VMC: Optimization trajectories for Restricted Boltzmann Machines

Johan Nereng

Department of Physics, University of Oslo, Norway

Spring, 2020

Abstract

In this project, which is an extension paper to of a previous project paper on Variational Monte Carlo (VMC) methods, training a restricted Boltzmann machine (RBM), with N hidden nodes, and M visible nodes, using VMC methods is the main focus. Two Metropolis Hastings methods, the brute force approach (BF) and Importance Sampling (IS) are tested and compared to Gibbs Sampling (GS). The methods are evaluated using the estimated energy $\bar{E}$ for both a system of one electron in a one-dimensional harmonic oscillator (HO) and an interacting system of two electrons in a two-dimensional HO. BF (using learning rate $\eta = 0.38$) and IS ($\eta = 0.34$) is shown to results in similar optimization trajectories for the RBM; a short initial phase (~ 10 iterations) of rapid optimization, followed by (~ 50 to ~ 100 training iterations for $N = 2$ and $N = 4$ respectively) of a slow and close-to linear optimization trajectory (regression coefficient $\sim 10^{-3}$ on iterations 10 to 20). Lastly, a phase of exponentially decaying training rates, which converged both for $N = 2$ and $N = 4$, was observed (up to 200 iterations). RBM optimization using Gibbs Sampling (GS) was conclusively shown to follow different optimization trajectories than BF and IS. GS ($\eta = 0.26$) only exhibited two different training behaviors; rapid initial training (< 10 iterations), and a phase of close to no optimization with constant $\bar{E}$ (regression coefficient $\sim 10^{-4}$, persistently up to 200 training iterations). The value of the standard deviation of the position distribution from GS, σ, was shown to directly control $\bar{E}$ in the "no training-phase" of GS.

Author's comments: *This spring I experienced an unprecedented loss of motivation and enthusiasm for my education after being ill for a few weeks. Happily, I found that this project thoroughly reminding me of why I study computational physics, as I found it very interesting to examine the training RBMS. In contrast to my previous projects in this course and similar courses, I did not focus very much at all on theory. Having understood what I judged to be a sufficient, I spent most of my time writing the C++ implementation and examining results. As I have previously spent a lot of time on the theory sections, I found it refreshing, challenging and fun to spend more time evaluating the simulations. As such, I know that my theory section is rather thin on details, but I hope that I have already proved that I am able to understand and describe theory well enough to pass muster. I would like to thank both Morten and Øyvind for being flexible with the hand-in of this project, as I am uncertain whether or not I would have passed my other exams if I could not have worked on this project through the last weeks of June.*

1 Introduction

This is an extention paper to my previous project paper *VMC: Effects of Importance Sampling and Jastrow factor on error estimates* [7], which is not published and not in review (not intended for publishing). As such, this extention paper will not include descriptions of theory already covered in [7], and should be read in context with said project paper.

This project introduces a new system, namely electrons in a harmonic oscillator, as such, the Hamiltonian of the system is changed from that in [7]. As the state of the system is represented by a neural network quantum state(NQS), the trial wave function is also new. These two alterations result in a different expression for the local energy, and parameter gradients. In this paper, I also introduce a new VMC method; Gibbs Sampling, which generate system states from a distribution. First and foremost however, this paper examines the training of a neural network, more specifically a restricted Boltzmann Machine (RBM), using VMC methods.

I start by introducing the system of electrons in an isotropic harmonic oscillator. Here I present the Hamiltonian of the system, as well as the exact energy values of the systems that are later used in simulation experiements. Then, the RBM is described briefly, and the transition to an NQS is stated, as well as the associated expressions for the local energy and drift force. I then go on to repeating the most crucial information from [7] on optimizing parameters, and derive the relevant expressions used in this project. The Gibbs Sampling scheme is then outlined, before presenting the algorithm used to train the RBM. I then restate (again, in context with [7]), my approach to the error analysis using the "automated blocking" method [5]. Lastly, I present simulation results for a system of a single particle in a one-dimensional harmonic oscillator, and then a system of two interacting electrons in a two-dimensional HO. The main focus when interpreting these results is the optimization of the RBM parameters, it's weights and biases.

In order to write this project paper and the code required to produce the results, I used a variety of tools, including: C++, Python 3.7.7, NumPy [8], as well as a number of books, web-pages and articles - of which most are listed under references. I would also like to emphasize that I used the C++ "blocking method" implementation developed by M.Jonsson (see [5]). All the code required to reproduce the results may be found on my github page .

2 Material and methods

2.1 System: Electrons in 2D isotropic HO

The system consists of P electrons in a D dimensional isotropic harmoinc oscillator (HO) potential, with the following idealized total Hamiltonian, when using natural units, ($\hbar = c = e = m_e = 1$), and energies in atomic units a.u:

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

Where ω is the oscillator frequency.

$$H_0 = \sum_{i=1}^{P} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right),$$

is the standard HO part of the Hamiltonian, while

$$H_1 = \sum_{i<j} \frac{1}{r_{ij}},$$

is the interactive part, where $r_{ij} = |\boldsymbol{r}_1 - \boldsymbol{r}_2|$, and $r_i = \sqrt{r_{i_x}^2 + r_{i_y}^2}$.

The Pauli exlusion principle requires that that two identical fermions, such as electrons, cannot occupy the same state [4]. As this project concerns itself with the ground state of the system of electrons (spin $\pm 1/2$), a maximum of two electrons are allowed, and these electrons must have opposite spins.

Analytic solutions for the system are available, which makes evaluating the performance of the algorithms possible under certain caveats. The energy of an harmonic oscillator in one dimension is [4];

$$E_n = \hbar\omega(n + \frac{1}{2}) \tag{2}$$

and with a.u and $\omega = 1.0$;

$$E_n = n + \frac{1}{2} \tag{3}$$

where n is the quantum number. Which means that in the ground state, $E_0 = \frac{1}{2}$ a.u. for one electron in a one dimensional HO.

For an interacting two-electron system in a two dimensional HO, $E_0 = 3$ a.u. [9]

2.2 Neural network quantum state

Instead of seeking to find suitable trial wave function ansatz for the system, this project represents the state of the system by a neural network quantum state (NQS), as was done in by Carleo and Troyer [1]. In their article, they suggest that one may view the wave function as a computational black box, which for a specific system configuration, outputs an amplitude corresponding to the value of the wave function.

As was done in [1], I've chosen to use a Restricted Boltzmann machine [6][p.983] (RBM) when representing the state of the system. RBM models latent variables, or variables that are not directly observed, but rather inferred from other quantities - in this case, the local energy. The network consists of a layer of M so called *visible nodes*, and a layer of N *hidden nodes*. These layers have associated weights and biases, with a two-way feed forward. This enables the network both to output values for the hidden nodes as a function of the visible nodes, and vice versa. This means that an RBM is a generative model, in this case a model which can generate particle positions from a distribution. This distribution is a function of the weights and biases, as well as the hidden nodes.

The marginal probability of the network is given by

$$F_{rbm}(\boldsymbol{X}) = \frac{1}{Z} \sum_{h} e^{-E(\boldsymbol{X},\boldsymbol{h})},$$

and corresponds to the probability of the system state, $\boldsymbol{X}$. As $\boldsymbol{X}$ in this case is a vector of continuous particle positions, the energy function, $E(\boldsymbol{X}, \boldsymbol{h})$ of the network (not to be confused with the energy of the system), must be defined accordingly. There are various forms of RBMs with different energy functions, specialized to different types of data. The one used here, which enables continous $\boldsymbol{X}$, is known as a *Gaussian-Binary RBM* [6][p.986], and also includes binary hidden nodes $\boldsymbol{H} = \{h_j\}$ for $j = 1, 2, .., N$, and $h_j \in \{0, 1\}$. Using it's associated energy function, and representing each particle position as X_i, such that $i = 1, 2, .., M$ where $M = PD$, results in the following NQS/wave form representation [3];

$$\begin{aligned}\Psi(\mathbf{X}) &= F_{rbm}(\mathbf{X}) \\ &= \frac{1}{Z}\sum_{\{h_j\}} e^{-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_j^N b_j h_j + \sum_{i,j}^{M,N} \frac{X_i w_{ij} h_j}{\sigma^2}} \\ &= \frac{1}{Z} e^{-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2}} \prod_j^N (1 + e^{v(j)})\end{aligned} \tag{4}$$

$$v(j) = b_j + \sum_{i=1}^{M} \frac{X_i W_{ij}}{\sigma^2} \tag{5}$$

Since all evaluations of the wave function is in the context of finding a ratio of probabilities; $\frac{\Psi_{rbm}^{new}}{\Psi_{rbm}^{old}}$, and all other quantities are derivatives of $\ln \Psi_{rbm}$, the normalization constant Z is not relevant.

2.2.1 Local Energy and drift force

As described in project 1 [7][2.2], the results obtained through the VMC methods used in this project relies on the variational principle. As such, the local energy

$$E_L(\mathbf{r}) = \frac{1}{\Psi_T(\mathbf{r})} H \Psi_T(\mathbf{r}). \tag{6}$$

is required in order to carry out the Monte Carlo integration for the ground state energy. Using the trial wave function (2.2), and (6) an analytic expression for the local energy may be fond. This is derived in detail in Appendix 1: Analytic expression for the local energy, and will not be discussed further here. Based on the same appendix, the correponding drift force, needed for the importance sampling [7][2.2.2], is;

$$F_i = \frac{2\nabla\Psi_T}{\Psi_T} = 2\left[-\frac{(X_k - a_k)}{\sigma^2} + \sum_i^N \frac{w_{kj}}{\sigma^2}\frac{1}{1 + e^{-v(\boldsymbol{X}, j)}}\right] \tag{7}$$

2.3 Optimizing parameters

As the weights of the RBM are initialized randomly, the resulting energy is unlikely be the best approximation the algorithm is capable of. In [7], the aim was to find α s.t the energy was minimized. Here, the aim is instead to optimize the network. This is done by optimizing the weights $\boldsymbol{X}$, and biases $\boldsymbol{a}$ and $\boldsymbol{b}$, using the energy as a cost function. Although this optimization is over multiple parameters of different types, the strategy for each parameter β_i of parameter type β, which can either be $\boldsymbol{a}$, $\boldsymbol{b}$, or $\boldsymbol{X}$, is the same as for α described in more detail in [7][2.2.3]. The update scheme for parameter β is;

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k - \eta_k \boldsymbol{g}(\boldsymbol{\beta}_k), \tag{8}$$

where η is the learning rate. (8) requires the gradient of the local energy w.r.t the parameters in $\boldsymbol{\beta}$, $\boldsymbol{g}(\boldsymbol{\beta}) = \nabla_\beta \langle E_L \rangle$. This is given by

$$\nabla_\beta \langle E_L \rangle = 2 \left(\langle \frac{\bar{\Psi}_\beta}{\Psi[\beta]} E_L[\beta] \rangle - \langle \frac{\bar{\Psi}_\beta}{\Psi[\beta]} \rangle \langle E_L[\beta] \rangle \right),$$

where $\frac{\bar{\Psi}_\beta}{\Psi[\beta]} = \frac{1}{\Psi_{rbm,\beta}} \frac{\partial \Psi_{rbm,\beta}}{\partial \beta}$. Expressions for $\beta = a, b, W$ can be found in Appendix 2: Derivatives w.r.t RBM parameters.

2.4 Gibbs Sampling

Gibbs sampling [6][ch.24.2] is a Markov Chain Monte Carlo (MCMC) algorithm, analogous to coordinate descent in an MCMC framework. In this implementation, the two-way feed forward network, the RBM, is used to sample the joint probability of $\boldsymbol{X}$ and bm $\boldsymbol{H}$ in a two step process. First, $P(\boldsymbol{H}|\boldsymbol{X})$ is used to determine the state of the binary nodes H, for the current system state $\boldsymbol{X}$. Then, after having updated $\boldsymbol{H}$, $P(\boldsymbol{X}|\boldsymbol{H})$ is used to sample a new system state. In contrast to the Metropolis Hastings algorithm, the probability of accepting this proposed system state equals one [2], ie. every proposal is accepted. The conditional probabilities are given by;

$$\begin{aligned} P(H_j = 1|\boldsymbol{X}) &= \frac{1}{1 + e^{-b_j - \sum_i^M \frac{X_i W_{ij}}{\sigma^2}}} = \text{logistic}(-(v(j)) \\ P(H_j = 0|\boldsymbol{X}) &= \text{logistic}((v(j)) \end{aligned} \tag{9}$$

and

$$P(X_i|\boldsymbol{H}) = \mathcal{N}(X_i; a_i + \boldsymbol{W}_{i*}\boldsymbol{H}, \sigma^2) \tag{10}$$

In order to sample $\Psi_{T,Gibbs}(\boldsymbol{X})$ s.t $|\Psi_{T,Gibbs}(\boldsymbol{X})|^2$ can be interpreted as a probability, $\Psi_{T,Gibbs}(\boldsymbol{X}) = \sqrt{F_{rbm}(\boldsymbol{X})}$, under the assumption that the wave function is positive definite. This slightly changes the expressions the local energy and parameter gradients, as is described briefly in Appendix 3: Local energy and parameter gradients for Gibbs sampling.

Having an acceptance rate of 1 means that the Gibbs algorithm should have a short burn-in phase (see the Implementation sub chapter). Given sufficient training, I also expect this method to have a smaller estimation error.

2.5 Implementation

I have chosen to structure my code s.t. a single experiment uses a single RBM object. This RBM object owns a set of weights and biases, which it updates after each Monte Carlo simulation. In each Monte Carlo simulation, I construct a system object, which holds information about the system, and which calls on a sampler object that samples the energy as well as relevant quantities for calculating the gradient of the parameters.

In each Monte Carlo simulation, I intend to discard all samples in an initial *burn-in phase* [6][p.856], or equilibrium phase, [6][p.856], as is common practice when using VMC methods. The effective length of this burn-in phase will be decided in the initial tests of the system, and is intended to suit the simulation length as well as other practicalities, such as the error analysis.

For the two Metropolis Hastings algorithm, the brute force approach and the Importance Sampling approach, I use the algorithm in [7][2.2.1], with the local energy expression derived in appendix 1.

This algorithm explicitly covers the brute force approach, but the changes needed to use the algorithm for Importance Sampling is explained in [7][2.2.2]. As the RBM does not inherently distinguish between particles, I chose to treat each position coordinate as a visible node. In each update step of the Metropolis Hastings methods, I therefor only change the value of a single visible node, instead of moving an entire particle. In the case of Gibbs sampling, all particle positions are sampled at each MC cycle. The energy contribution from the interacting term of the Hamiltonian is the only quantity which directly distinguishes between particles, and is therefor implemented in a way that ensures this.

2.5.1 Algorithm

The following algorithm gives an overview of the operations in a single experiment;

```
Algorithm: VMC using RBM with parameter optimization

  1. Initialize algorithm
        - Set number of iterations, $G$
        - Set the number of Monte Carlo cycles, $MC$, and "burn in fraction".
        - Set VMC method
          - Set expression for $E_L$ and $\frac{\bar{\Psi}_\beta}{\Psi[\beta]}$ according to choice of method
          - If using a variant of the Metropolis Algorithm, set step length, $l$
        - Initialize RBM parameters $\boldsymbol{a} = \mathcal{U}(0, 1/M), \boldsymbol{b} = \mathcal{U}(0, 1/N), \boldsymbol{W} = \mathcal{U}(0, 1/(M \times N)$
  2. Execute Monte Carlo scheme
        - Initialize visible nodes, $\boldsymbol{X} = \mathcal{U}(-1, 1)$, set $E = 0$
        - For $cycle = 1, 2, .., MC$
          - Propose new configuration according to VMC method
          - Evaluate proposal, reject or accept.
          - If $cycle \geq MC\times$ "burn in fraction", calculate $E_L$ using current state.
  3. Calculate $\langle H \rangle = \frac{E}{M}$
  4. Optimize parameters $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{W}$
  5. If the number of iterations of step 2 is less than $G$, go to step 2. Else, end simulation
```

2.6 Error analysis

In [7] I found that the error estimate, σ, on correlated data without methods such as "blocking" is large deceptive. Hence, I have chosen to exclusively use the standard error (error estimate using "blocking"), $\hat{\sigma}$ in this project. This means that the error analysis of this extension paper is similar to the error analysis of project paper one. For the sake of clarity, I would like to stress that when using $\hat{\sigma}$ I am referring to the standard error of the energy estimate $\bar{E}$, and when I am using σ is am referring to standard deviation of the probability distribution of the RBM.

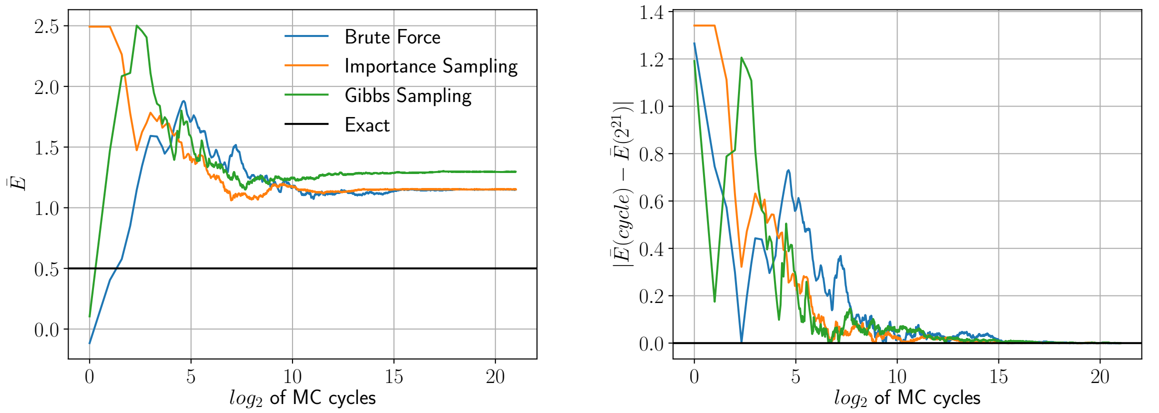
3 Results

In the following simulations I will seek to verify the integrity of algorithm(s) in the C++ implementation. I will then study and compare the two Metropolis Hastings methods (the brute force approach and importance sampling), and Gibbs sampling, when used with an RBM. This will be done using a single particle in a single dimension, $P = D = 1$, $N = 2$. Subsequently, I will examine the simulations results when using two interacting electrons in two dimensions, $P = D = 2$, $N = 2$ and $N = 4$.

3.1 Initial testing of VMC methods

In order to ascertain whether or not the implementation of the various VMC methods were successful, I ran an MC simulation over 2^{21} cycles $P = 1$, $D = 1$, $N = 2$, using Brute Force ($l = 1.0$), Importance Sampling ($\Delta t = 1.0$), and Gibbs Sampling.

Figure 1 shows the estimated energy $\bar{E}$ (left), and the absolute difference between the estimated energy at each cycle $\bar{E}(cycle)s$ and final estimated energy $\bar{E}$ (right), using the same seed for all three simulations. $\bar{E}$ is, as may be expected from an untrained network, relatively far from the exact energy. It is however clear that all three methods find a more-or-less stable value for $\bar{E}$ after a burn-in phase. Similarly stable $\bar{E}$ values after burn-in was observed in subsequent tests using different seeds. As such, the VMC methods are deemed successfully implemented.



(a) Energy estimate

(b) Difference in energy estimate from final estimate

Figure 1: Initial test of the three VMC methods (same seed)

3.2 Standard simulation settings

Based on trial and error when initially testing the networks, I came to the conclusion that 2^{21} MC cycles give sufficient convergence to illuminate the qualities of the model. As such, I will use 2^{21} cycles unless otherwise stated.

As the blocking method requires data of a length which is a power of 2, I will henceforth discard all samples taken in the first half of the MC cycles (as I run cycles in powers of 2). This should (more than) ensure that samples in the burn-in phase is discarded, as well as provide a suitable data format for the automated blocking method.

Having established that the three VMC methods are capable of achieving energy convergence, I did a survey of the effects of l (BF) and Δt (IS). I averaged 12 experiments, each using 7 optimization runs, and evaluating the results of the 8'th run by $\hat{\sigma}$. From the results presented in Appendix 4: Testing step length l and Δt, in figures 5 and 6, I conclude that $l = 1.0$ (BF) and $\Delta t = 2.0$ are suitable parameters, and will use those values in subsequent experiments, unless otherwise stated.

3.3 Comparison of Metropolis Hastings algorithms in training and RBM, n=2

Before directly comparing the methods, I wanted to find a suitable range of values for η to test the methods on. As such, I used 10 values of $\eta = [0.1, 1.0]$. Figure 7 and 8 in Appendix 4: Testing learning rate η, show the results after the least suitable values of η were discarded IOT clean up the figures. Cross checking the two plots, I found that $\eta = [0.3, 0.4]$ was the only two adjacent values (with $\Delta = 0.1$) which were both within 10^{-3} of $\bar{E}$ for both methods. The figures also show that for $\eta = [0.3, 0.4]$, $\bar{E} - 0.5 < 10^{-4}$ for all training iterations ≥ 10, indicating a minima of the cost function, which I judge to mean that the RBM is "sufficiently trained".

3.3.1 Focused examination of training RBMs using Metropolis Hastings algorithms

Next, I wanted to simultaneously compare BF and IS, and zero in on the best suited learning rate η. As $\hat{\sigma}$, measures the variance of the samples adjusted for correlation, ie. variance of the underlying sample distribution, I also needed an evaluation metric that reflected how close $\bar{E}$ was to the analytic $E = 0.5$. For this, I chose to use the squared difference between estimated energy, $\bar{E}$ and $E = 0.5$, $(\Delta E)^2$.

I averaged $\hat{\sigma}$ and $(\Delta E)^2$ in each experiment for each η after 10 training iterations, over the next 5 training iterations. Smaller values of mean $(\Delta E)^2$ indicate that the loss function is sufficiently minimized, while smaller values of $\hat{\sigma}$ indicate that the network is estimating consistently. As such, the best suited η should produce the smallest $(\Delta E)^2$, and a sufficiently small $\hat{\sigma}$. In order to average out noise, I repeated the experiments 15 times using different seeds and averaged the results, which are shown in table 1. I included experiments for IS using $\Delta t = 1.0$ so that I could validate my previous findings on the choice of time step.

η	BF		IS $\Delta t = 1.0$		IS $\Delta t = 2.0$	
	Mean $(\Delta E)^2$	Mean $\hat{\sigma}$	Mean $(\Delta E)^2$	Mean $\hat{\sigma}$	Mean $(\Delta E)^2$	Mean $\hat{\sigma}$
0.26	1.04×10^{-4}	2.66×10^{-8}	4.92×10^{-5}	2.09×10^{-8}	8.33×10^{-5}	2.62×10^{-8}
0.28	1.08×10^{-4}	4.33×10^{-8}	5.09×10^{-5}	4.43×10^{-8}	8.62×10^{-5}	4.79×10^{-8}
0.30	1.05×10^{-4}	5.32×10^{-8}	4.96×10^{-5}	5.78×10^{-8}	8.50×10^{-5}	6.07×10^{-8}
0.32	1.01×10^{-4}	6.00×10^{-8}	4.80×10^{-5}	7.90×10^{-8}	8.12×10^{-5}	7.80×10^{-8}
0.34	1.09×10^{-4}	6.85×10^{-8}	5.16×10^{-5}	6.79×10^{-8}	8.70×10^{-5}	7.11×10^{-8}
0.36	5.43×10^{-5}	4.58×10^{-9}	2.60×10^{-5}	4.10×10^{-9}	4.42×10^{-5}	4.81×10^{-9}
0.38	8.70×10^{-5}	1.69×10^{-8}	4.14×10^{-5}	1.78×10^{-8}	7.09×10^{-5}	1.66×10^{-8}
0.40	8.52×10^{-5}	7.28×10^{-9}	4.05×10^{-5}	6.45×10^{-9}	6.94×10^{-5}	7.00×10^{-9}
0.42	1.11×10^{-4}	4.05×10^{-8}	5.26×10^{-5}	3.39×10^{-8}	8.92×10^{-5}	3.46×10^{-8}
0.44	1.20×10^{-4}	1.87×10^{-7}	5.63×10^{-5}	1.48×10^{-7}	9.54×10^{-5}	1.47×10^{-7}

Table 1: Comparison of methods: squared difference between estimated energy, $\bar{E}$ and $E = 0.5$, $(\Delta E)^2$, and standard error using blocking, $\hat{\sigma}$, averaged over gradient descent iterations $10, 11, 12, 13, 14, 15$, which are again averaged over 15 experiments for different learning rates η.

Table 1 shows that for the Brute Force method, mean $(\Delta E)^2$ ranges from $\sim 10^{-4}$ to $\sim 10^{-5}$ for different values of η. $\eta = 0.36, 0.38, 0.40$ give the best average $(\Delta E)^2 \sim 10^{-5}$. Mean $\hat{\sigma}$ ranges from $\sim 10^{-8}$ to $\sim 10^{-9}$. For BF ($l = 1.0$), $\eta = 0.40$ appear to give both the lowest $(\Delta E)^2 (\sim 10^{-5})$ and the lowest $\hat{\sigma}$ ($\sim 10^{-9}$).

All values of $(\Delta E)^2$ is $\sim 10^{-5}$ for both choices of Δt when using IS, while $\hat{\sigma}$ is similar for all values of η across both BF and the two tested cases of IS. Comparing IS $\Delta t = 1.0$ and $\Delta t = 2.0$ clearly indicates that $\Delta t = 2.0$ is better suited, as all values of $(\Delta E)^2$ are twice as high in the $\Delta t = 1.0$ case, strengthening the validity of initially choosing $\Delta t = 2.0$.

For IS $\Delta t = 2.0$, $(\Delta E)^2$ is lowest for $\eta = 0.34$, albeit at $\hat{\sigma} \sim 10^{-8}$. Hence, I conclude that $\eta = 0.34$ is the best suited for IS (with $\Delta t = 2.0$).

As all values of $(\Delta E)^2$ was $\sim 10^{-5}$ for IS, compared to $\sim 10^{-4}$ to $\sim 10^{-5}$ depending on η for BF, I conclude that when using an RBM, BF is more sensitive than IS to the choice of learning rate w.r.t accuracy of energy estimates. I furthermore conclude that the effects of η on $\hat{\sigma}$ is the same for both BF and IS, regardless of the value of Δt. When using the optimal η values based on $(\Delta E)^2$ alone, BF and IS performed equally well, producing $(\Delta E)^2 \approx 8.7 \times 10^{-8}$ with $\hat{\sigma} = 7 \times 10^{-8}$, when averaged over 15 experiments.

3.4 Gibbs Sampling, n=2

As GS samples positions from a distribution ((10)), the standard deviation of that distribution, σ, directly influences the state of the system, and hence also the energy. In the initial test of VMC methods shown in figure 1, it is clear that the two Metropolis Hastings methods converged to a different $\bar{E}$ than the Gibbs Sampler (GS). This offset in convergence for GS is likely the result of it's direct dependence on σ. As such, I ran 8 experiments ($\eta = 0.3$) with σ ranging from 0.65 to 1.0, and plotted the results (see Appendix 4: Testing σ for GS) From figure **??**, it is clear that the value of σ influences $\bar{E}$, as expected. For $P = D = 1$, $n = 2$ it appears that the correct value of

σ lies somewhere between 0.9 and 0.95. In order to narrow down the value of σ further, I ran 8 experiments using $\sigma = [0.9, 0.935]$.

Figure 1 shows $\bar{E}$ for $\sigma = [0.9, 0.935]$. It is clear that $\bar{E}$ converges after few iterations compared to BF and IS (see figures in Appendix 4). Where $\bar{E}$ from BF and IS fluctuate (although marginally after ~ 10 iterations) for all tested η values across all iterations, $\bar{E}$ reaches a stable value after ~ 8. This indicates that the RBM is either fully trained, or training very slow (relative to BF and IS), when using η of the same magnitude ~ 0.3. Judging by the trajectories of $\bar{E}$ for the different values of σ, it also appears that $\bar{E}$, after ~ 6 iterations, may be moved arbitrarily close to $E = 0.5$ by adjusting σ.

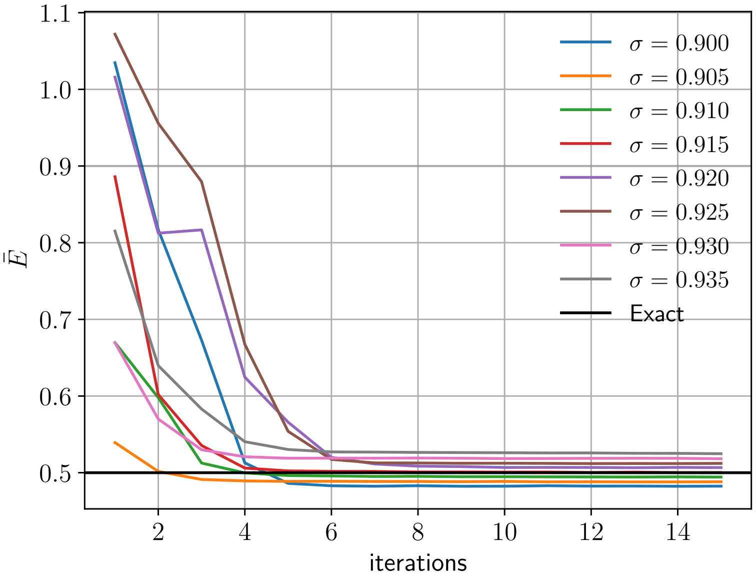


Figure 2: Gibbs Sampling: $\bar{E}$ for different values of σ

3.4.1 Focused examination of training RBMs using Gibbs Sampling

When producing the results presented in figure 1, $\eta = 0.3$ was used. This value was picked arbitrarily for testing the effects of σ. In order to obtain a better suited value of η, I ran an experiment for a wide range of η values. Based on the results (shown in figure 10 in Appendix 4) I chose to do a closer examination of η between 0.22 and 0.38 in increments of 0.2. I ran a simulation for each value, averaging $(\Delta E)^2$ and $\hat{\sigma}$ as in the focused examination of training RBMs using Metropolis Hastings algorithms, the results of which are in table 2.

Table 2 shows that $(\Delta E)^2$ between $\sim 10^{-5}$ and $\sim 10^{-6}$, σ is $\approx 1.4 \times 10^{-3}$, for all tested values of η. $\eta = 0.26$ has the lowest $(\Delta E)^2$ (8.41×10^{-6}), and is therefor deemed as the best suited value among the ones that were tested.

Comparing the best values of $(\Delta E)^2$ from BF($\approx 8.70 \times 10^{-5}$), IS ($\approx 8.62 \times 10^{-5}$), and GS ($\approx 8.41 \times 10^{-6}$), indicates that GS approximates E for $P = D = 1$ with $n = 2$. However, the associated $\hat{\sigma}$ was $\sim 10^{-3}$ for GS, compared to $\sim 10^{-8}$ for BF and IS. This means that in the case of GS, the energy samples fluctuate (comparatively) wildly, around mean which is (comparatively) close to the exact energy. BF and IS on the other hand exhibit a lot smaller fluctuations around a mean which is further from the exact energy. Because of this, $\bar{E}$ should be sampled for a longer time when using GS, than what is necessary when using BF or IS in the $P = D = 1$, $n = 2$ cse.

η	Mean $(\Delta E)^2$	Mean $\hat{\sigma}$
0.22	7.17×10^{-6}	1.40×10^{-3}
0.24	2.21×10^{-5}	1.41×10^{-3}
0.26	8.41×10^{-6}	1.40×10^{-3}
0.28	6.07×10^{-6}	1.40×10^{-3}
0.30	1.76×10^{-5}	1.41×10^{-3}
0.32	6.95×10^{-6}	1.40×10^{-3}
0.34	5.35×10^{-6}	1.40×10^{-3}
0.36	1.35×10^{-5}	1.41×10^{-3}

Table 2: Gibbs Sampling: squared difference between estimated energy, $\bar{E}$ and $E = 0.5$, $(\Delta E)^2$, and standard error using blocking, $\hat{\sigma}$, averaged over gradient descent iterations $10, 11, 12, 13, 14, 15$, which are again averaged over 15 experiments for different learning rates η.

3.5 Comparison of VMC methods: 2 interacting electrons

Having examined the three VMC methods for $P = D = 1$ and $n = 2$, I here move on to the $P = D = 2$ case with interaction between the two electrons.

I performed 10 experiments for each of the three methods, using values of $\eta, \sigma, l, \Delta t$ from table 3 (based on the results from the previously presented $P = D = 1$ simulations), for 20 optimization iterations, and averaged the results, for both $N = 2$ and $N = 4$.

Method	η	σ	Step length
BF	0.38	1.0	$l = 1.0$
IS	0.34	1.0	$\Delta t = 2.0$
GS	0.26	0.914	-

Table 3: $P = D = 2$, interacting: simulation settings

Figure 11 and figure 12 in Appendix 4: Training RBMs with interaction show average the $\bar{E}$ in the first 20 training iterations when using $N = 4$ and $N = 2$. Plots of the corresponding average $\hat{\sigma}$ values (figure 15 and figure 13) are in the same appendix. By-eye comparison of these plots indicate that both $N = 2$ and $N = 4$ result in similarly estimated energies ($\bar{E} \approx 3.4$ for BS and IS and $\bar{E} \approx 2.9$ for GS) for each VMC method after the first 20 iterations.

In order to gain more information about the training of RBMs from these simulations, I used linear regression on samples of $\bar{E}$ from the 10 experiments for each VMS method, and each N. Based on this, I predicted where $\bar{E}$ would intercept $E = 3.0$ (visualized in figures 16 and 17 for $N = 4$ in Appendix 4). As all methods had steep learning in the first few iterations, I discarded iterations 0 to 10 before fitting. Table 4 show the regression coefficients and predicted intercept with $E = 3.0$ for all methods using both $N = 2$ and $N = 4$.

Method	N	Regression coefficient	Predicted iteration for $\bar{E} = 3.0$
BF	2	-1×10^{-3}	247
IS	2	-1×10^{-3}	207
GS	2	1×10^{-4}	352
BF	4	-5×10^{-4}	462
IS	4	-7×10^{-4}	367
GS	4	9×10^{-5}	433

Table 4: Comparison of VMC methods: Linear Regression fit

Based on the linear regression results in table 4, $N = 2$ is predicted to intercept $E = 3.0$ intercepts after fewer iterations than $N = 4$ for all methods. It is also worth noting that GS is predicted to intercept $E = 3.0$ later than BF and IS, and that IS is predicted to intercept $E = 3.0$ first. These predictions however are only valid if the training trajectory of $\bar{E}$ is linear w.r.t. the iterations. In order to test whether this is the case or not, I carried out 6 more experiments, one for each VMC method for both $N = 2$ and $N = 4$. In these experiments, I used 200 training iterations, with the same simulations settings as in the previous experiments.

Figure 3 ($(\Delta E)^2$), and 18 ($\hat{\sigma}$) which is located in Appendix 4: Training RBMs with interaction show the results from the experiments that ran for 200 training iterations. Averages over iterations 180 to 200 are shown in table 5.

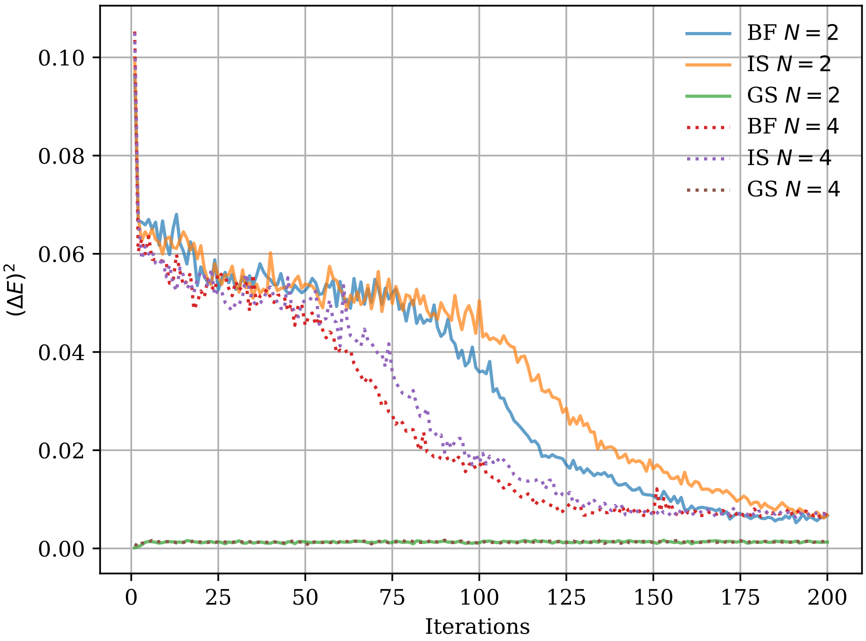


Figure 3: Comparison of methods: $(\Delta E)^2$ vs. training iterations, using $N = 2$ and $N = 4$ hidden nodes.

Figure 3 shows three distinctly different types of trajectories for $(\Delta E)^2$:

1. The RBMs using GS (both $N = 2$ and $N = 4$) quickly (< 10 training iterations) reaches a stable $(\Delta E)^2$ value $\approx 0.1 \times 10^{-3}$, which appears to be consistently throughout the next 190 iterations.

2. BF and IS for $N = 4$, has an initial reduction in $(\Delta E)^2$ (from $\approx 1.0 \times 10^{-1}$ to $\approx 0.5 \times 10^{-2}$), after $\sim 10^1$ iterations. From iterations ≈ 10 to ≈ 50, $(\Delta E)^2$ appears to linearly decrease with the training iterations. In the approximate region $[50, 125]$ iterations, $(\Delta E)^2$ drops at a higher rate per iteration, down to $\sim 10^{-2}$. From iteration ≈ 150, $(\Delta E)^2$ is reduced (at a low rate) from $\sim 10^{-2}$ to $\approx 0.7 \times 10^{-3}$ (see figure 4).

3. BF and IS for $N = 2$ has the same shape as for $N = 4$, but maintains $(\Delta E)^2 \approx 0.5 \times 10^{-2}$ for approximately twice as long as $N = 4$. The drop off from 0.5×10^{-2} to $\sim 10^{-3}$ also appears to take twice as long (from ≈ 100 to ≈ 200 iterations), where it reaches $\approx 0.7 \times 10^{-3}$, just as for $N = 4$.

Method	N	Mean $(\Delta E)^2$	Mean $\hat{\sigma}$
BF	2	6.19×10^{-3}	3.02×10^{-3}
IS	2	7.78×10^{-3}	3.34×10^{-3}
GS	2	1.34×10^{-3}	1.96×10^{-3}
BF	4	7.15×10^{-3}	2.81×10^{-3}
IS	4	7.11×10^{-3}	2.77×10^{-3}
GS	4	1.37×10^{-3}	1.94×10^{-3}

Table 5: Comparison of VMC methods: Average values over training iterations from 180 to 200

The $(\Delta E)^2$ trajectories of BF and IS (for both values of N), are clearly not linear, meaning that the predictions in table 3 are not correct. The RBM using BF or IS appear to have different "learning phases"; an initial and large correction of parameters, then a slow and more-or-less linear phase, followed by what may appear to be an exponential phase. The length of these phases appear to be dependent on the number of hidden nodes N, as $N = 2$ had a longer linear phase than $N = 4$.

Figure 19 in Appendix 4 shows linear regression on the $(\Delta E)^2$ for GS, there are clearly too few data points to accurately fit a line to the sample distribution. However, the figure follows the same trends as has been observed previously; GS trains very slowly, if at all, after the first few iterations, meaning that training an RBM using GS is mostly dependent on finding the right value of σ.

Figures 3 and figure 4, shows that the RBMs using BF trained faster than when using IS in all tested training phases. In the linear training phase however, table 4 indicated faster learning for IS compared to BF. The regrresion model predicted intercepts with the correct energy after 247 iterations for IS compared to 207 for BF ($N = 2$), and 462 compared to 367 for $N = 4$. This descrepency in results may either be a results of lacking noise reducing in the last simulations, as the higher acceptance rate of IS (documented in [7]) should lead to less variance in local energy, supposedly resulting in faster parameter optimization.

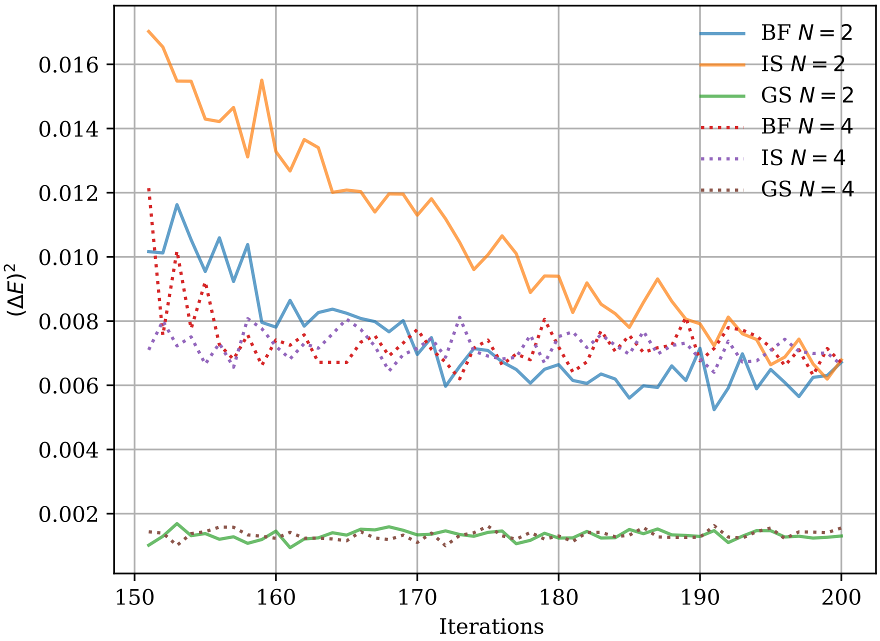


Figure 4: Comparison of methods: $(\Delta E)^2$ vs. training iterations in the final 50 iterationsusing $N = 2$ and $N = 4$ hidden nodes.

3.6 Validity of results

When determining the suitability of l and Δt I did not document the produced $\bar{E}$, as this would not have been feasible at that stage of the project, as I had not yet evaluated the training regime. This means that $l = 1.0$ and $\Delta t = 2.0$ might not be the best suited values after all, as lower variance is no guarantee for correct energy by it self. I did however later confirm that $\Delta t = 2.0$ performed better than $\Delta t = 1.0$. Furthermore, as I did not test values higher than $\Delta t = 2.0$, higher values of Δt may give lower $\hat{\sigma}$ still.

Table 1 was produced using noise reducing averaging over 5 optimization steps and 15 experiments, with consistent standard errors of $\sigma \sim 10^{-8}$ to $\sim 10^{9}$. Optimally, I should have included the MSE of $\hat{\sigma}$, but in order to limit the amount of information, I chose not to. A more in-depth study could be done by also looking into the variance of the error, yielding even more robust results.

For the results from the interacting electron system, I averaged over 10 runs in the first experiments, and in the last experiments, I did not average over multiple runs at all (due to the long run time). Any further use of the subsequent results should therefor first confirm any of the conclusions through further noise reducing averaging.

I did not test a wide range of learning rate values η in the interacting case, nor step length l, time step Δt or σ. It is therefor likely that one may obtain better suited values for these parameters.

4 Conclusions and main findings

All values of $(\Delta E)^2$ (single particle in one dimension) were $\sim 10^{-5}$ when using RBMs with the Metropolis Hastings algorithm with Importance Sampling (IS). When using brute force Metropolis Hastings approach (BF), $(\Delta E)^2$ was $\sim 10^{-4}$ to $\sim 10^{-5}$ depending on the learning rate η. Based on this, I conclude that when using an RBM, BF is more sensitive than IS to the choice of learning rate w.r.t accuracy of energy estimates. I furthermore conclude that the effect of η on $\hat{\sigma}$ is the same for both BF and IS, regardless of the value of Δt.

Based on $(\Delta E)^2$ alone, BF and IS performed equally well, producing $(\Delta E)^2 \approx 8.7 \times 10^{-8}$ with $\hat{\sigma} = 7 \times 10^{-8}$, when averaged over 15 single particle experiments. When used in an interacting system two electrons, I found that the RBMs were optimized similarly when using BF ($\eta = 0.38$) and IS ($\eta = 0.34$). I observed a short initial phase (~ 10 iterations) of rapid optimization, followed by a longer phase (~ 50 t0 ~ 100 training iterations for $N = 2$ and $N = 4$ respectively) of a slower, close-to linear optimization trajectory (regression coefficient $\sim 10^{-3}$ on iterations 10 to 20). After this "linear phase", sampling $(\Delta E)^2$ suggested a phase of exponentially decaying training rate, which converged both both $N = 2$ and $N = 4$ after ~ 200 iterations.

RBM training using Gibbs Sampling (GS) was conclusively shown to follow different optimization trajectories than BF and IS. Instead of the three phases described above, GS ($\eta = 0.26$) only exhibited two different training behaviors; rapid initial training after < 10 iterations, followed by a phase (persistently up to 200 training iterations) in which $\bar{E}$ remained close to constant (regression coefficient $\sim 10^{-4}$), ie. no optimization. The standard deviation of the distribution used in GS was shown to directly influence the estimated energy, $\bar{E}$, in the "no training-phase" of GS. Experiments using different σ values suggested that the training intercept of $\bar{E}$ with the exact energy of the system could be manipulated arbitrarily by changing σ. This project paper makes no general assertions on any optimization method, or other ways to find a suitable σ for a given system. I do however believe that such methods may already exist, or may be possible to derive.

References

[1] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science (New York, N.Y.)*, 355(6325):602–606, 2017.

[2] Vilde Moe Flugsrud. Solving quantum mechanical problems with machine learning. Master's thesis, University of Oslo, 2018.

[3] FYS4411 Project 2: The restricted Boltzmann machine applied to the quantum many body problem. `https://github.com/CompPhysics/ComputationalPhysics2/tree/gh-pages/doc/Projects/2020`, 2020.

[4] David J Griffiths. Introduction to quantum mechanics, 1995.

[5] Marius Jonsson. Standard error estimation by an automated blocking method, 2018.

[6] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Adaptive computation and machine learning. MIT Press, Cambridge, 2012.

[7] Johan Nereng. Vmc: Effects of importance sampling and jastrow factor on error estimates. `https://github.com/johanere/FYS4411/tree/master/Project%201`, 2019.

[8] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[9] M. Taut. Two electrons in an external oscillator potential: Particular analytic solutions of a coulomb correlation problem. *Phys. Rev. A*, 48:3561–3566, Nov 1993.

Appendices

A Appendix 1.

A.1 Analytic expression for the local energy

$$\begin{aligned} E_L &= \frac{1}{\Psi_{rbm}} \sum_{i=1}^{M} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 X_i^2 \right) \Psi_{rbm} + \frac{1}{\Psi_{rbm}} \sum_{i<j}^{P} \frac{1}{r_{ij}} \Psi_{rbm} \\ &= \frac{1}{\Psi_{rbm}} \sum_{i=1}^{M} \left(-\frac{1}{2}\nabla_i^2 \Psi_{rbm} \right) + \sum_{i=1}^{M} \frac{1}{2}\omega^2 X_i^2 + \sum_{i<j}^{P} \frac{1}{r_{ij}} \end{aligned} \tag{11}$$

And

$$\begin{aligned} \frac{1}{\Psi_{rbm}} \nabla^2 \Psi_{rbm} &= \frac{1}{\Psi_{rbm}} \nabla (\Psi_{rbm} \frac{1}{\Psi_{rbm}} \nabla \frac{1}{\Psi_{rbm}}) \\ &= (\frac{1}{\Psi_{rbm}} \nabla \Psi_T)^2 + \nabla (\frac{1}{\Psi_{rbm}} \nabla \Psi_{rbm}) \\ &= (\nabla \ln \Psi_{rbm})^2 + \nabla^2 \ln \Psi_{rbm} \end{aligned} \tag{12}$$

So

$$E_L = -\frac{1}{2} \sum_{k=1}^{M} \left((\nabla \ln \Psi_{rbm})^2 + \nabla^2 \ln \Psi_{rbm} \right) + \sum_{k=1}^{M} \frac{1}{2}\omega^2 X_i^2 + \sum_{i<j}^{P} \frac{1}{r_{ij}} \tag{13}$$

First derivative:

$$\begin{aligned} \frac{1}{\Psi_{rbm}} \nabla_k \Psi_{rbm} &= \nabla_k \ln \Psi_{rbm} \\ &= \nabla_k \big(\ln \frac{1}{Z} - \sum_{i}^{M} \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_{j}^{N} \ln(1 + e^{b_j + \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}) \big) \\ &= -\frac{(X_k - a_k)}{\sigma^2} + \sum_{j}^{N} w_{kj} \frac{\exp(b_j + \sum_i^M \frac{x_i w_{ij}}{\sigma^2})}{1 + e^{b_j + \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}} \\ &= -\frac{(X_k - a_k)}{\sigma^2} + \sum_{j}^{N} \frac{w_{kj}}{\sigma^2} \frac{1}{1 + e^{-v(\boldsymbol{X},j)}} \end{aligned} \tag{14}$$

Second derivative:

$$\begin{aligned} \nabla_k^2 \ln \Psi_{rbm} &= \nabla_k \big(-\frac{(X_k - a_k)}{\sigma^2} + \sum_{j}^{N} \frac{w_{kj}}{\sigma^2} \frac{1}{1 + e^{-b_j - \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}} \big) \\ &= -\frac{1}{\sigma^2} + \sum_{j}^{N} \frac{w_{kj}^2}{\sigma^4} \frac{e^{-v(\boldsymbol{X},j)}}{(1 + e^{-v(\boldsymbol{X},j)})^2} \end{aligned} \tag{15}$$

or

$$\nabla \ln \Psi_{rbm} = -\frac{(X_k - a_k)}{\sigma^2} + \sum_j^N \frac{w_{kj}}{\sigma^2} \text{logistic}(-v(j)) \tag{16}$$

and

$$\nabla^2 \ln \Psi_{rbm} = -\frac{1}{\sigma^2} + \sum_j^N \frac{w_{kj}^2}{\sigma^4} \text{logistic}^2(-v(j)) \exp(-v(j)) \tag{17}$$

Thus to compute E_L, one may use (13), (16) and (17).

B Appendix 2.

B.1 Derivatives w.r.t RBM parameters

Where $\frac{\bar{\Psi}_\beta}{\Psi[\beta]} = \frac{1}{\Psi_T[\beta]} \frac{d\Psi[\beta]}{d\beta}$. For parameter k: $\frac{1}{\Psi_T} \frac{\partial \Psi_T}{\partial \beta_k} = \frac{\partial}{\partial \beta_k} \log \Psi_T$
for $\beta = \boldsymbol{a}$:

$$\frac{\partial}{\partial a_k} \log \Psi_T = \frac{X_k - a_k}{\sigma^2} \tag{18}$$

for $\beta = \boldsymbol{b}$:

$$\frac{\partial}{\partial b_k} \log \Psi_T = \frac{1}{1 + e^{-v(\boldsymbol{X},k)}} \tag{19}$$

for $\beta = \boldsymbol{W}$:

$$\frac{\partial}{\partial W_{kl}} \log \Psi_T = \frac{X_k e^{v(\boldsymbol{X},l)}}{(1 + e^{v(\boldsymbol{X},l)})\sigma^2} = \frac{X_k}{(1 + e^{-v(\boldsymbol{X},l)})\sigma^2} \tag{20}$$

C Appendix 3.

C.1 Appendix 3: Local energy and parameter gradients for Gibbs sampling

As the only parts of Ψ_T that is traceable in E_L for this system are the derivatives of $ln\Psi_T$,

$$\Psi_{T,Gibbs} = \sqrt{F_{RBM}(\boldsymbol{X})} = \sqrt{\Psi_{rbm}} \tag{21}$$

Means that the expression for E_L remains the same as before, with the exceptions

$$\nabla \ln \Psi_{rbm} = \frac{1}{2} \nabla \ln \Psi_{rbm} \tag{22}$$

and

$$\nabla^2 \ln \Psi_{rbm} = \frac{1}{2} \nabla^2 \ln \Psi_{rbm} \tag{23}$$

And similarly for the parameter gradients, specifically for parameter number k of $\boldsymbol{\beta}$;

$$\frac{1}{\Psi_{T,Gibbs}} \frac{\partial \Psi_{T,Gibbs}}{\partial \beta_k} = \frac{1}{2} \frac{\partial}{\partial \beta_k} \log \Psi_{rbm} \tag{24}$$

D Appendix 4.

D.1 Appendix 4: Testing step length l and Δt

I averaged the standard error (using blocking) $\hat{\sigma}$ from 12 experiments. In each experiment, I used 8 MC simulations with parameter optimization (learning rate $\eta = 0.3$), with 2^{21} MC cycles with a 0.5 burn-in factor, for the shown values of l and Δt in figures 6 and 5. I found that $l = 1.0$ and $\Delta t = 2.0$ gave the best results among the tested values.

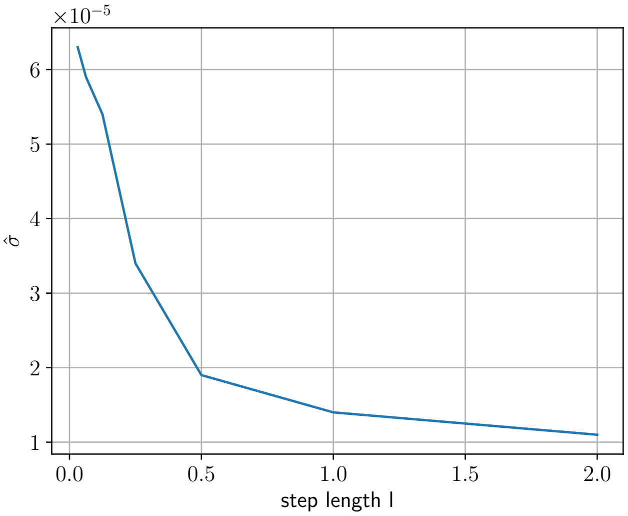


Figure 5: Importance Sampling: $\hat{\sigma}$ as a function of Δt

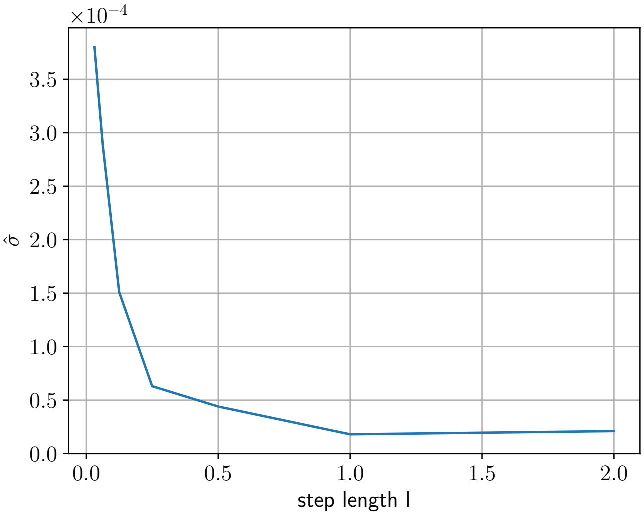


Figure 6: Brute Force $\hat{\sigma}$ as a function of l

D.2 Appendix 4: Testing learning rate η

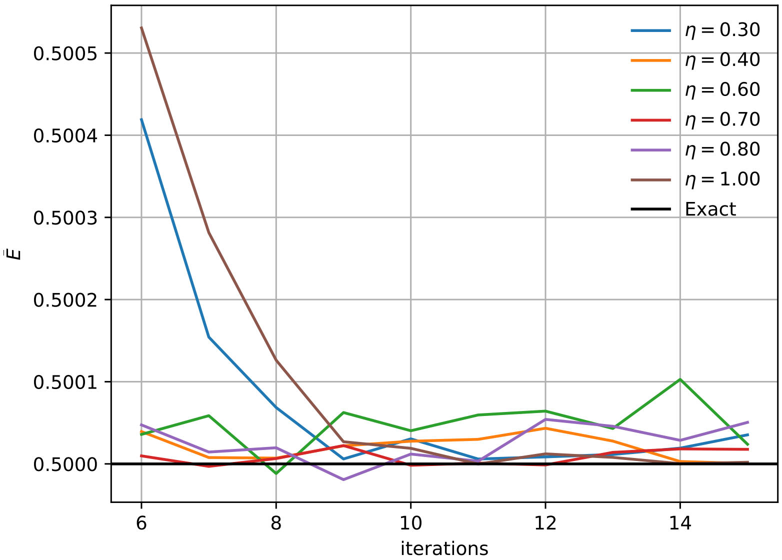


Figure 7: Brute Force: $\bar{E}$ for different values of η

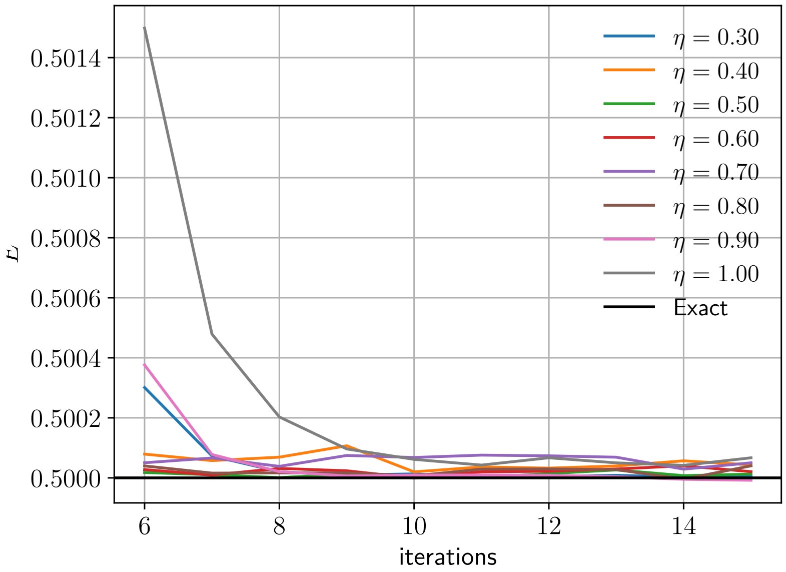


Figure 8: Importance Sampling: $\bar{E}$ for different values of η

D.3 Appendix 4: Testing σ and η GS

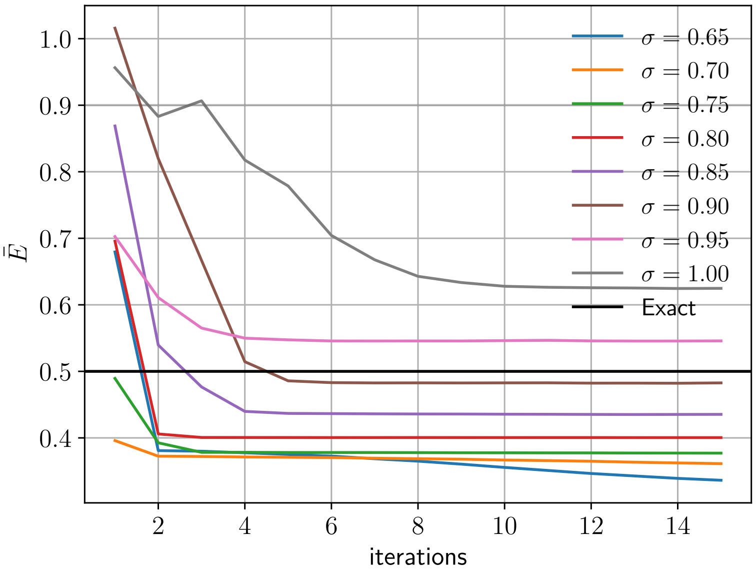


Figure 9: Gibbs Sampling: $\bar{E}$ for different values of σ

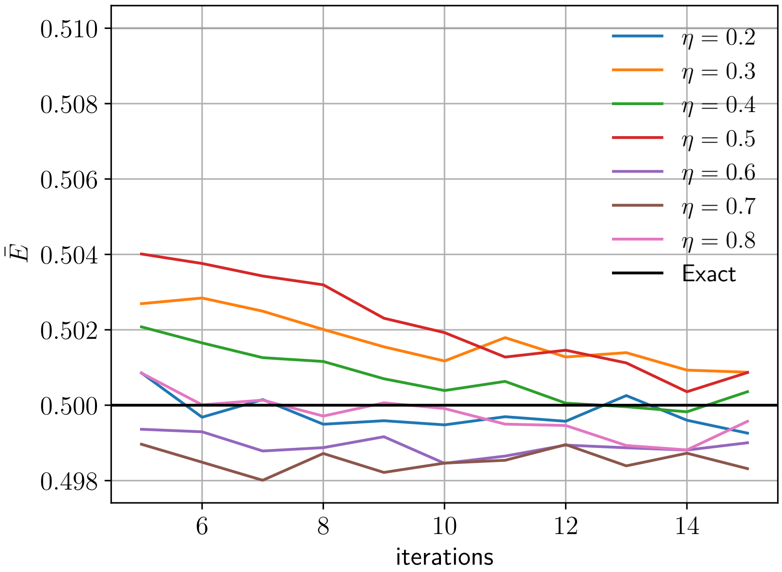


Figure 10: Gibbs Sampling: $\bar{E}$ for different values of η

D.4 Appendix 4: Training RBMs with interaction

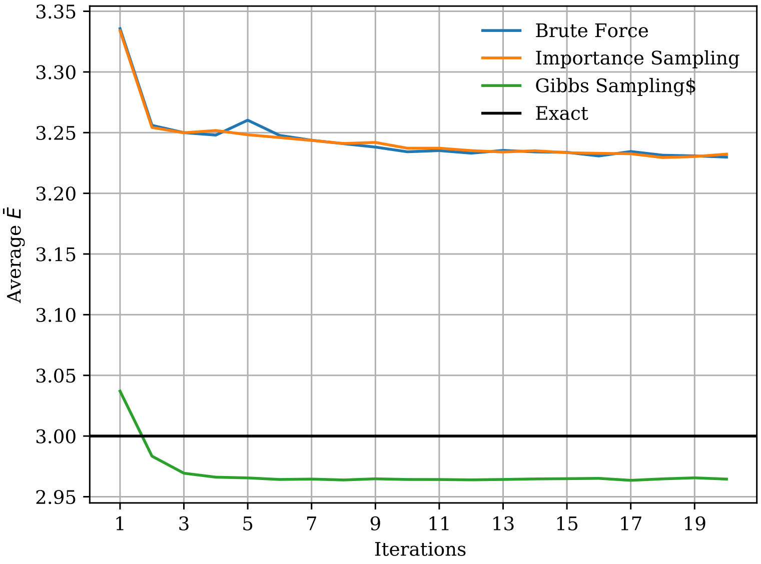


Figure 11: $P = D = 2$, $N = 4$: Mean $\bar{E}$ as a function of training iterations.

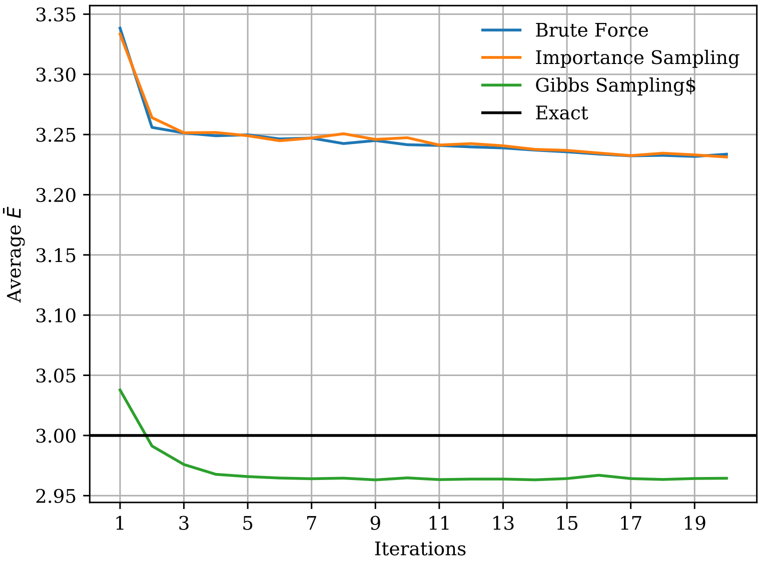


Figure 12: $P = D = 2$, $N = 2$: Mean $\bar{E}$ as a function of training iterations.

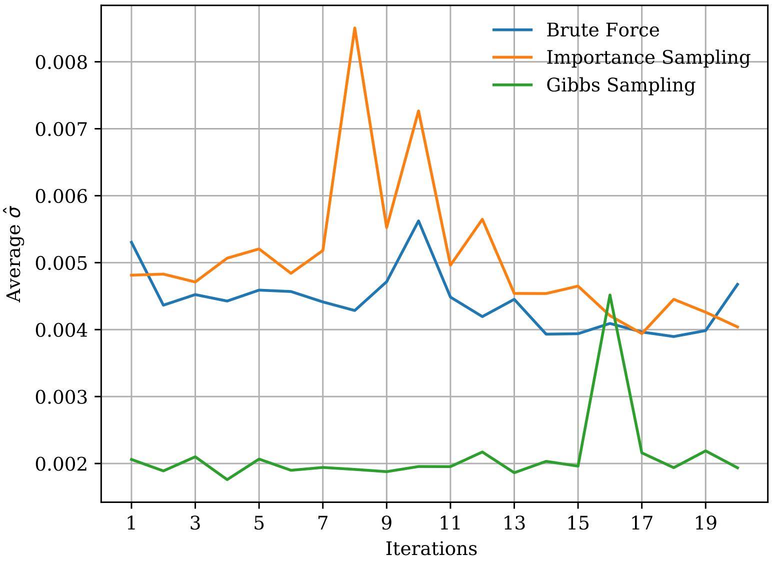


Figure 13: $P = D = 2$, $N = 2$: Mean $\hat{\sigma}$ as a function of training iterations.

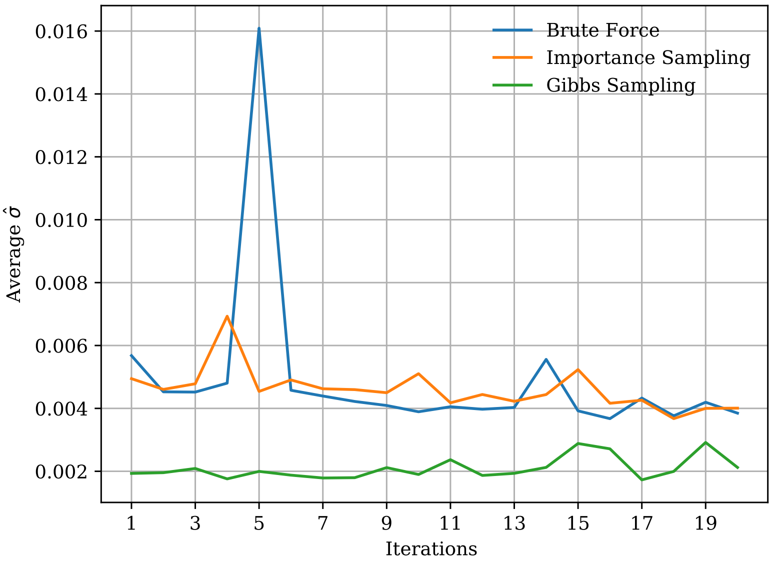


Figure 14: error

Figure 15: $P = D = 2$, $N = 4$: Mean $\hat{\sigma}$ as a function of training iterations.

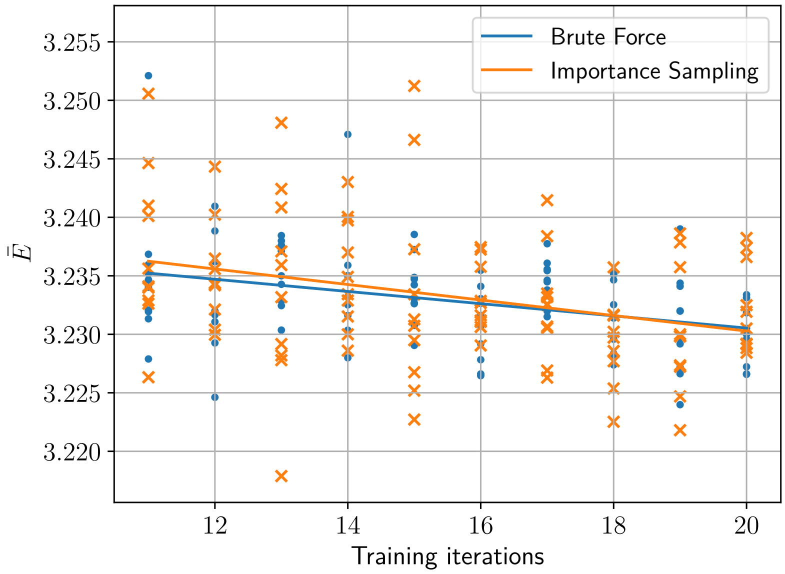


Figure 16: Brute Force & Importance Sampling: Linear regression on $\bar{E}$ from training iterations 10 to 20, based on 10 simulations

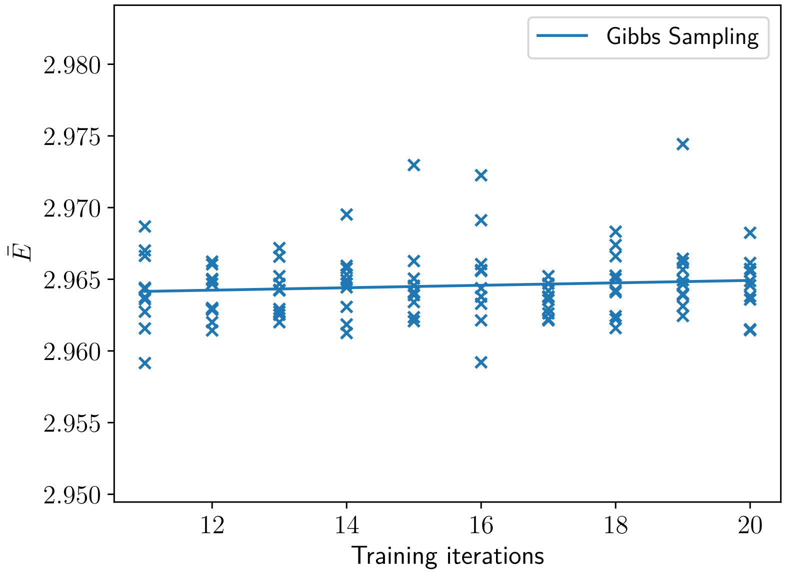


Figure 17: Gibbs Sampling: Linear regression on $\bar{E}$ from training iterations 10 to 20, based on 10 simulations

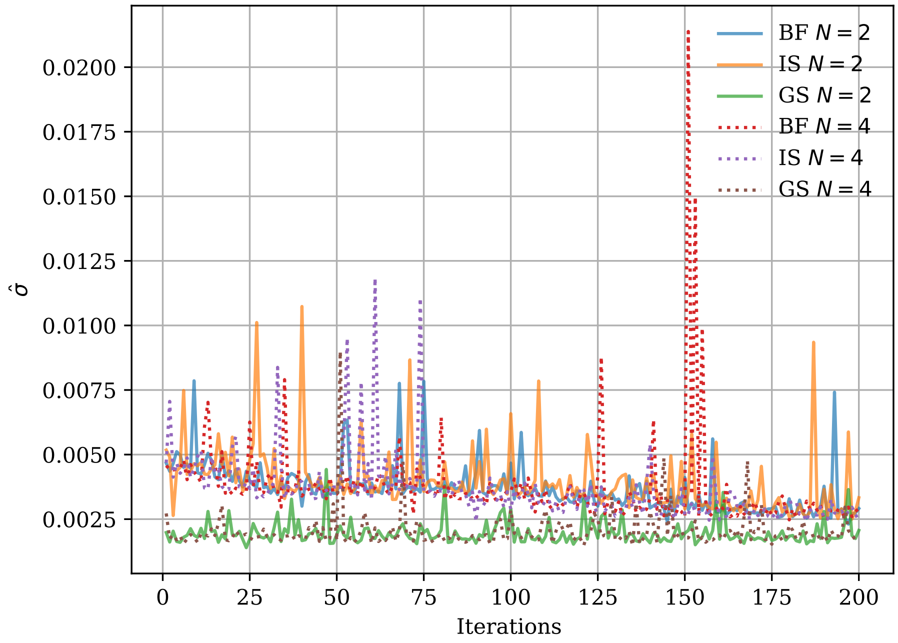


Figure 18: Comparison of methods: $\hat{\sigma}$ as a function of training iterations for all VMC methods, using $N = 2$ and $N = 4$.

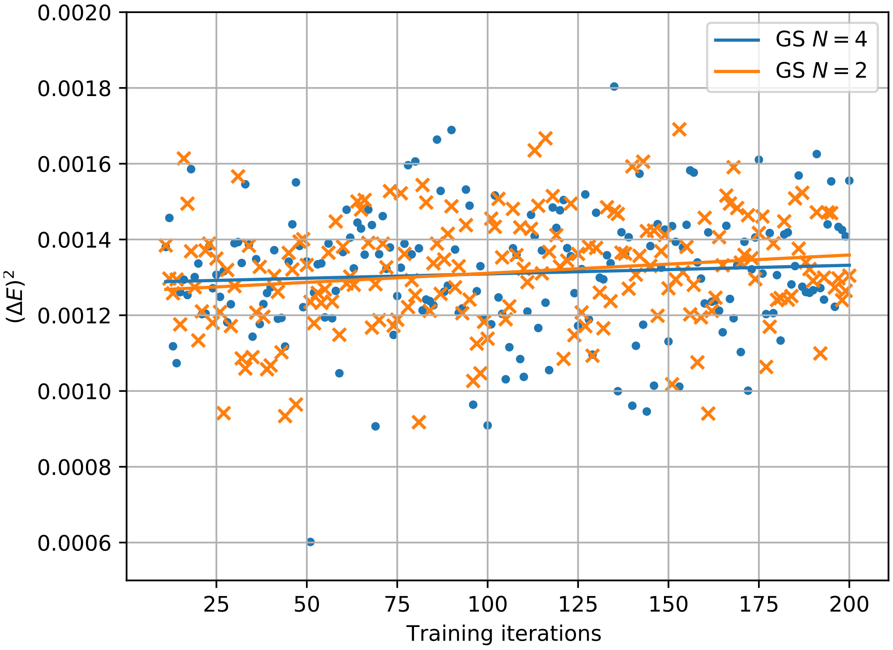


Figure 19: GS interacting: Regression on $(\Delta E)^2$ for $N = 2$ and $N = 4$ hidden nodes.