Ising Models: Numerical Simulations and Physical Applications

David Dye, Jon Huml, Harsha Tanneru

Introduction

Ising models [9] are idealized mathematical models of particles with binary spins. While this paper will focus on their use in equilibrium statistical mechanics, the models have numerous applications in physics, engineering, or neuroscience [6]. A central object in statistical mechanics is the partition function, or normalizing constant in a probability sense. Enumerating all possible states and transitions for this partition function, however, is a combinatorial problem and therefore must be approximated statistically and numerically for all but the smallest systems of particles. Furthermore, properties of the Boltzmann distribution dictate that most of the values in the partition function are zero or close to zero, and so a conventional numerical integration effort would be wasteful, even if possible [1].

The main goal of this paper will be to overcome these obstacles in simulating Ising systems efficiently and accurately with Markov Chain Monte Carlo methods. We verify our simulations with analytical solutions in certain cases where such solutions exist. The first third of the paper focuses on the prime method of simulation in the Ising literature: the Metropolis-Hastings algorithm [5]. Developed during the Manhattan Project in the 1940s, the algorithm remains one of the most well-known Monte Carlo methods in physics. Of course, such an old method is bound to lack some desired computational properties. For this, we evaluate more recent cluster lattice algorithms [7] like Swendsen–Wang [11] or the Wolff method [13] in the second third of the paper. In contrast to the single-spin flip approach of Metropolis-Hastings, these methods assign bonds between every pair of nearest neighbors and flip *clusters* of spins. The last third of the paper shows how Ising simulations are not merely for their own sake, but can be used in other physical tasks like numerical quantum path integration on a lattice [8]. Our paper should prepare the reader to implement Monte Carlo methods in statistical mechanics and apply the simulations to various physical problems.

Methods and Results

Background and Overview

The Ising model is a mathematical model of ferromagnetism in statistical mechanics. It was proposed by Wilhelm Lenz in 1920 and solved exactly by Ernst Ising in 1925. In the Ising model, each site on a lattice (a regular arrangement of points in space) is occupied by a "spin," which can take on one of two values, usually represented by +1 and -1. These spins interact with each other through a simple pairwise interaction, in which adjacent

spins tend to align with each other. The strength of this interaction is represented by a parameter called the coupling constant.

The Ising model can be used to study the behavior of ferromagnetic materials, in which the spins of the electrons align in the same direction, leading to a net magnetization. It can also be used to study phase transitions, such as the transition from a disordered state to an ordered state as the temperature is lowered. The Ising model has been widely studied and has many applications in fields such as statistical physics, condensed matter physics, and computer science.

Mathematical formulation and Analytical Solution

We look at Ising models in 2-D and 3-D lattices and study how the properties - energy, magnetisation, and heat capacity - vary with temperature. The following notation is used to define equations in the upcoming sections.

s_i is the spin at site i in a 1-D lattice
$s_{i,j}$ is the spin at site (i, j) in a 2-D lattice
$s_{i,j,k}$ is the spin at site (i, j, k) in a 3-D lattice
α_j is a state of the lattice i.e; $(s_0, s_1, s_2, \ldots, s_n)$
J is the exchange coupling constant
k_B is the Boltzmann constant
T is the temperature
N is the number of spins along one axis of the lattice (a 10×10 lattice has $N = 10$)
$Z(T)$ is the partition function
$P(\alpha)$ is the probability mass function of the system being in state α
$M(H, T)$ is the mean magentisation of the lattice
m is the magentisation of the lattice at current state
H is the Hamiltonian, or total energy, of the system.
E is used interchangeably with H to denote energy.

1-D Ising Model

The energy of the one-dimensional Ising model with no external magnetic field at state α_j is determined by the Hamiltonian as:

$$H(\alpha_j) = -J \sum_{i,j} s_i s_j$$

And the probability of the system being in state α_j is given by $P(\alpha_j)$:

$$P(\alpha_j) = \frac{e^{-\frac{E(\alpha_j)}{k_B T}}}{Z(T)}$$

The partition function, which is the fundamental quantity in statistical mechanics, is obtained by summation over all possible states. It can be written as:

$$Z(T) = \sum_{\alpha_s} e^{-\frac{E(\alpha_s)}{k_B T}}$$

From [1], the magnetisation $M(H,T)$ is given by:

$$M(H,T) = \frac{e^k sin(\beta H)}{\left(e^{2k} sin^2(\beta H) + e^{-2k}\right)^{1/2}}$$

$M(H,T)$ is an analytical function for for real H and positive T. It is worth noting that the one-dimensional Ising model does not exhibit a phase transition at positive temperatures, unlike the two-dimensional and three-dimensional versions of the model.

2-D Ising Model

Synonymous functions are defined for the 2-D and 3-D cases. The 2-D Hamiltonian with no external magnetic field is:

$$H = \frac{-J}{2}\left(\sum_{i,j} s_{i,j}s_{i+1,j} + \sum_{i,j} s_{i,j}s_{i,j+1}\right)$$

The 2-D partition function is:

$$Z(T) = \sum_{s} e^{-\frac{H}{kT}}$$

The total lattice magnetization for a given state is:

$$m = \sum_{i,j} s_{i,j}$$

And the mean lattice magnetization for a given state is:

$$M = \frac{1}{N^2}\sum_{i,j} s_{i,j}$$

3-D Ising Model

Similarly to the 2-D case, the 3-D functions are as follows. The Hamiltonian in three dimensions is:

$$H = -\frac{J}{2}\sum_{i,j,k}\left(s_{i+1,j,k} + s_{i-1,j,k} + s_{i,j+1,k} + s_{i,j-1,k} + s_{i,j,k+1} + s_{i,j,k-1}\right)$$

The partition function is:

$$Z(T) = \sum_s e^{-\frac{H}{kT}}$$

The total magnetization is:

$$m = \sum_{i,j,k} s_{i,j,k}$$

And the mean magnetization of a lattice state is:

$$m = \frac{1}{N^3} \sum_{i,j,k} s_{i,j,k}$$

Numerical Solution

Solving the Ising model can be challenging because the number of possible states scales exponentially with the number of lattice sites (the number of states in an p-dimensional lattice with n sites along each dimension is $(2p)^n$). Numerically computing the exact expectation of energy and magnetisation involves iterating over all possible states, and therefore is not feasible at high values of n and p.

One approach to solving the Ising model is through the use of Monte Carlo simulations. Monte Carlo methods are a class of computational algorithms that use random sampling to obtain numerical results. Monte Carlo methods leverage the law of large numbers to compute integrals described by the expected value of some random variable as the empirical mean of independent samples of the variable. The algorithm is relatively simple to implement and can be easily parallelized, making it an efficient tool for studying complex systems.

When the probability distribution of the variable is parameterized (state α_j is the parameter in our case), a class of Monte Carlo methods called Markov chain Monte Carlo (MCMC) methods are used. Unlike Monte Carlo sampling methods that are able to draw independent samples from the distribution, Markov Chain Monte Carlo methods draw samples where the next sample is dependent on the existing sample, called a Markov Chain. That is, as the Markov chain evolves, the samples being generated by the MCMC method will converge to an equilibrium (stationary) distribution. Thus, samples drawn from the Markov chain after many iterations will be approximately equal to samples drawn from the desired target distribution.

In an MCMC simulation of the Ising model, random configurations of the spins in the system are generated and the energy of each configuration is calculated using the Ising model's exponential function. The probability of a particular configuration being observed can then be estimated using statistical techniques, and the properties of the system, such as its magnetization and susceptibility, can be calculated using empirical mean. We will now discuss the Metropolis-Hastings and Gibbs Sampling MCMC methods in detail.

MCMC Method: Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm [10] is as follows:

1. Create a proposal distribution $V(\alpha_{k+1}|\alpha_k)$ to generate the next sample α_{k+1} given the current sample α_k.
2. Compute un-normalized probabilities of each state. Note that we are only interested in the ratio of probabilities, so it is not necessary to compute the partition function. The acceptance ratio for moving from state α_k to α_{k+1} is:

$$r = \frac{P(\alpha_{k+1})}{P(\alpha_k)} = \frac{\frac{exp(-\frac{E(\alpha_{k+1})}{k_B T})}{Z(T)}}{\frac{exp(-\frac{E(\alpha_{k+1})}{k_B T})}{Z(T)}} = \exp\left(-\frac{\Delta E}{k_B T}\right)$$

The pseudo code for Metropolis-Hastings [1] MCMC sampling algorithm is given below.

1. Start with an arbitrary spin configuration $\alpha_k = \{s_1, s_2, s_3, \ldots, s_n\}$
2. Generate a proposal configuration α_{prop} by:
   a. Choosing a particle (i, j) (or (i, j, k) for 3-D) at random
   b. Flipping its spin
3. Calculate the energy of the proposal configuration $E_{\alpha_{prop}}$
4. If $E_{\alpha_{prop}} \leq E_{\alpha_k}$, accept the trial by setting $\alpha_{k+1} = \alpha_{prop}$
5. If $E_{\alpha_{prop}} > E_{\alpha_k}$, accept with relative probability $R = \exp\left(-\frac{\Delta E}{k_B T}\right)$
   a. Choose a uniform random number $0 \leq r \leq 1$
   b. Set $\alpha_{k+1} = \alpha_{prop}$ if $R \geq r$ (accept)
   c. Set $\alpha_{k+1} = \alpha_k$ if $R < r$ (reject).
6. Repeat until convergence.

MCMC Method: Gibbs Sampling Algorithm

Like Metropolis-Hastings, Gibbs sampling [4] is a variant of MCMC, but it is conceptually simpler. If we want to sample from a distribution over several random variables, Gibbs sampling fixes all but one random variable, samples that one conditioned on the

others, and then repeats the process for each random variable. So, all we need are the conditional distributions. Note that, the acceptance ratio computed with the conditional distributions of site $s_{i,j}$ is same as that with joint distribution α_j.

The pseudo code for Gibbs MCMC sampling algorithm [10] is given below.

1. Start with an arbitrary spin configuration $\alpha_k = \{s_1, s_2, s_3, \ldots, s_n\}$
2. For each site (i, j) (or (i, j, k) for 3-D lattice)
   (a) Generate a proposal configuration α_{prop} by flipping spin of particle (i, j) (or (i, j, k) for 3-D lattice)
   (b) If $E_{\alpha_{prop}} \leq E_{\alpha_k}$, accept the trial by setting $\alpha_{k+1} = \alpha_{prop}$
   (c) If $E_{\alpha_{prop}} > E_{\alpha_k}$, accept with relative probability $R = \exp\left(-\frac{\Delta E}{k_B T}\right)$
      i. Choose a uniform random number $0 \leq r \leq 1$
      ii. Set $\alpha_{k+1} = \alpha_{prop}$ if $R \geq r$ (accept)
      iii. $\alpha_{k+1} = \alpha_k$ if $R < r$ (reject).
3. Repeat until convergence

MCMC Simulation of 2-D Ising Model

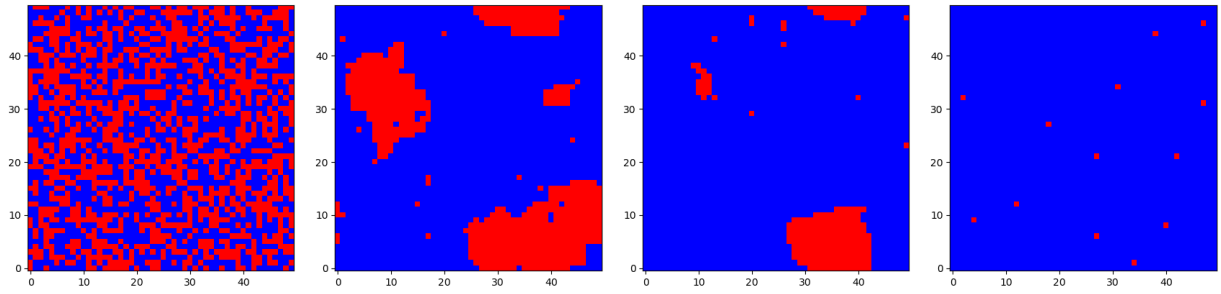

Figure 1: **Metropolis-Hastings**. From left to right, lattices at iteration 0, $t/3$, $2t/3$, and t iterations timestamps of the Metropolis-Hastings algorithm (where t is the number of iterations for convergence). Simulations carried out on a 50×50 lattice of spins with $J = 1$ and $T = 1.5$. A blue cell represents a spin of $+1$ and a red cell represents a spin of -1.

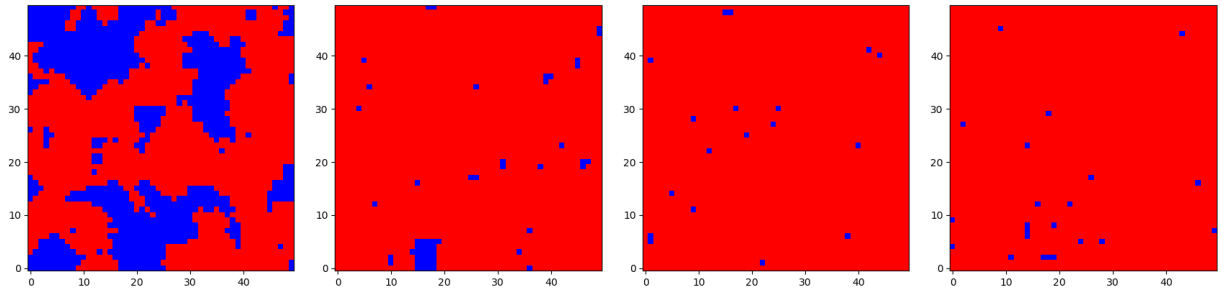

Figure 2: **Gibbs Sampling**. From left to right, lattices at iteration 0, $t/3$, $2t/3$, and t iterations timestamps of the Gibbs sampler (where t is the number of iterations for convergence). Simulations carried out on a 50×50 lattice of spins with $J = 1$ and $T = 1.5$. A blue cell represents a spin of $+1$ and a red cell represents a spin of -1.

The energy of the system is minimized when the spins in the system are aligned. This is because the interaction energy between spins in an Ising model is minimized when the spins are aligned. This is consistent with the observed results from both Monte Carlo simulations, where all the spins are aligned in the final converged state.

3.

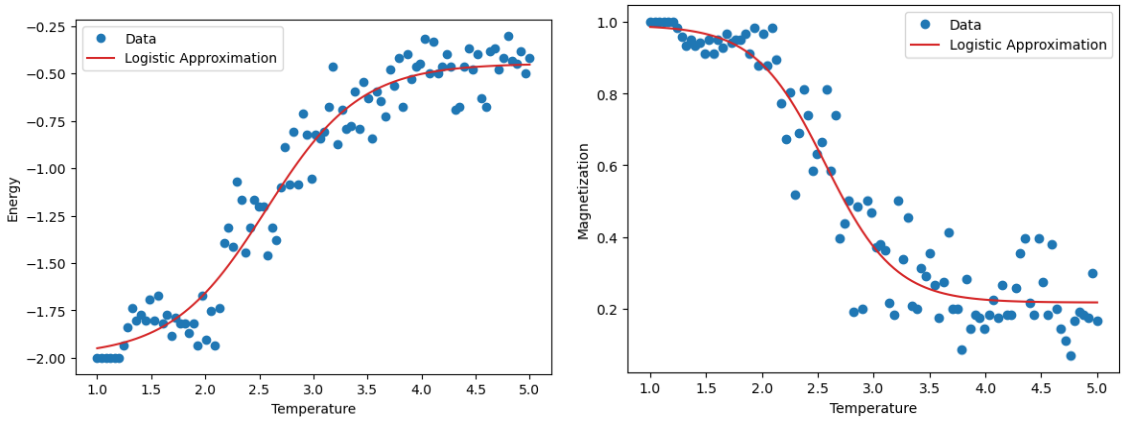

Figure 3: **Left**: Metropolis-Hastings sampling. The average of the final energies of 5 2-D spin states simulated using Metropolis-Hastings sampling at $N = 100$ temperatures ranging from 1 to 5. **Right**: The average of the magnitude of the final magnetizations of 5 2-D spin states simulated at $N = 100$ temperatures ranging from 1 to 5. Both plots consider the average measurement *per spin particle*.

4.

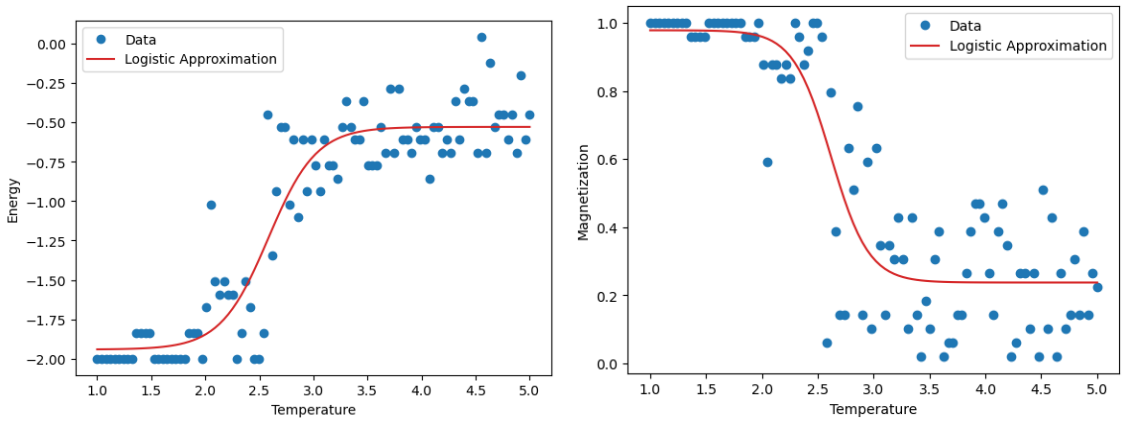

Figure 4: **Left**: Gibbs sampling. The average of the final energies of 5 2-D spin states simulated using Gibbs sampler at $N = 100$ temperatures ranging from 1 to 5. **Right**: The average of the magnitude of the final magnetizations of 5 2-D spin states simulated at $N = 100$ temperatures ranging from 1 to 5. Both plots consider the average measurement *per spin particle*.

Magnetization is a measure of the overall alignment of the spins in the system. When the spins are randomly oriented, the magnetization is zero. When the spins are all aligned in the same direction, the magnetization is maximum. At high temperatures, the thermal energy is large and the spins are more likely to be randomly oriented, leading to a lower magnetization and higher energy. At low temperatures, the thermal energy is small and the spins are more likely to be aligned, leading to a higher magnetization and lower energy. We observer the same trends in empirical results from MCMC simulations.

MCMC Simulation of 3-D Ising Model

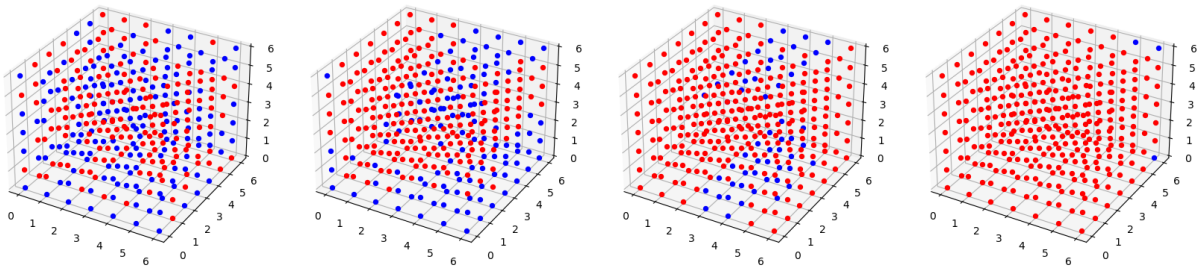

Figure 5: **Metropolis-Hastings**. From left to right, lattices at iteration 0, $t/3$, $2t/3$, and t iterations timestamps of the Metropolis-Hastings algorithm (where t is the number of iterations for convergence). Simulations were carried out on a $7 \times 7 \times 7$ lattice of spins with $J = 1$ and $T = 2$. A blue cell represents a spin of $+1$ and a red cell represents a spin of -1.

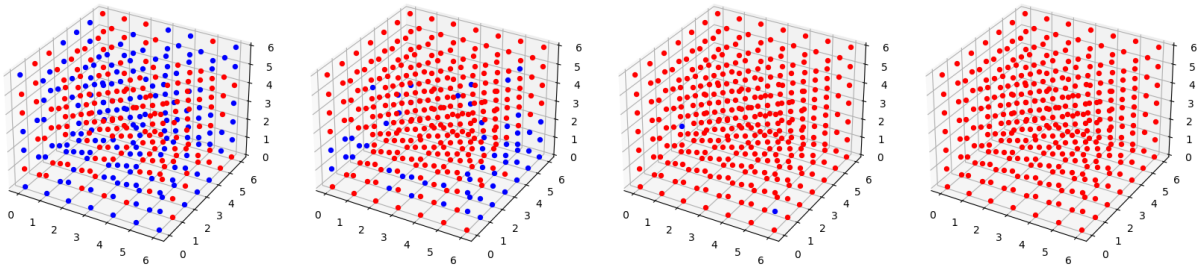

Figure 6: **Gibbs Sampling**. From left to right, lattices at iteration 0, $t/3$, $2t/3$, and t iterations timestamps of the Gibbs sampler (where t is the number of iterations for convergence). Simulations carried out on a $7 \times 7 \times 7$ lattice of spins with $J = 1$ and $T = 2$. A blue cell represents a spin of +1 and a red cell represents a spin of −1.

Just like the 2-D lattice, all spins are aligned at convergence in 3-D Ising model simulation.

7.

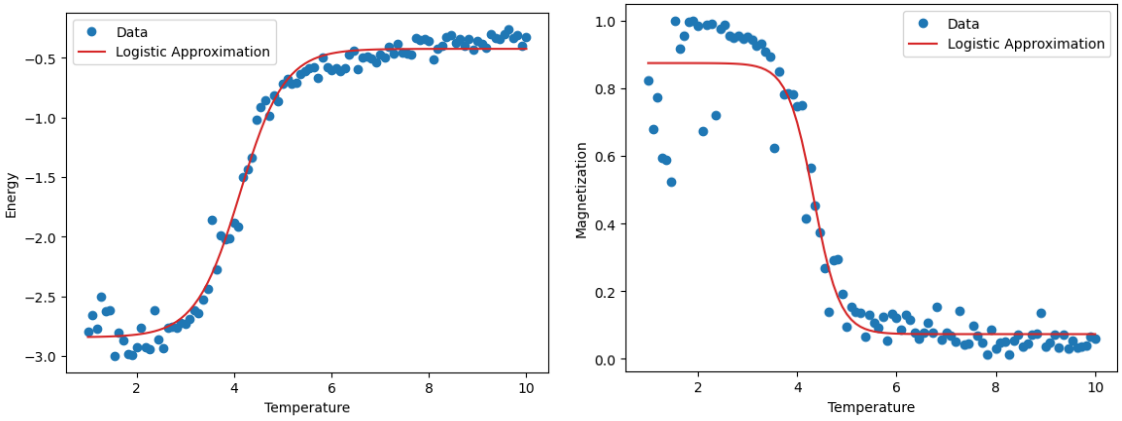

Figure 7: **Left**: Metropolis-Hastings sampling. The average of the final energies of 5 3-D spin states simulated using Metropolis-Hastings sampling of a $8 \times 8 \times 8$ lattice at $N = 100$ temperatures ranging from 1 to 1−. **Right**: The average of the magnitude of the final magnetizations of 5 2-D spin states simulated at $N = 100$ temperatures ranging from 1 to 1−. Both plots consider the average measurement *per spin particle*.

8.

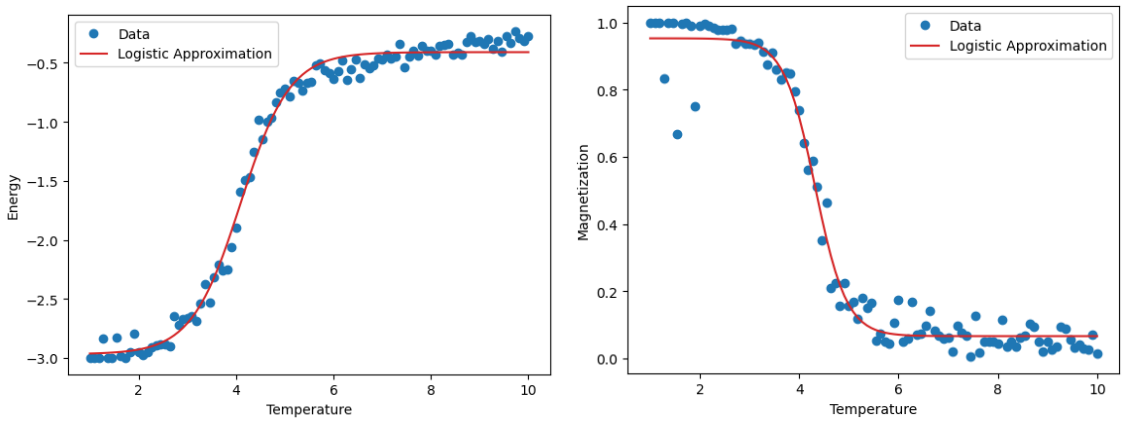

Figure 8: **Left**: Gibbs sampling. The average of the final energies of 5 3-D spin states simulated using Metropolis-Hastings sampling of a $8 \times 8 \times 8$ lattice at $N = 100$ temperatures ranging from 1 to 1−. **Right**: The average of the magnitude of the final magnetizations of 5 2-D spin states simulated at $N = 100$ temperatures ranging from 1 to 1−. Both plots consider the average measurement *per spin particle*.

Critical Temperature

We fit a logistic curve on each magnetisation v/s temperature plot to find the critical temperature T_c. More detail on this method is provided in the Swendsen-Wang section. We observe that the critical temperature T_c is higher for 3-D Ising model compared to 2-D Ising model.

Algorithm	$2-D$	$3-D$
Metropolis-Hastings	2.5667	4.3333
Gibbs Sampling	2.6176	4.3221

Table 1: Critical Temperature $T_c(^\circ C)$ for $2-D$ and $3-D$ lattices

MCMC Diagnostics: Acceptance rate

The acceptance rate is the fraction of proposed moves that are accepted in the sampling process. Acceptance rate determines the efficiency of the sampling process. We notice that both Metropolis-Hastings and Gibbs sampling have similar acceptance rates. Acceptance rate increases with temperature. This is expected as the acceptance ratio $R = \exp -\frac{\Delta E}{k_B T}$ increases with temperature. Physically, this means that fewer proposals for a spin flip are rejected at higher temperatures.

9.

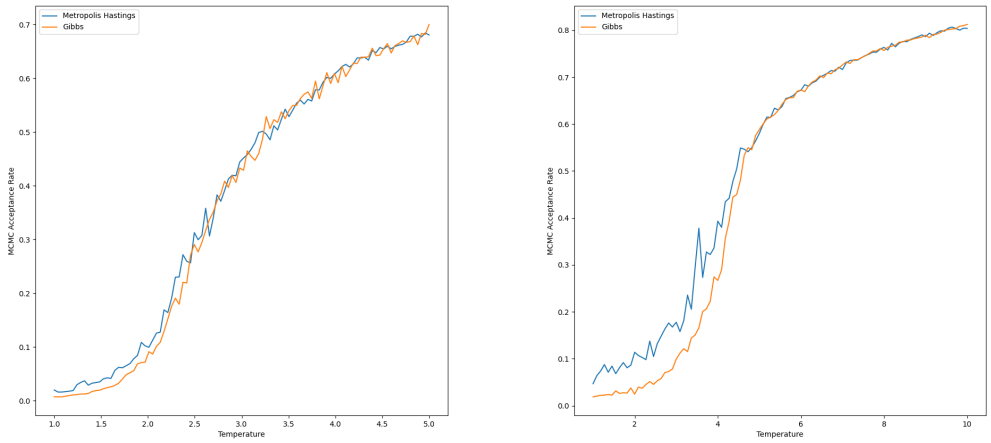

Figure 9: **Left**: Acceptance rate. The acceptance rate of 5 2-D spin lattices of shape 7×7 simulated using Metropolis-Hastings and Gibbs sampler at $N = 100$ temperatures ranging from 1 to 5. **Right**: The acceptance rate of 5 3-D spin lattices of shape $8 \times 8 \times 8$ simulated using Metropolis-Hastings and Gibbs sampler at $N = 100$ temperatures ranging from 1 to 10.

Alternative Update Methods: Swendsen-Wang Algorithm

The Metropolis-Hastings algorithm creates a Markov Chain which updates according to probabilistic flips of each spin in the Ising lattice, and its convergence rate near critical points (i.e. a phase transition) is very slow [11]. Therefore, alternative methods have been developed to avoid the critical slowing-down effect. Two of these methods are essayed in this paper, namely the Swendsen-Wang Algorithm and the Wolff Algorithm. Both methods replace entire spin "clusters" rather than individual spins, which greatly increases the efficiency of evaluating large spin systems near the critical temperature.

Consider a 2- or 3-dimensional Ising model lattice of randomly arranged particles with spins +1 or −1. A "spin cluster" is a group of bonded, orthogonally-adjacent particles with the same spin. A bond forms (event B) between identical-spin particles i and j with probability $P(B|\sigma_i = \sigma_j) = 1 - \exp(-2J/T)$. If i and j do not have the same spin, then the probability that a bond forms between them is $P(B|\sigma_i \neq \sigma_j) = 0$.

The Swendsen-Wang Algorithm converges to the same stationary state as the Markov chain given by the Metropolis-Hastings algorithm. The Swendsen-Wang Algorithm is as follows [11]:

1. Form an initial spin state, where each spin is +1 or −1 with probability $\frac{1}{2}$.

2. Create a list of spin clusters within that state according to the bond probabilities given above.

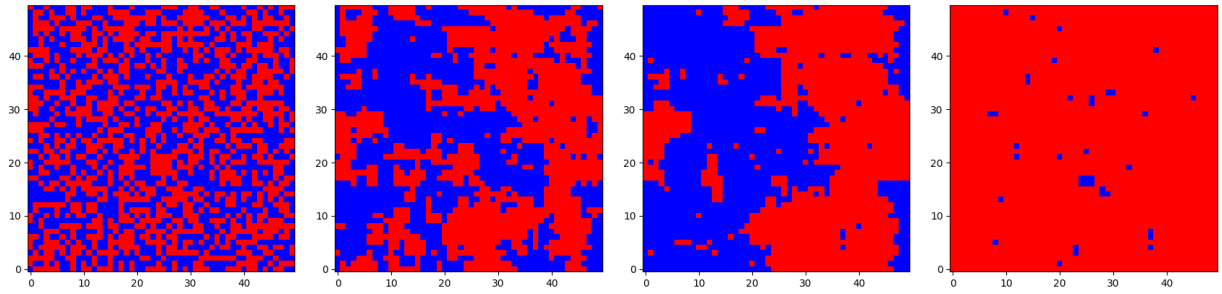

Figure 10: The first, third, fifth, and seventh iterations of the Swendsen-Wang Algorithm for a 50×50 lattice of spins with $J = 1$ and $T = 1.5$. A blue cell represents a spin of +1 and a red cell represents a spin of −1.

3. Iterate over the list of spin clusters, and with probability $\frac{1}{2}$, flip the sign of every spin within a cluster.

4. Using the resulting spin state from step 3 as the updated state, repeat steps 2 and 3 until a pre-specified number of iterations is reached.

Our method for finding all spin clusters was as follows:

1. Define empty lists named ”blacklist” and ”connections.”

2. Iterate over the locations of all spins in the spin state. If a spin is in the blacklist, skip that spin. Otherwise, add a new empty list named ”cluster” to the connections list. Add the location of the spin to both the blacklist and the cluster list.

3. To form the cluster, find all non-blacklisted spins that are adjacent and bonded to the current spin. Add the locations of these spins to both the blacklist and the cluster list. Then repeat this for every adjacent spin.

4. At the end of the iterations, the blacklist will contain the locations of all of the spins (once each), and the connections list will contain all of the spin clusters.

This method is recursive, and is computationally expensive for large spin states. There is no obvious way to make this process more efficient (such as saving bonded spins) because bonds form randomly for every new spin state. However, numerical testing of the algorithm shows that it is able to operate without reaching the maximum recursion depth for up to a sixty by sixty lattice, which is large enough for our purposes.

An example of how an initial spin state evolves according to the Swendsen-Wang Algorithm is given in Figure 10. A spin state that is almost entirely red or blue is magnetized with low energy, and a spin state that has a random distribution of red and blue is not magnetized with high energy. As can be seen from the figure, it took a total of seven iterations for the Swendsen-Wang algorithm to converge to a magnetized state from the given random initial state, which has a temperature reasonably close to the analytically-proven

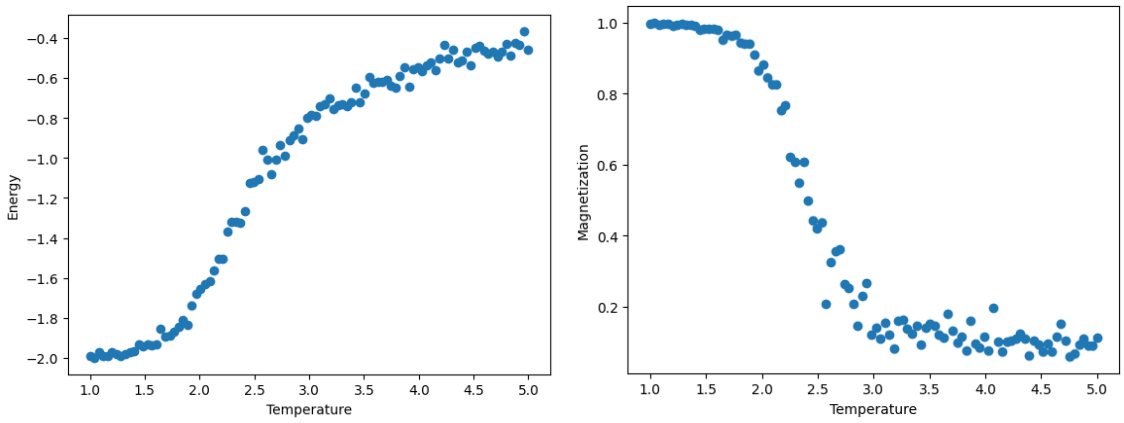

Figure 11: **Left**: The average of the final energies of 10 2-D spin states simulated at $N = 100$ temperatures ranging from 1 to 5. **Right**: The average of the magnitude of the final magnetizations of 10 2-D spin states simulated at $N = 100$ temperatures ranging from 1 to 5. Both plots consider the average measurement *per spin particle*.

critical temperature of 2.269. Because of how quickly the Swendsen-Wang Algorithm converges, we took the maximum number of iterations to be 10 for all future plots.

An issue with the Swendsen-Wang Algorithm (and cluster algorithms generally) is that its final state is highly dependent on the initial state. Therefore, some simulations of low-temperature states may take longer to converge and will appear non-magnetic, and some simulations of high-temperature states may begin in a magnetized state and will appear magnetic. Therefore, it makes sense to take the average measurements from many simulations of a given temperature. With this technique, we generated data points representing the average absolute magnetizations and energies of final spin states from 10 repeated measurements for each of $N = 100$ uniformly-spaced temperature values between 1 and 5. These data points are plotted in Figure 11.

A natural continuation is to try and estimate the critical temperature $T_c = 2.269$ using these datasets. To make the estimate, we minimized the root mean squared error (RMSE) of the logistic function and the data set. Mathematically, we found L, k, x_B, and a by minimizing the sum:

$$\min_{L,k,x_B,a} \sqrt{\sum_{i=1}^{N} \left(y_i - \frac{L}{1 + e^{-k(x_i - x_B)}} - a \right)^2} \tag{1}$$

Where x_i is temperature i and y_i is the corresponding data point at x_i. We used scipy.optimize.minimize to calculate the optimal L, k, x_B, and a to fit the logistic function f:

$$f(x) = \frac{L}{1 + e^{-k(x - x_B)}} + a \tag{2}$$

Overlays of these logistic functions are given in Figure 12.

The value of x_B can be used to approximate the critical temperature at which a phase

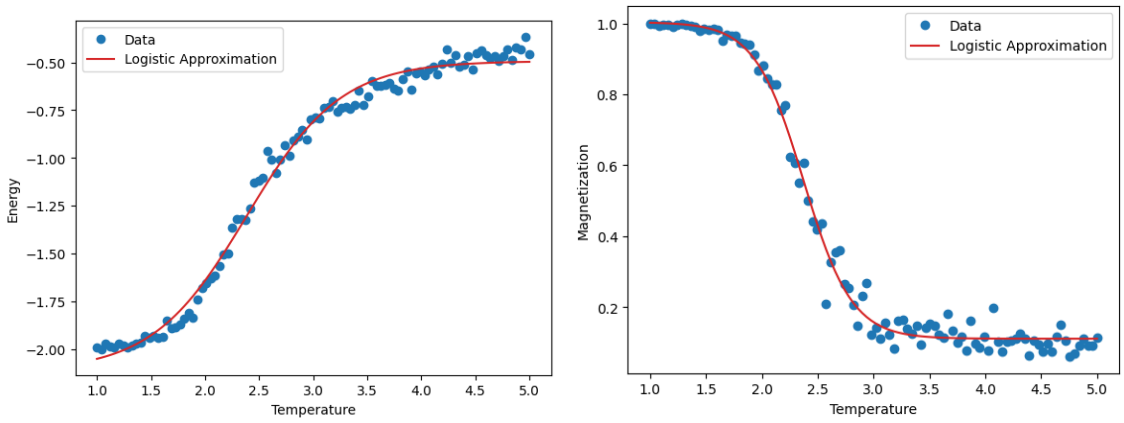

Figure 12: **Left**: See Figure 11 for data interpretation. The overlaid logistic curves minimize the RMSE between the curve and the data points.

transition occurs, since it is where the logistic approximation changes from concave to convex. For the 2-D Swendsen-Wang algorithm, using the logistic curve approximating simulated magnetization values, we obtain $x_B = 2.34$. The relative error between this and the exact analytical result is:

$$\frac{|T_c - x_B|}{T_c} = \frac{|2.269 - 2.34|}{2.269} = 0.0313 = 3.13\% \tag{3}$$

So, the logistic approximation applied to data obtained using the 2-D Swendsen-Wang Algorithm estimates the critical temperature for a phase transition with relative error 3.13%. To be absolutely clear, the obtained value of x_B varies between trials due to randomness caused by cluster algorithms and the fact that only 10 repetitions were averaged over. The reported values of x_B for all cluster algorithms vary between simulations, but on the order of 0.1.

Extrapolating the Swendsen-Wang Algorithm to 3-dimensions is straightforward, and the only algorithmic difference involves considering neighbors along the z-axis when forming spin clusters. Figure 13 gives an example of how an initial 3-D spin state evolves according to the Swendsen-Wang Algorithm. We used matplotlib's mpl_toolkits.mplot3d Axes3D to create these figures. The 3-D case can simulate up to a $15 \times 15 \times 15$ spin lattice with the algorithm described, after which the maximum recursion depth is reached.

Figures 14 and 15 were created following the exact same procedure as in the 2-D case, and have the same interpretations. A noteworthy feature is that the average energy of a particle in a magnetized state in 3 dimensions is -3, whereas in 2 dimensions it was -2. This makes sense, since the number of neighbors with the same spin increased from approximately four to approximately six when going from 2-D to 3-D.

As expected, we see that a higher temperature is required to induce a phase transition from a magnetized state to a non-magnetized state in the 3-D case versus the 2-D case.

The logistic regression to the magnetization curve produces an x_B value of $x_B = 4.27$. There is no analytical solution for the 3-D case of the Ising model, but numerical simulations

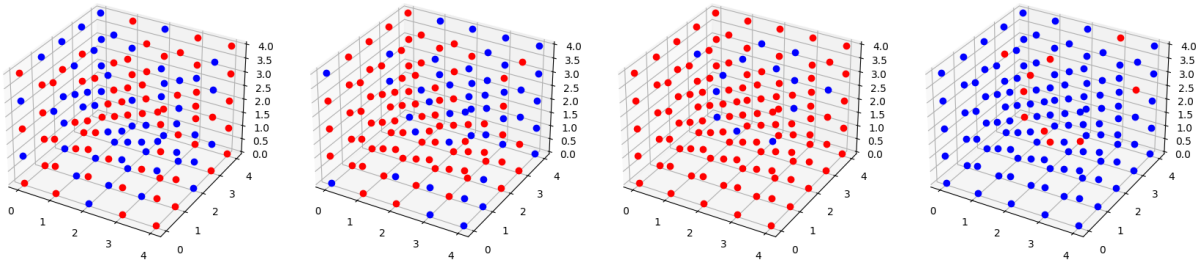

Figure 13: The first, second, third, and fourth states of the Swendsen-Wang Algorithm for a $5 \times 5 \times 5$ lattice of spins with $J = 1$ and $T = 3$. A blue dot represents a spin of +1 and a red dot represents a spin of −1.

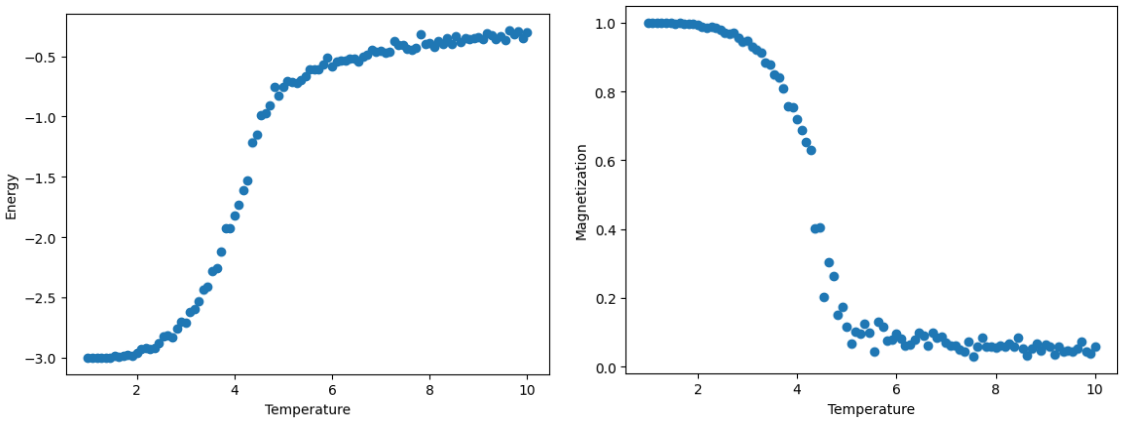

Figure 14: **Left**: The average of the final energies of 10 3-D spin states simulated at $N = 100$ temperatures ranging from 1 to 10. **Right**: The average of the magnitude of the final magnetizations of 10 3-D spin states simulated at $N = 100$ temperatures ranging from 1 to 10. Both plots consider the average measurement *per spin particle*.

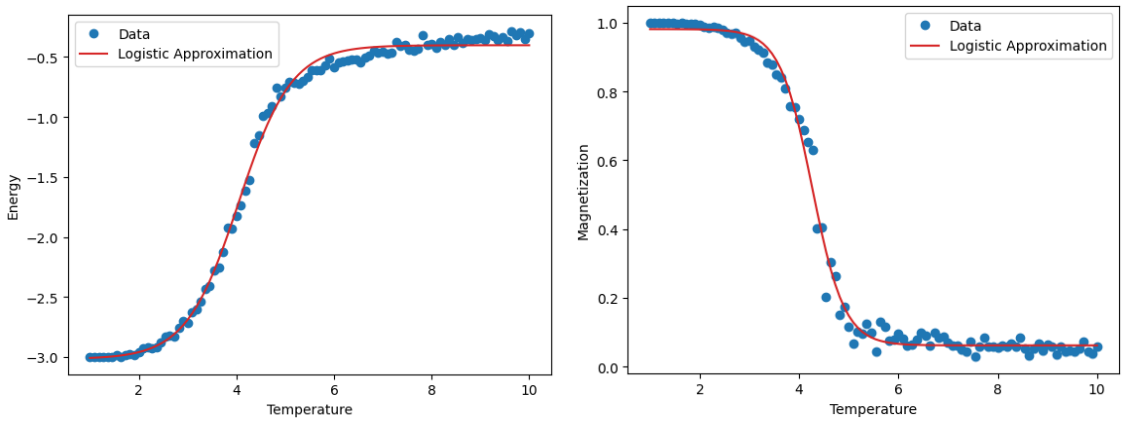

Figure 15: **Left**: See Figure 14 for data interpretation. The overlaid logistic curves minimize the RMSE between the curve and the data points.

by other sources estimate the critical temperature as $T_c = 4.51$ [3]. Therefore, the relative error using a logistic approximation on simulations with the Swendsen-Wang Algorithm for the 3-D Ising model is:

$$\frac{|T_c - x_B|}{T_c} = \frac{|4.51 - 4.27|}{4.51} = 0.0532 = 5.32\% \tag{4}$$

Because the relative error is low for both the 2-D case and the 3-D case, using a logistic approximation is a viable way to estimate the critical temperature of a spin system.

Alternative Update Methods: Wolff Algorithm

This section will assume terminology and methods from the previous section.

The Wolff Algorithm is an adjustment to the Swendsen-Wang algorithm that also uses spin clusters to find equilibrium states of the Ising model Markov Chain. However, this algorithm only involves flipping the spin of a single spin cluster at any given iteration [13]. As a result, it takes less time to perform a single update to a spin state, and it still avoids the critical slowing-down effect that poses a challenge to local state updates. However, the restriction of only updating one cluster per iteration decreases the convergence rate of the Wolff Algorithm as compared to the Swendsen-Wang Algorithm. So, for large systems and a high number of iterations, it may be more efficient to use the Wolff Algorithm, but it is not always the better option.

Identically to the Swendsen-Wang Algorithm, a bond forms (event B) between identical-spin particles i and j with probability $P(B|\sigma_i = \sigma_j) = 1 - \exp(-2J/T)$. If i and j do not have the same spin, then the probability that a bond forms between them is $P(B|\sigma_i \neq \sigma_j) = 0$ [13].

The Wolff Algorithm is as follows [13]:

1. Form an initial spin state, where each spin is +1 or −1 with probability $\frac{1}{2}$.

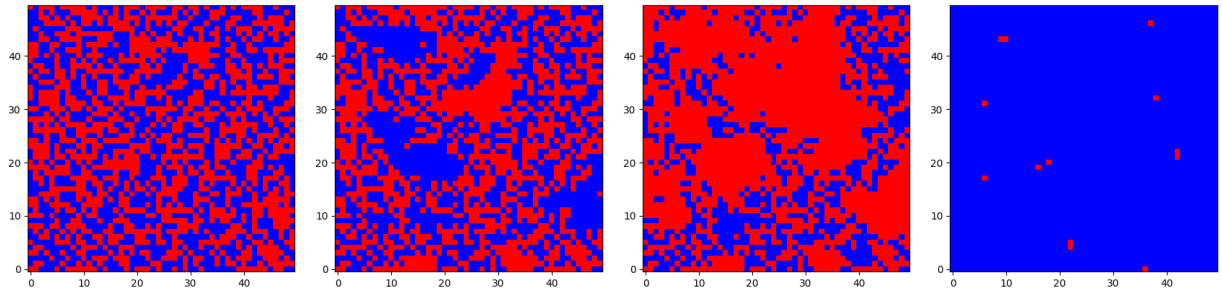

Figure 16: The 1st, 18th, 36th, and 54th iterations of the Wolff Algorithm for a 50×50 lattice of spins with $J = 1$ and $T = 1.5$. A blue cell represents a spin of +1 and a red cell represents a spin of −1.

2. Choose a uniformly random spin location within that state.

3. Find the spin cluster containing that spin by following the blacklist-connection method given in the Swendsen-Wang section.

4. Flip the sign of every spin within that spin cluster.

5. Using the resulting spin state from step 4 as the updated state, repeat steps 2, 3, and 4 until a pre-specified number of iterations is reached.

An example of the Wolff algorithm for a 2-D initial spin state is given in Figure 16. Note that this takes 54 iterations to converge to a magnetized state, whereas the Swendsen-Wang algorithm took only 7 iterations to converge from the same starting temperature.

The Wolff algorithm is extremely fast for spin states that do not converge to a magnetized state (i.e. initial states with a high temperature). However, the time complexity per iteration approaches the Swensden-Wang time complexity as the Wolff algorithm converges to a magnetized state. This is because most cells will be clustered together as a magnetized state is reached.

The magnetization and energy versus temperature plots for the 2-D case are given in Figure 17. This figure is similar to the one obtained for the Swendsen-Wang Algorithm.

For this 2-D case, the logistic approximation to the magnetization curve gives $x_B = 2.39$ as an estimate for the critical temperature at which a phase transition occurs. Because the analytical value for the critical temperature is $T_c = 2.269$, the relative error between the logistic approximation to the Wolff algorithm simulation of the 2-D Ising model is:

$$\frac{|T_c - x_B|}{T_c} = \frac{|2.269 - 2.39|}{2.269} = 0.0533 = 5.33\% \tag{5}$$

Performing the same analysis for the 3-D case gives Figures 18 and 19. The logistic approximation to the magnetization curve gives $x_B = 4.35$ as an estimate for the critical temperature at which a phase transition occurs. Because the numerically accepted value for

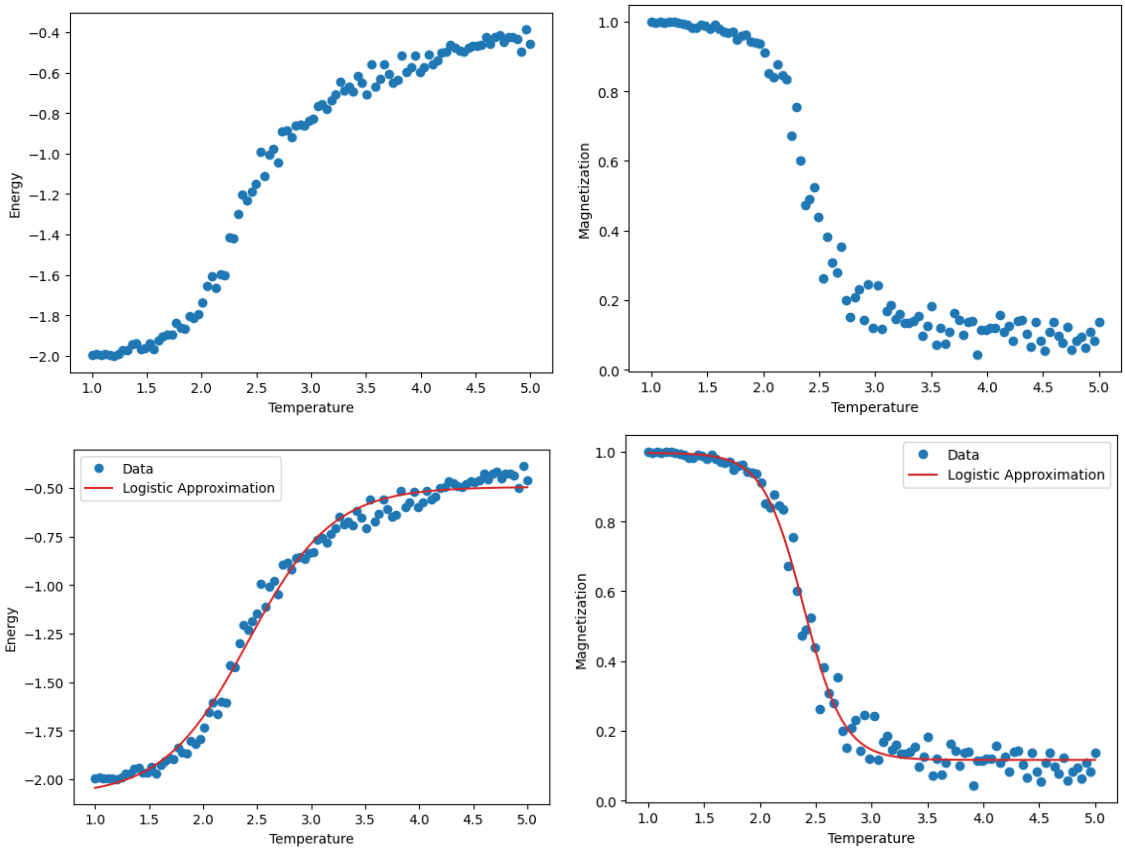

Figure 17: **Left**: The average of the final energies of 10 2-D spin states simulated at $N = 100$ temperatures ranging from 1 to 5. **Right**: The average of the magnitude of the final magnetizations of 10 2-D spin states simulated at $N = 100$ temperatures ranging from 1 to 5. Both plots consider the average measurement *per spin particle*.

the critical temperature is $T_c = 4.51$ [3], the relative error between the logistic approximation to the Wolff algorithm simulation of the 3-D Ising model is:

$$\frac{|T_c - x_B|}{T_c} = \frac{|4.51 - 4.35|}{4.51} = 0.0355 = 3.55\% \tag{6}$$

Just like the Swendsen-Wang Algorithm, the logistic approximation for the Wolff Algorithm gives a reasonable estimate for the critical temperature at which a phase transition occurs. The relative error between the known value and the estimated value is low for both the 2-D case and the 3-D case.

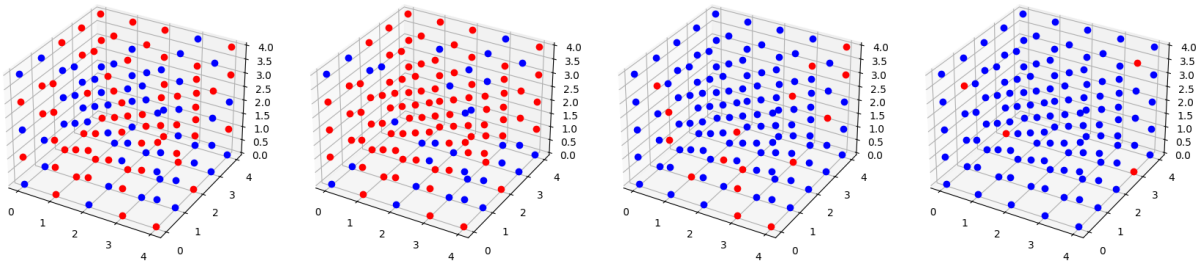

Figure 18: The first, third, fifth, and seventh states of the Wolff Algorithm for a $5 \times 5 \times 5$ lattice of spins with $J = 1$ and $T = 3$. A blue dot represents a spin of +1 and a red dot represents a spin of −1.

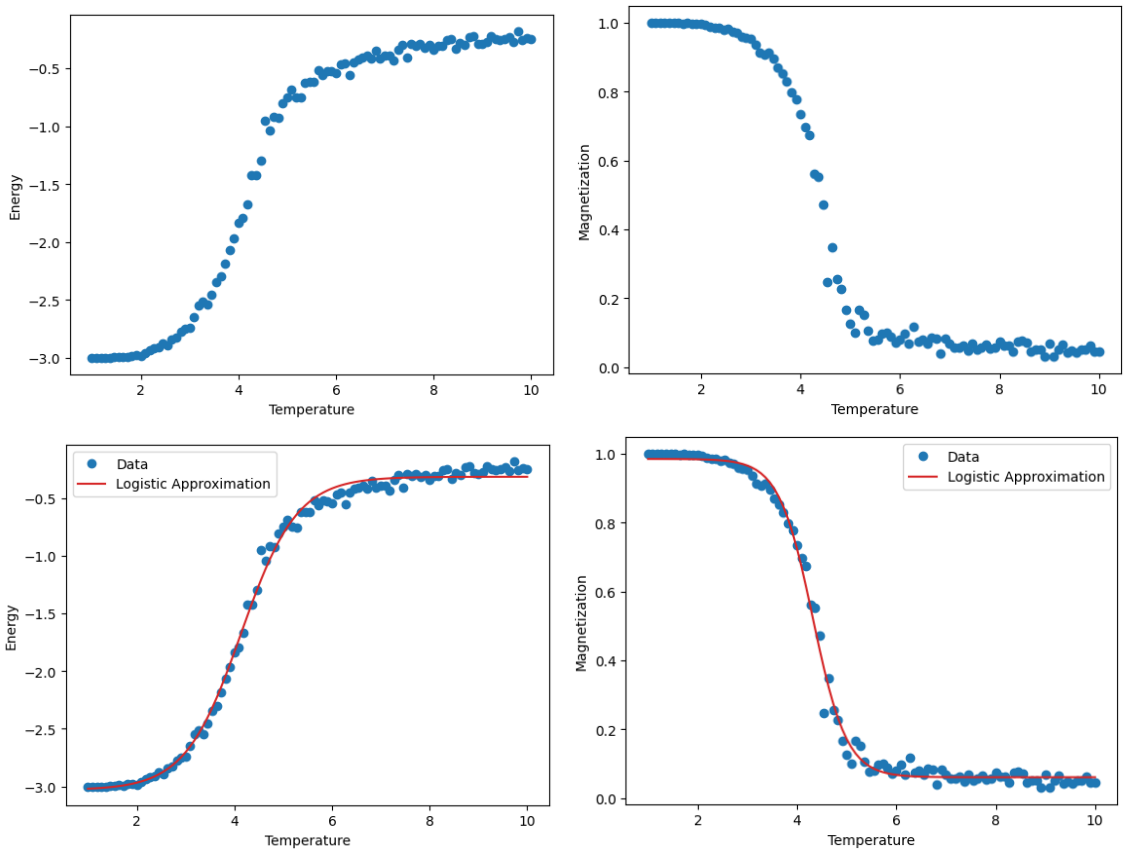

Figure 19: **Left**: The average of the final energies of 10 3-D spin states simulated at $N = 100$ temperatures ranging from 1 to 10. **Right**: The average of the magnitude of the final magnetizations of 10 2-D spin states simulated at $N = 100$ temperatures ranging from 1 to 10. All plots consider the average measurement *per spin particle*.

Physical Applications: Numerical Quantum Path Integration on a Lattice

As we've seen in the previous section, different algorithms for numerical simulation of Ising models can exploit the problem structure of the problem in arbitrarily complex ways for increased computational efficiency. However, while the physics of these idealized models are simple in concept, they can also be used within the context of intricate and compelling physical problems, such as Feynman path integrals [2]. The high level idea of quantum path integration starts with the classical Hamiltonian *principle of least action*.

An action S is a time integral over the Lagrangian L, which characterizes the state of the physical system. For us (and mostly in general) this Lagrangian is the difference between the kinetic and potential energy of a system. The principle works as such, say for a ball thrown in the air: while we may imagine any hypothetical path for the ball that we like, the true path is remarkably the one for which S is least.

One of Feynman's many contributions to science was to find this principle of least

action an incomplete approximation and only a limiting behavior in $\frac{S}{\hbar} \to \infty$ (where $\hbar$ is the reduced Planck's constant). *All* paths contribute to this calculation, while the one of least action simply contributes the most. Our goal for this section will be to simulate and sum over these trajectories for a quantum harmonic oscillator, which will use Ising lattice simulation via Metropolis-Hastings as a subroutine during numerical integration.

The harmonic oscillator with potential function $V(x) = \frac{1}{2}m\omega^2x^2$ has a ground state wave function analytically described by $|\psi_0|^2 = \sqrt{\frac{m\omega}{\pi}}e^{-m\omega x^2}$ [12], which will be our job to recover numerically. In the code, we set the constants m, ω to $m = \omega = 1$ for computational convenience.

Note that the wave function in quantum mechanics is a probability density function that describes the probability of observing a particle at a certain position, which is an integral. Before computing this, however, we will set up the problem as follows. The harmonic oscillator is a one-dimensional function of space whose trajectory we will simulate over time. We denote the starting and final positions in space-time as $a = (x_a, t_a)$ and $b = (x_b, t_b)$, respectively. In Figure 20 we show a few sample paths of these trajectories. As we can see by the areas of the trajectories that overlap, we expect our wave function to have mean zero. This is the classical trajectory, and observe this because we initialize our system with zero potential energy. All quantum paths should be a fluctuation about this constant path.

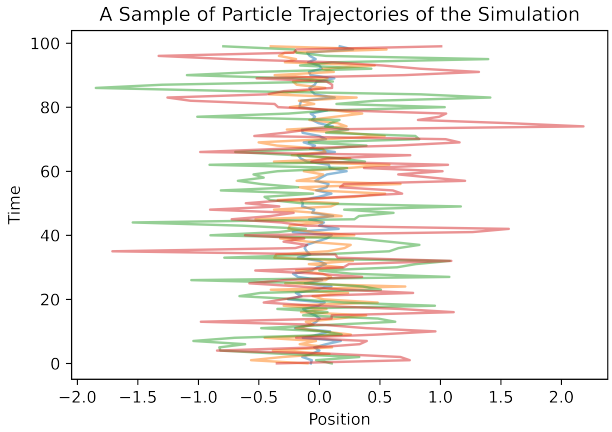

Figure 20: A few particle paths of the harmonic oscillator over time, imposing periodic boundary conditions such that $x_a = x_b$

As the theory for this problem setup is quite difficult, we will try to motivate our problem numerically where possible, which will require omitting some key steps in the derivations. First, we discretize the spatial coordinates into a grid of N equidistant points of size ε, or $\varepsilon N = t_b - t_a$. We use $j \in [N]$ to denote specific points along this time domain.

One of Feynman's most important mathematical observations was to link Green's space-time propagator function $G(x_a, t_a; x_b, t_b)$ to the action of a free particle (i.e. $V = 0$). The propagator is the integral over all paths between x_a and x_b, defined over our discretized

space as:

$$G(x,t;x_0,t_0) = \nsum dx_1 dx_2 \dots dx_{N-1} e^{iS[x,x_0]}$$

where $S[a,b]$ denotes the action of a path from a to b, and brackets denote that this a functional dependent on the path $x(t)$. Feynman's second important observation was to link this propagator with the wave function of a particle, where:

$$|\psi_n|^2 = \lim_{\tau\to\infty} e^{E_n\tau} G(x, t = -i\tau; x_0, 0)$$

with $E_n = n + \frac{1}{2}$ for $n \in \mathbb{N}$ for the case of the harmonic oscillator. While we have skipped over a few steps, we are almost there by the connection between the action and Hamiltonian H of a system, where $S[x_{j+1}, x_j] = -i\int_\tau^{\tau_{j+1}} H(\tau)d\tau$. The theoretical behavior in imaginary time allows us the compute the problem in entirely real coordinates:

$$G(x, -i\tau; x_0, 0) = \int dx_1 dx_2 \dots dx_n e^{-\int_0^\tau H(\tau)d\tau}$$

We now have an expression for which we can solve numerically. Denoting the energy at each point $j \in [N]$ by $U(x_j)$, we can obtain the integral over H by averaging over the energies by:

$$\int_0^\tau H(\tau)d\tau \approx \sum_j \varepsilon U_j$$

We also have an expression for this energy U_j, which is just the sum of the kinetic and potential energies. For the harmonic function, this involves discretizing a derivative $\frac{dx}{dt}$ and averaging over potential energies. For this, we use a central difference method and Simpson's rule, or:

$$U(x_j) = \sum_{j=1}^{N}\left[\frac{m}{2}\left(\frac{x_{j+1}-x_{j-1}}{2\varepsilon}\right)^2 + \frac{V(x_j) + 4V((x_j+x_{j-1})/2) + V(x_{j-1})}{6}\right]$$

In our experiments, we note that a simpler first order Euler scheme or linear average seemed to do just as well. This is likely because the harmonic function is fairly well behaved, as it is smooth (and, though technically not a restriction, the position values are statistically likely to be centered around zero, so the function is also basically Lipschitz). We can see that this calculation also requires us to discretize space as well as time, whose fineness we denote as δ. Though the average space step size is a fixed number, we introduce a small amount of randomness by drawing a uniform number in $[0,1]$ and multiplying the step size by its difference with 0.5. We find that this makes the path action calculations look slightly more "realistic" in the sense that they should appear to be a random walk.

To start the algorithm, we choose a random path along this lattice grid. All paths are

constrained to have periodic boundary conditions such that $x_a = x_b$. Each lattice point in the grid has an associated wave value function, which is the probability of finding the particle at this point in space. This probability is merely the sum of all times the point is visited divided by the total number of traversals from one point in the grid the the other. The wave function will therefore be obtained by creating a density plot over positions (and frequencies thereof) at each step in the simulation.

Each path also has an associated energy, which we calculate with the above scheme. This is now where Metropolis-Hastings is used in the computation. A random position x_j associated with time t_j is changed to a new position x'_j. For each coordinate that is changed, its change is weighted by the Boltzmann distribution with Metropolis-Hastings (M-H), where $\mathcal{P} = e^{-\varepsilon U} = e^{-U/k_B T}$ and k_B is Boltzmann's constant. The rejection/acceptance criteria for the move is then the same as the M-H step (i.e. the change in energy must be larger than some random number).

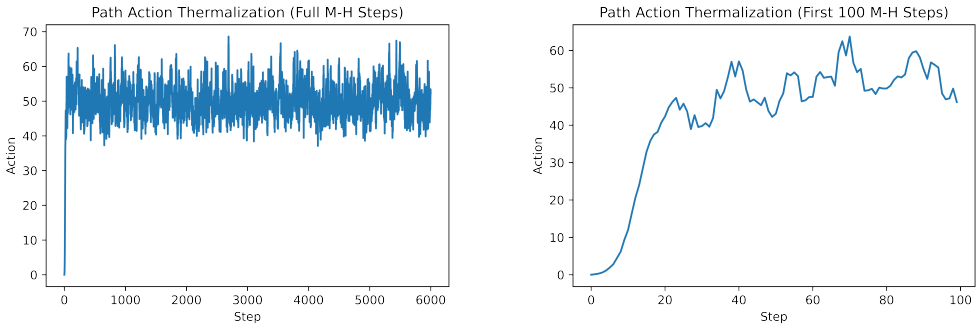

Figure 21: Path energies over Metropolis sweeps. These energies are averaged over $n = 50$ experiments, as trial-to-trial variability can be quite high. When we zoom in on the first 100 sweeps, we note that the system only takes about 20 to reach energy stabilization.

We plot the energies over the number of Metropolis sweeps ($k = 6000$) in Figure 21. Each sweep is essentially a path simulation over the discretization we set up previously, where each path has an associated energy. The process is averaged over a number over Metropolis-Hastings experiments as the energies can fluctuate quite a bit between experiments. We see that the thermalization stabilizes quite fast from a cold starting point with no energy in the system. After about only 20 sweeps, the system stabilizes at its asymptotic limit around $S = 50$. Thus, we can conclude that for this simple problem, Metropolis-Hastings quickly converges to the realistic physical system. Our next and final question will be to compare the rate of the simulation fidelity to the calculation of the actual wave function.

The wave function is the probability of finding a particle in a given state. Having done our numerical simulation to obtain the paths and energies, we can now calculate $|\psi_0|^2$ by building up a density plot in the paths over the number of sweeps. Below, we can see that the time to convergence in Metropolis-Hastings is not necessarily commensurate with the time to get a wave function that looks like the analytical solution. At iteration/sweep 50,

we are still quite far away from the solution. However, our algorithm eventually merges to the true solution in the limit, and the numerical integration succeeds. For such a simple algorithm, Metropolis-Hastings can be used in a variety of physical contexts.

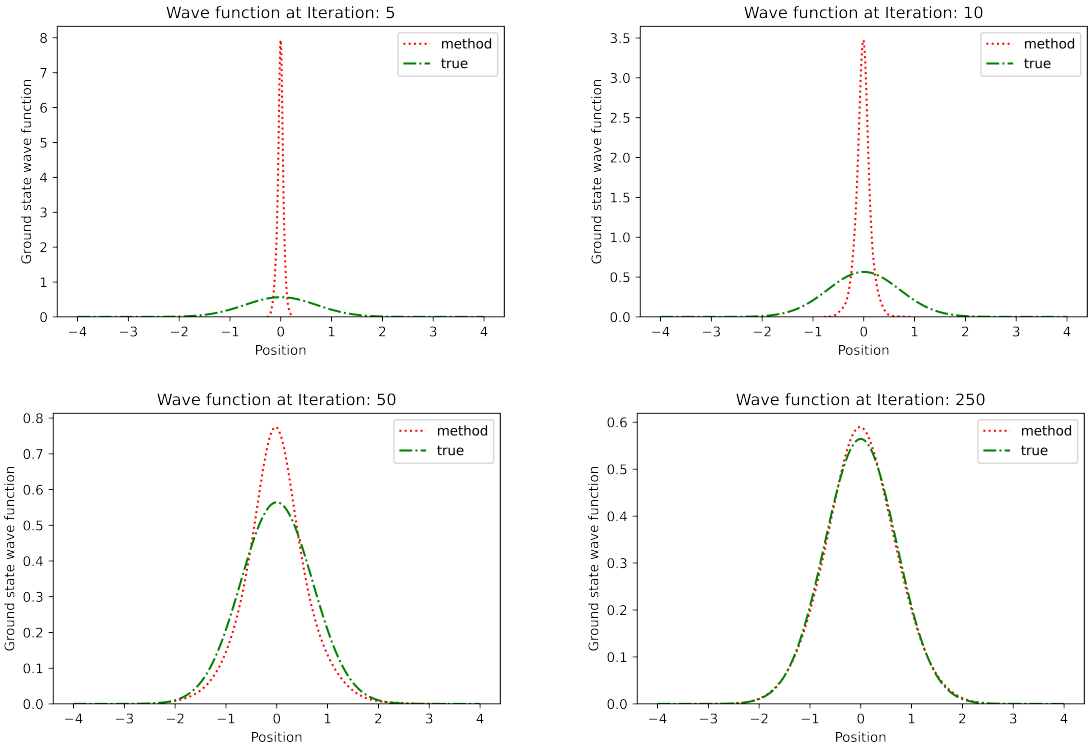

Figure 22: Even though the energies converge quite fast, this does not imply the same number of M-H sweeps to converge to the true wave function when $m = \omega = 1$

Conclusion

In this paper, we showed how to simulate 2-D and 3-D Ising models with the Metropolis-Hastings, Gibbs, Swendsen-Wang, and Wolff algorithms. We found that we could use regression to estimate the critical temperature of the spin system, which showed the viability and accuracy of each respective method. Furthermore, we showed that we were able to use Metropolis-Hastings to simulate a quantum wave function using numerical integration for a simple harmonic oscillator.

The algorithms presented in this paper face various limitations. Our implementations of cluster model algorithms did converge to expected values, but the random nature of Monte Carlo methods caused estimates for the critical temperature in both 2-D and 3-D cases to vary slightly between simulations. Additionally, fitting a logistic curve to data requires minimizing an extremely nonlinear function over four variables. It is possible that the Scipy minimizer did not find the global minimum for the RMSE, which would lead to

further discrepancies between simulations.

Another constraining factor of our analyses is that analytical solutions for the physics of the problems were sometimes lacking, and therefore the exactness of our methods could only be verified for specific, simple cases. Dimensionality is a fundamental consideration in choosing amongst Markov Chain Monte Carlo methods, and we were constrained by the capabilities and algorithmic complexities of these methods as we scaled the Ising model into different dimensions. For example, cluster methods are only sensible to use in dimensions higher than 1. In the case of our quantum numerical integration scheme, we had to limit the scope of our physical systems as even simple anharmonic oscillators have poorly understood analytical solutions. In numerical analysis, one of the most important objectives is to create sub-problems with recoverable solutions to weigh the benefits of various approaches. We see that, in practice, we might only be able to find a limited set of such sub-problems.

Overall, we were successful in demonstrating how statistical algorithms may be applied to simulate the evolution of Ising model spin states. We additionally showed how those same techniques may be used to visualize quantum wave functions. Interesting further directions of this work include extrapolating the Ising model into higher dimensions and modeling other, more complex quantum wave functions.

References

[1] Cristian C. Bordeianu, Manuel J. Paez, and Rubin H. Landau. Computational physics: Problem solving with computers. 2015.

[2] RP Feynman and AR Hibbs. Quantum mechanics and path integrals. *Journal of Neurophysiology*, 58, December 1965.

[3] B. Fierro, F. Bachmann, and E.E. Vogel. Phase transition in 2d and 3d ising model by time-series analysis. *Physica B: Condensed Matter*, 384, October 2006.

[4] Hills S. E. Racine-Poon A Gelfand, A. E. and A. F. M. Smith. Illustration of bayesian inference in normal data models using gibbs sampling. 1990.

[5] WK Hastings. Monte carlo sampling methods using markov chains and their application. 1970.

[6] JJ Hopfield. Neural networks and physical systems with emergent collective computational abilities. 1982.

[7] Erik Luijten. Introduction to cluster monte carlo algorithms. 2006.

[8] Dennis V. Perepelitsa. Path integrals in quantum mechanics. 2010.

[9] Satya Singh. The ising model: Brief introduction and its application. 2020.

[10] Ramesh Sridharan. The ising model and markov chain monte carlo.

[11] Robert H. Swendsen and Jian-Sheng Wang. Nonuniversal critical dynamics in monte carlo simulations. *Physical Review Letters*, 58(2), January 1987.

[12] Bruno Gimenez Umbert. Quantum mechanics by numerical simulation of path integral. 2017.

[13] Ulli Wolff. Collective monte carlo updating for spin systems. *Physical Review Letters*, 62(4), January 1989.