply reduced to a constant in front of the term when taking the natural logarithm of the wave function.

$$\ln\Psi = -\frac{1}{2}\left(\ln Z - \sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_j^N \ln\left(1 + \exp\left(b_j + \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\right)\right)\right)$$

Now we can easily see that the derivatives of the logarithm of the nqs wave function becomes,

$$\begin{aligned}\nabla_i \ln\Psi &= \nabla_i\left(-\sum_i^M \frac{(X_i - a_i)^2}{4\sigma^2} + \frac{1}{2}\sum_j^N \ln\left(1 + \exp\left(b_j + \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\right)\right)\right)\\ &= \frac{-(X_i - a_i)}{2\sigma^2} + \frac{1}{2\sigma^2}\sum_j^N \frac{\omega_{ij}}{1 + \exp\left(-b_j - \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\right)},\end{aligned}$$

$$\begin{aligned}\nabla_i^2 \ln\Psi &= \nabla_i^2\left(-\sum_i^M \frac{(X_i - a_i)^2}{4\sigma^2} + \frac{1}{2}\sum_j^N \ln\left(1 + \exp\left(b_j + \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\right)\right)\right)\\ &= -\frac{1}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_j^N \frac{\left(\omega_{ij}\right)^2}{\left(1 + \exp\left(-b_j - \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\right)\right)^2}\exp\left(-b_j - \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\right)\end{aligned}$$

and are the derivative to be replaced in the expression of the local energy when we employ the Gibbs sampling.

F Gradients with respect to RMB parameters, $\psi(x) = \sqrt{F_{rbm}}$

We also need to update the gradients to be used in the gradient descent method. These are also easily found, a simple constant i front of each term.

$$\frac{\partial \ln(\Psi(\mathbf{X}))}{\partial a_k} = \frac{1}{2\sigma^2}(X_k - a_k) \tag{20}$$

$$\frac{\partial \ln(\Psi(\mathbf{X}))}{\partial b_n} = \frac{1}{2}\left(1 + \exp\left(-b_n - \sum_i^M \frac{X_i w_{in}}{\sigma^2}\right)\right)^{-1} \tag{21}$$

$$\frac{\partial \ln(\Psi(\mathbf{X}))}{\partial w_{kn}} = \frac{X_k}{2\sigma^2}\left(1 + \exp\left(-b_n - \sum_i^M \frac{X_i w_{in}}{\sigma^2}\right)\right)^{-1} \tag{22}$$

References

[1] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC, 1st edition, 2009.

[2] M Nilsen, Jon K Hjorth-Jensen. Data blocking. `http://www.uio.no/studier/emner/matnat/fys/FYS4410/v08/undervisningsmateriale/Material%20for%20Part%20I%20by%20Morten%20HJ/Slides%20from%20Lectures/blocking.pdf`, 2008.