Monte Carlo Simulations & Phase Transitions

Ellis Benitez[1], Jakia Chowdhury[1], Owen Gemignani[1], Carter Rauch[1]

[1]*Department of Physics and Astronomy, Amherst College, MA*

Abstract

In this study we investigate Monte Carlo methods and phase transitions through random walks and lattice percolation, and how to utilize the Metropolis and Worm algorithms to calculate observables for the 2D Ising model. The random walks and percolation are primarily done to learn about Monte Carlo methods and phase transitions, focusing on observable such as mean squared displacement as a function of step count , and various observables for lattice percolation and clusters. When looking at the 2D Ising model we measure various observables with both Metropolis and Worm algorithms. We then compare their values to verify the functionality of our Worm algorithm. We find the worm algorithms functions more efficiently and avoids critical slowing down unlike the metropolis algorithm. The dynamical critical exponent is calculated for both algorithms: Metropolis: $z_{metro} = 1.856 \pm 0.063$, Worm: $z_{worm} = 0.833 \pm 0.030$.

Contents

1 Random Walks

A random walk is a mathematical model describing a path consisting of a sequence of random steps on a given space, such as a lattice. (More about random walk here.) In this study, we investigated several types of random walks, including non-reversing, self-avoiding, and biased walks in up to five dimensions. We explored various properties such as the relationship between the mean squared displacement, the number of loops, and the number of steps taken.

To simulate the behavior of random walkers we used Python's built-in random number generator, which is based on the *Mersenne Twister* algorithm. Each randomly generated number was used to assign a direction for the walker's next step. This highlights the importance of using a high-quality pseudo-random number generator to ensure the validity of such stochastic simulations. Various methods for evaluating the quality of random number generators, including the well-known Diehard tests, can be found here.

1.1 Basic Random Walk

An example of a two-dimensional random walk with 100 steps and step length 1 is shown in Figure 1. The simulation code for this example is available in the accompanying notebook [1].

We simulated multiple walkers (e.g., 100 walkers) for a range of step counts, from 10 to 1000 in increments of 10. Walkers, here and for the rest of the paper, represents number of trials. For each number of steps, we calculate the average squared distance from the origin for all walkers. The results for two dimensions are shown in Figure 2. The corresponding code and results for three dimensions are documented in [2].

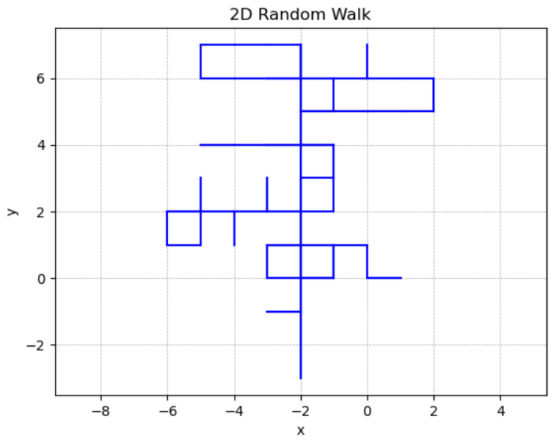


Figure 1: A 2D random walk trajectory with 100 steps and step length 1

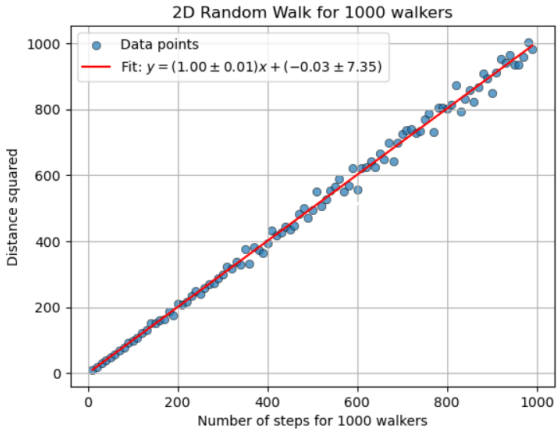


Figure 2: Mean squared displacement $\langle R^2 \rangle$ versus number of steps for a 2D random walk.

As predicted in [14], the mean squared displacement $\langle R^2 \rangle$ grows linearly with the number of steps t:

$$\langle R^2 \rangle \propto t$$

where R is the end-to-end distance of the walk. From the linear fits, the slope in 1D is found to be 1.000 ± 0.007, in 2D it is 1.000 ± 0.005, in 3D it is 1.000 ± 0.004, in 4D it is 0.999 ± 0.003, and in 5D it is 1.000 ± 0.003. All the analysis can be found here: [3]. We predict from these results that this linear relationship holds for higher dimensions as well.

1.1.1 Loops in Random Walks

We define a *loop* as a situation in which a walker returns to a previously visited site by forming a closed path that encloses a finite area. Furthermore, we do not include the re-tracing of loops as another loop. This definition aligns with visual aspect of loop counting, so when we plot a random walk trajectory, it is quite easy to double-check the output of the algorithm by counting the number of loops.

For instance, in the trajectory shown in Figure 1, eight loops are observed in 100 steps. When two loops occur side by side, we count them as separate loops and do not consider the larger combined loop that would be counted in Kirchhoff-style loop analyses.

To detect loops programmatically, we implement the following conditions:

1. The walker visits a coordinate that has already been visited.
2. The line segment corresponding to the current step has not been traversed before. So if the walker traces same loop more than once, it will still be counted as 1 loop.

The implementation of this loop-detection algorithm is provided in the notebook [4].

We counted the number of loops in 2D random walks and analyzed how it varies with the number of steps. Figure 3 shows an example of loop counting in a single trajectory. To generalize this observation, we computed the loop counts for walks ranging from 1000 to 5000 steps, with intervals of 100. The results are plotted in Figure 4, which clearly indicates a linear relationship between the number of loops and the number of steps. A best-fit line with slope approximately 0.1 supports the proportionality:

$$\text{Number of loops} \propto \text{Number of steps}$$

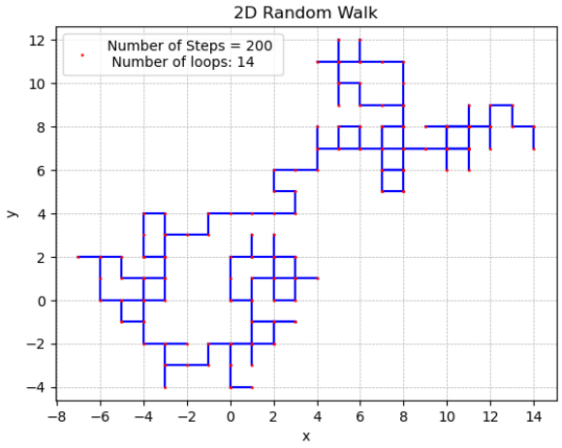


Figure 3: Loops in a 2D Random Walk

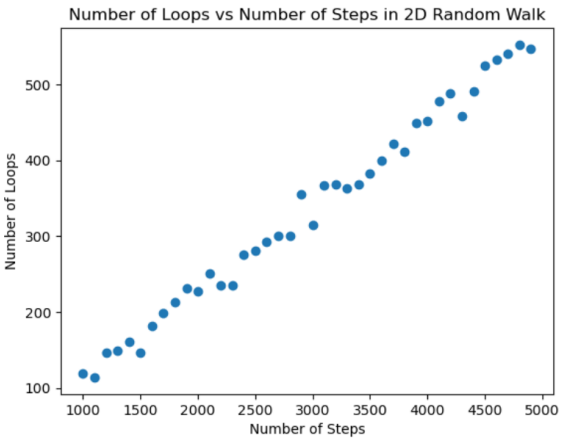


Figure 4: Number of Loops vs Number of Steps

1.1.2 Biased Random Walk

Building upon the basic random walk model, we introduced directional bias by assigning unequal probabilities to the possible directions of movement. For instance, in the two-dimensional biased walk, the probability of stepping in the positive x-direction was increased by reserving seven random values out of a total of ten for this direction, while each of the other directions (negative x, positive y, and negative y) was assigned one random value. So, the biased walk had a probability of 70% for moving in $+x$ direction and a probability of 10% for moving in the $-x, +y$, or $-y$ directions respectively.

As a result, the cluster of walker's origins shifts noticeably in the positive x-direction, and the shape of the cluster is compressed on the y axis, as illustrated in Figure 5. The left panel shows the trajectory of an unbiased 2D random walk, where the center remains around the origin. From running multiple trials we discovered that this shift had a positive relation with the strength of the bias. Notably, the cluster does not shift significantly in the y-direction, confirming that the bias is directional.

The code and simulation results for the biased walk are available in [3].

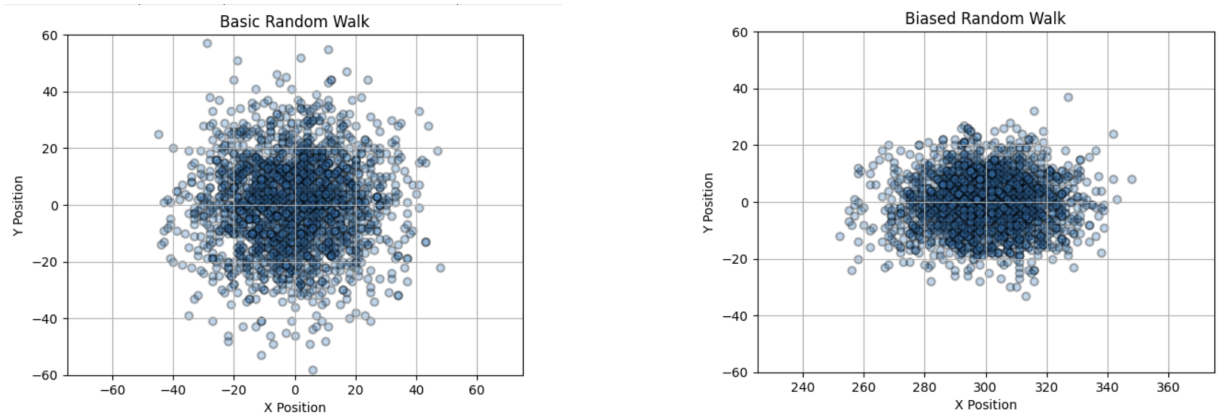


Figure 5: Comparison of unbiased (left) and biased (right) 2D random walks. This shows clear drift in the $+x$ direction.

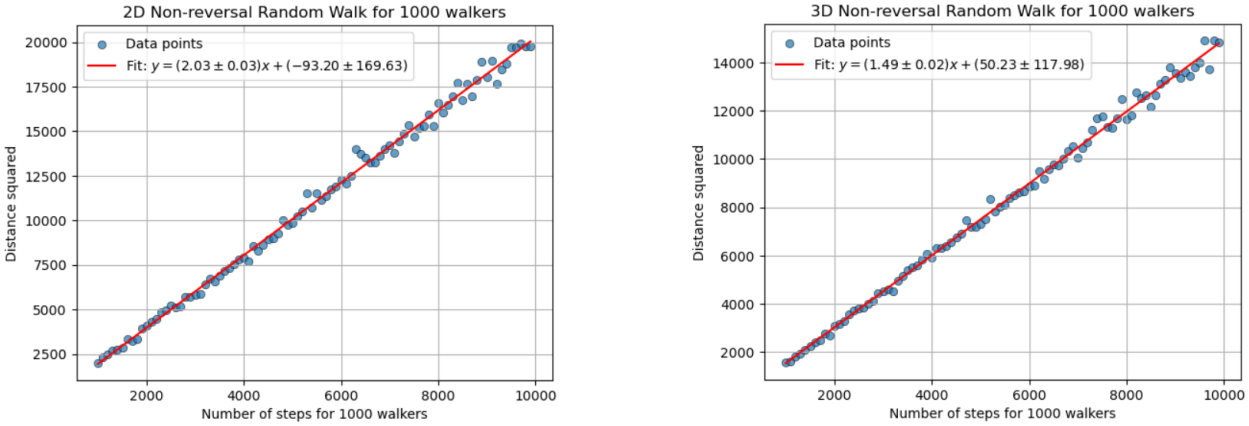


Figure 6: Mean squared displacement $\langle R^2 \rangle$ vs. number of steps in 2D (left) and 3D (right) non-reversing random walks.

1.2 Non-Reversing Random Walk

A non-reversing random walk is defined as one in which the walker is prohibited from taking a step that exactly reverses the previous step. For example, if the last move was in the $+y$ direction, the next move cannot be in the $-y$ direction. This constraint introduces memory into the system, making the walk non-Markovian.

Figure 44 presents an example of a two-dimensional non-reversing random walk. The corresponding simulation code is provided in [1]. The implementation eliminates the possibility of selecting the direction that is the exact inverse of the most recent step.

To quantify the effect of the non-reversing constraint, we conducted simulations for multiple walkers (e.g., 1000 walkers) over a wide range of steps, from 1000 to 10,000 in increments of 100. At each step count, we calculated the mean squared displacement $\langle R^2 \rangle$ from the origin. The simulation functions `many_non_rev2d()` and `many_non_rev3d()` were used for 2D and 3D analyses, respectively. Full details of the code and results are provided in [1].

The simulation results, presented in Figure 6, indicate that even under the non-reversing constraint, a linear relationship is maintained between the mean squared displacement and the number of steps:

$$\langle R^2 \rangle \propto t, \tag{1}$$

where R is the end-to-end distance and t is the number of steps. Linear regression yields slope values of 2.0331 ± 0.0281 for 2D and 1.4887 ± 0.0195 for 3D.

We extended this analysis to higher dimensions ($d = 4, 5$), with simulation code and results available in [5]. The slopes and associated uncertainties from the linear fits across different dimensions are summarized in Table 1.

Table 1: Linear Fit Parameters for Non-Reversing Random Walks in Various Dimensions

Dimension	**Slope (m)**	**Uncertainty**
1	1	0
2	2.0331	0.0281
3	1.4887	0.0195
4	1.34	0.02
5	1.25	0.01

These results are consistent with the results obtained from David Schaich's thesis [22], which is shown in table 45.

1.3 Self-Avoiding Walk

A self-avoiding walk (SAW) is a type of random walk in which the path is not allowed to intersect itself. In other words, the walker must avoid revisiting any previously occupied site. The walk is defined on a two-dimensional Cartesian plane and terminates when the walker is unable to proceed in any direction without retracing a previous step.

To implement this in code, each coordinate visited by the walker is stored in an array. Before selecting the next step, the program checks whether the new coordinate already exists in this array. If all possible neighboring coordinates are already visited, the walk halts. The detailed algorithm can be found in [1].

Representative examples of 2D self-avoiding walks are shown in Figure 7.

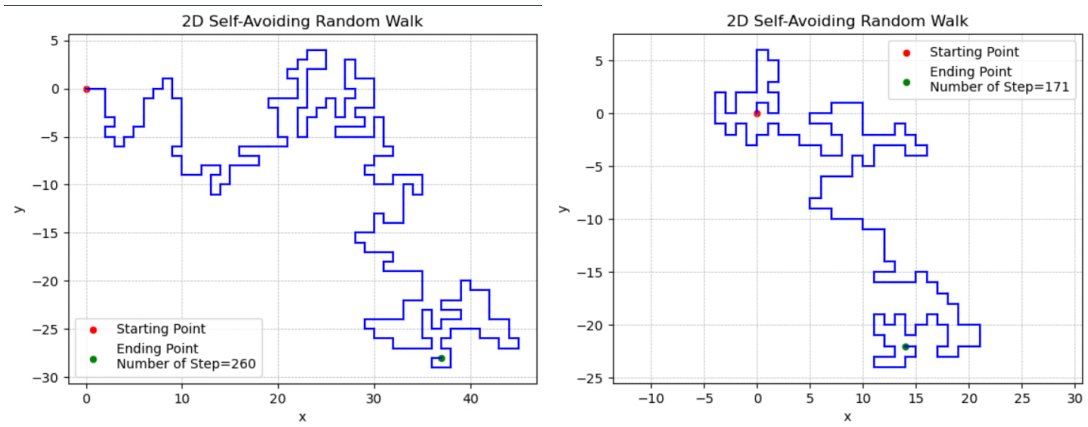


Figure 7: Examples of 2D self-avoiding random walks

As with non-reversal walks, we also computed the mean square end-to-end distance as a function of the number of steps for self-avoiding walk. The result for the 2D simulation is shown in Figure 8. Initially, the mean square distance increases with the maximum number of steps that walkers were allowed to take, but it quickly reaches a plateau, indicating that the walk is terminated due to the lack of available directions. In the two-dimensional case, this plateau is reached after approximately 100 steps, with the mean square distance saturating between 10 and 14.

The simulation code used to produce these results is available at [1], and additional implementations for higher-dimensional walks can be found in [5]. As expected, walks in higher dimensions tend to terminate later due to increased degrees of freedom. It is also noteworthy that no analytical solution has yet been found for this system.

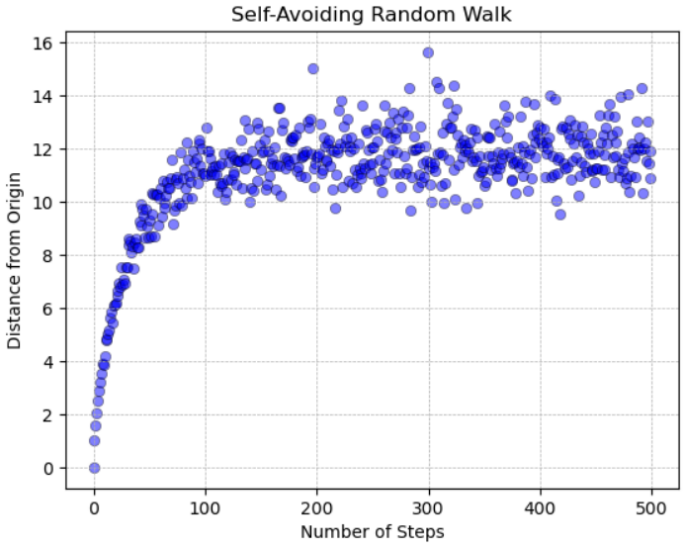


Figure 8: Mean square distance vs. number of steps for 2D self-avoiding random walk

2 Lattices

We consider a two-dimensional rectangular lattice, represented as a Cartesian grid in which each site is identified by integer coordinates (x, y). A lattice site is considered *occupied* (denoted by 1) if it is filled, and *unoccupied* (denoted by 0) otherwise. An example is shown in figure 9.

A path on the lattice is defined as a sequence of adjacent occupied sites. Two sites are considered adjacent if they share a side — that is, they are separated by a displacement of $(-1, 0)$, $(1, 0)$, $(0, 1)$, or $(0, -1)$.

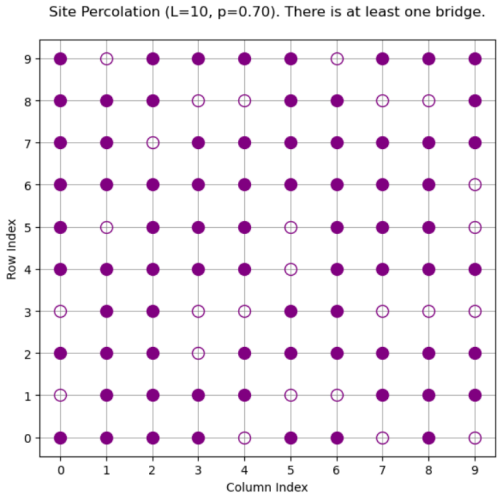


Figure 9: Example of a 10×10 2D lattice

Our objective is to determine whether a continuous path of adjacent occupied sites exists that connects the left edge of the lattice to the right edge. This phenomenon is known as *percolation*. The probability that such a path exists is referred to as the *percolation probability*, which depends on the occupation probability of individual sites. If a spanning path exists, the system is said to have undergone percolation or to be *reachable*.

2.1 Site Percolation and Clusters

To study this, we generated lattices where each site is occupied with a fixed probability p. This is known as *site percolation*. We simulated this system and computed the probability of a spanning path across the lattice. For this and most other simulation for lattice was performed by using a *Depth First Search (DFS)* algorithm. This algorithm works by picking a path and following it deeper and deeper. If the path hits a dead end, the path is backtracked to the last junction where there was another option. From that junction, an unexplored path is taken, and the deep exploration and backtracking is carried out until an end is found. More about DFS can be found here.

The function `is_reachable()` (which implements this algorithm) along with other codes can be found in [6]. An example of a generated lattice is shown in Figure 9, where filled circles denote occupied sites and hollow ones denote unoccupied sites. Predicted critical probability of percolation: $0.59274621 \pm 0.00000013$ according to Wikipedia.

2.1.1 Critical Probability for Percolation

We plotted the percolation probability as a function of the occupation probability and observed a threshold behavior, shown in Figure 10.

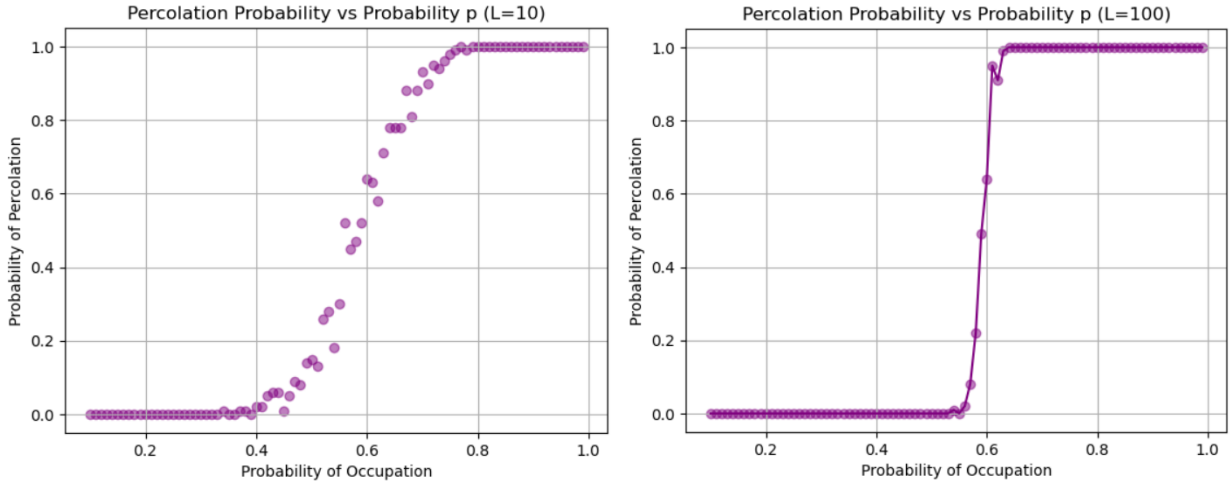


Figure 10: Percolation probability in 2D lattices of size $L = 10$ (left) and $L = 100$ (right). The transition from lower to higher percolation probability, which occurs at $p = 0.6$, is steeper for larger lattice.

Below a certain threshold probability, the percolation probability remains nearly zero. However, beyond this threshold, it abruptly increases to one — exhibiting a behavior similar to a step function. As shown in Figure 10, this critical probability is approximately 0.6 ± 0.1. The sharpness of this transition increases with lattice size. For instance, the transition in the $L = 10$ lattice (left panel) is more gradual compared to the sharper transition observed in the $L = 100$ lattice (right panel).

2.1.2 Cluster Counting

We also investigated how the number of clusters varies with occupation probability. The relevant code and detailed results are available in [6, 7]. A summary of our findings is shown in Figure 11.

Across all lattice sizes, the number of clusters follows a right-skewed Gaussian-like distribution, peaking near $p = 0.30 \pm 0.05$. As lattice size increases, the distribution becomes more well-defined, and the peak number of clusters also increases. Specifically, the peak values are approximately 14 ± 2 for $L = 10$, 55 ± 5 for $L = 20$, and 120 ± 10 for $L = 30$. We discovered that the peak value and the size of the lattice follow the following relation:

$$\text{Peak Value} \approx 0.1333 \times L^2$$

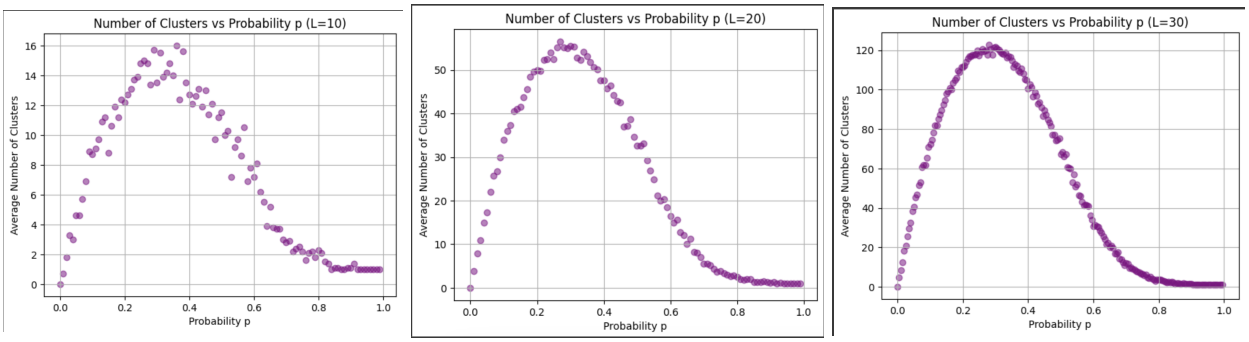


Figure 11: Average number of clusters vs. site occupation probability for lattice sizes $L = 10$, 20, and 30

These results make sense intuitively. For very low occupation probabilities, only a few isolated sites are occupied, resulting in a small number of clusters. Conversely, at very high occupation probabilities, the majority of sites belong to one large cluster, again yielding a small number of total clusters. Therefore, the number of distinct clusters peaks at intermediate values of p.

2.1.3 Cluster Size Distribution

Finally, we examined the distribution of cluster sizes at various occupation probabilities. The corresponding plots are shown in Figure 12, and the code used to generate these figures is available in [7].

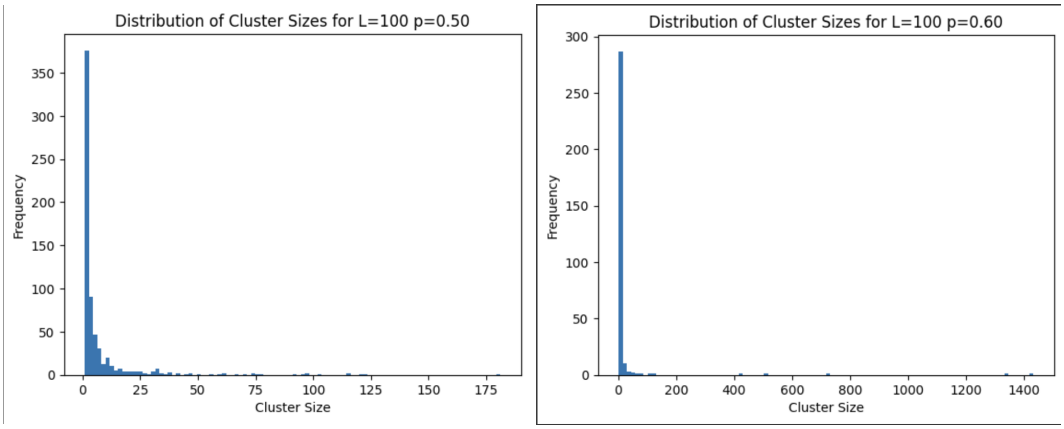


Figure 12: Cluster size distributions for lattice size L=100 with occupation probabilities $p = 0.50$ and 0.60

As the occupation probability increases, a few large clusters emerge, while the number of smaller clusters continues to grow. The distribution becomes increasingly skewed, reflecting the dominance of large connected components amid a background of smaller isolated clusters.

2.2 Anomalous Diffusion

Anomalous diffusion uses both the concepts of random walk and cluster in a lattice. An occupied site is randomly selected from a randomly generated 2D square lattice with occupation probability 0.52. A random walk is then initiated from that occupied site with the condition that the walk is confined within the cluster where the occupied site belongs. This requires that the walker is only able to travel between adjacent "occupied" sites.

We simulate an anomalous diffusion with 100 walkers, 1500 steps and 50x50 lattice and calculated the mean squared displacement $\langle R^2 \rangle$ as a function of number of steps. The result is shown in figure 13. Some large deviation in the data is noticed with increasing number of steps, which seems to be consistent with

lattice size. However, there is a positive relation between distance and number of steps:

$$\langle R^2 \rangle \sim t^x$$

Then by plotting and exponential fit in figure 13 we observed that $x = 0.28 \pm 0.02$.

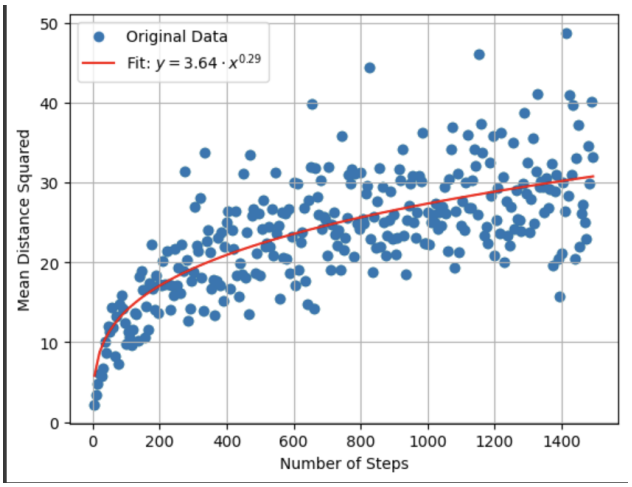


Figure 13: Mean squared displacement of a Anomalous Walk vs. Number of Steps Entered for a 50x50 2D Square Lattice With Fitted Exponential Line and Corresponding Equation

3 Introduction to Ising Model

The Ising model is a foundational statistical physics model used to study phase transitions and magnetic behavior in materials. It consists of a lattice where each site hosts a spin that can be in one of two states: up (+1) or down (−1), akin to tiny magnetic dipoles. The total energy of the system follows a Boltzmann distribution, with interactions limited to nearest neighbors. The fact that the state of the system depends only on the local interaction i.e. the state of the nearest neighbor makes it a simple but very useful model.

3.1 Simulation of Metropolis Algorithm

The implementation of the simulation is available in [8]. The system is initialized using the function `initialize_lattice(L)`, which generates a random $L \times L$ lattice with spins randomly set to either $+1$ or -1.

The energy difference resulting from a proposed spin flip at site (i, j) is calculated using the function `compute_energy_difference(lattice, i, j)`, based on the following expression:

$$\Delta H = 2 s_i J \sum_{\text{nn}} s_j \tag{2}$$

where:

- s_i is the spin at site (i, j),
- s_j denotes the neighboring spins of s_i,
- J is the coupling constant ($J > 0$ for ferromagnetic, $J < 0$ for antiferromagnetic),
- k_B is the Boltzmann constant,
- T is the system temperature.

The interaction energy depends on the coupling constant J, and two main cases arise:

- **Ferromagnetic** ($J > 0$): Neighboring spins tend to align, minimizing energy when spins are parallel.
- **Antiferromagnetic** ($J < 0$): Neighboring spins tend to anti-align, minimizing energy when spins are antiparallel.

These interaction preferences lead to probabilistic spin flips as the system evolves toward lower energy configurations.

The lattice evolves via the Metropolis algorithm implemented in `metropolis_sweep(lattice, T)`, which iterates over each site, calculates ΔH using Equation 2, and probabilistically flips spins based on:

$$P_{flip} = \begin{cases} e^{-\Delta H / k_B T}, & \Delta H > 0 \\ 1, & \Delta H \leq 0 \end{cases} \tag{3}$$

Edge effects are addressed using periodic boundary conditions (wrap-around).
The average energy and magnetization per spin are computed using `calculate_energy(lattice)` and `calculate_magnetization(lattice)`, respectively. The total energy is derived from the Hamiltonian:

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} s_i s_j \tag{4}$$

where the sum is over all nearest-neighbor spin pairs. The energy per spin is then given by H/L^2.

The mean magnetization is calculated as the average spin value and ranges between -1 and 1.

The simulation proceeds for a total of `mcs_max` Monte Carlo steps, and we track the evolution of the energy and magnetization over number of steps. One Monte Carlo step is defined as sweeping over every site of the lattice and checking whether it flips the spin depending on the Hamiltonian. Final visualizations are generated using `plot_results()`.

3.2 Results and Discussion

We conducted simulations on a lattice of size $L = 100$ for $mcs\_max = 1000$ Monte Carlo steps at three different temperatures: below, at, and above the critical temperature T_c. The critical temperature for the 2D square lattice is analytically known (see [17]) and approximated as $T_c \approx 2.27$. For simplicity, we set $J = k_B$, placing the system in the ferromagnetic regime.

The simulation results are presented in Figures 14, 15, and 16.

3.2.1 Low Temperature Regime ($T \ll T_c$)

At low temperatures (e.g., $T = 1$), the system rapidly transitions from a random configuration to an ordered state:

- Spins strongly align, forming large domains or a fully magnetized state ($+1$ or -1).
- Magnetization converges close to ± 1, indicating high order.
- The energy decreases rapidly and stabilizes, as most neighboring spins become aligned.

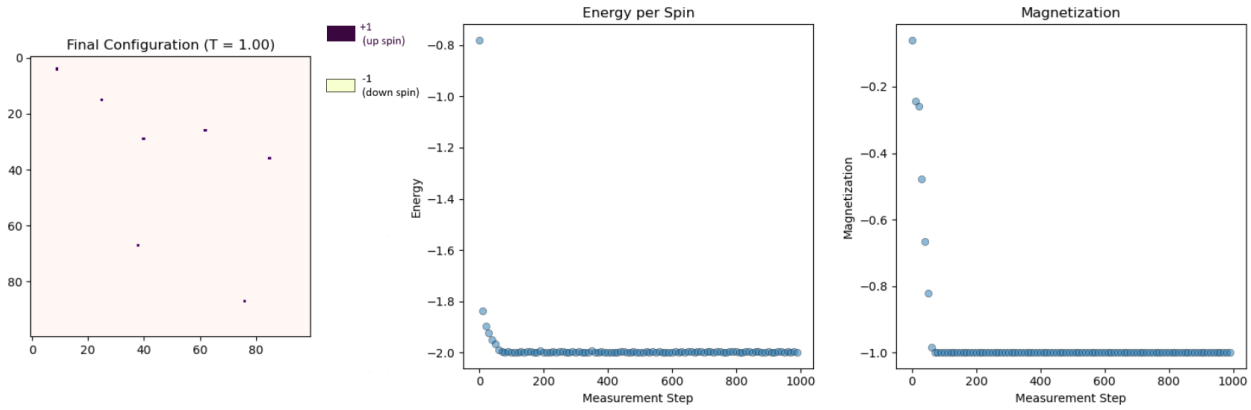


Figure 14: Final configuration and the evolution of energy and magnetization for L=100 $T = 1$. For Final Configuration up and down spin are binary such that 1 is up while -1 is down and there is no in between

3.2.2 Critical Temperature Regime ($T \approx T_c$)

At $T = T_c = 2.27$, the system exhibits critical behavior:

- Large clusters of aligned spins form and disintegrate dynamically.
- Spin configurations resemble fractal-like patterns with no global alignment.
- Both energy and magnetization fluctuate significantly, showing no clear tendency toward complete order or disorder.

3.2.3 High Temperature Regime ($T \gg T_c$)

At high temperatures (e.g., $T = 3$), thermal agitation dominates the spin dynamics:

- Spins remain mostly random; no long-range magnetic ordering is observed.
- Energy remains relatively high, as many neighboring spins are misaligned.
- Magnetization fluctuates around zero, consistent with equal numbers of up and down spins.

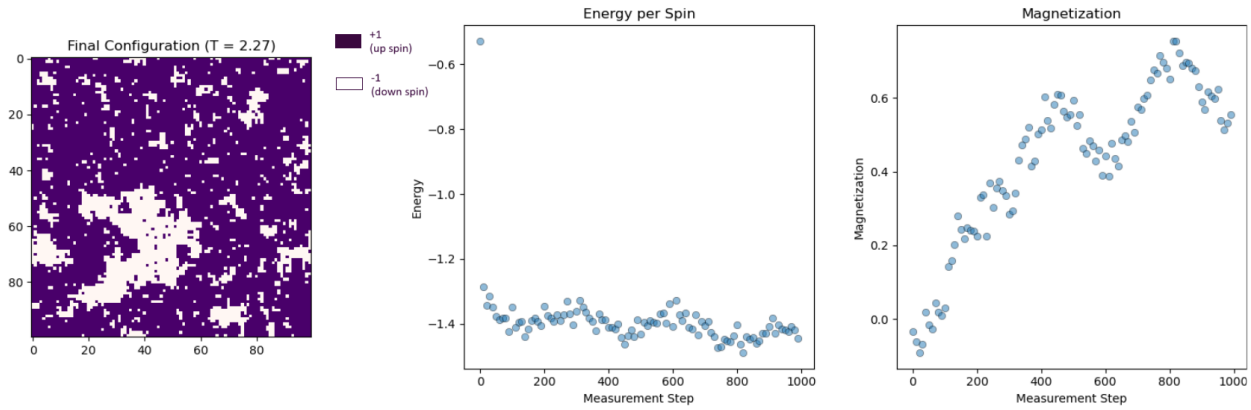


Figure 15: Final configuration and the evolution of energy and magnetization for $T = 2.27$

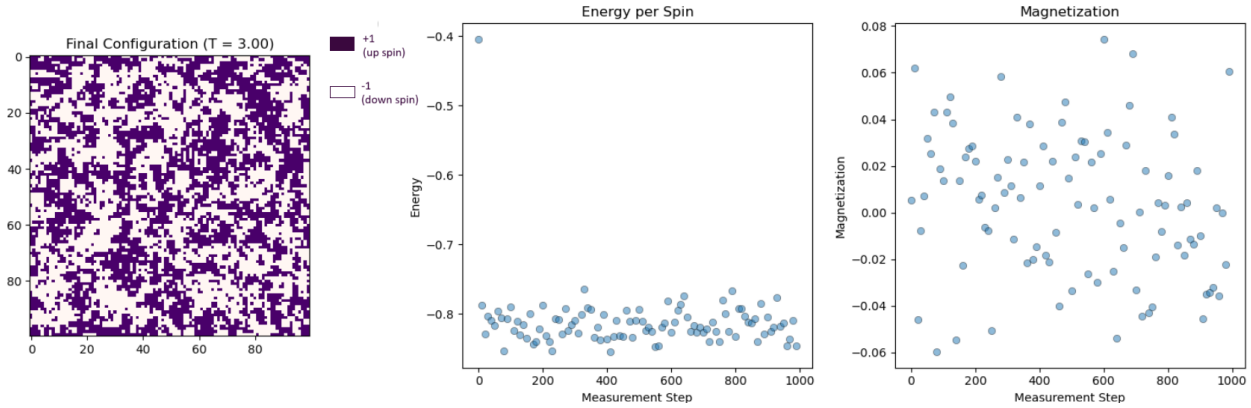


Figure 16: Final configuration and the evolution of energy and magnetization for $T = 3$

4 Phase Transition

As the temperature increases, the Ising lattice undergoes a transition from an ordered phase—where spins are predominantly aligned—to a disordered phase characterized by random spin orientations. This transformation, known as a phase transition, occurs near a characteristic point called the *critical temperature* (T_C). For the two-dimensional square lattice Ising model, the theoretical critical temperature is given by $T_C \approx 2.269$ [17]. Near this temperature, the system exhibits large-scale fluctuations, forming extended clusters of aligned spins, as illustrated in the previous sections.

Phase transitions are of fundamental importance in statistical physics. They help us understand how collective behavior emerges from microscopic interactions, how criticality leads to universal properties across seemingly different systems, and how physical observables diverge or behave non-trivially near the transition point.

To investigate the phase transition and determine T_C more precisely, we study a variety of physical quantities that reflect the macroscopic behavior of the system:

1. Energy
2. Magnetization
3. Binder Cumulant
4. Cluster Size
5. Two-Point Correlation Function
6. Specific Heat

7. Magnetic Susceptibility

Finally, to connect the behavior observed in finite lattices to the thermodynamic limit, we perform a Finite Size Scaling analysis.

5 Thermalization (Metropolis)

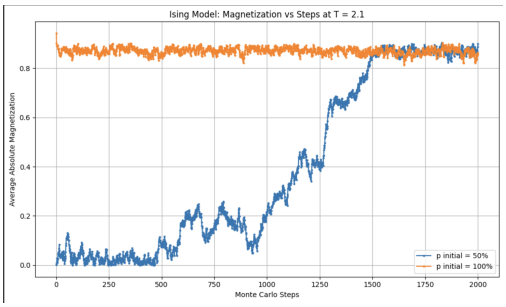
5.1 Introduction

Before we examine the behavior of a lattice at a specific temperature, we need to make sure that the lattice has reached temperature equilibrium. Thermalization refers to the step time that a lattice initialized with any spin configuration needs to reach its equilibrium for any given temperature.

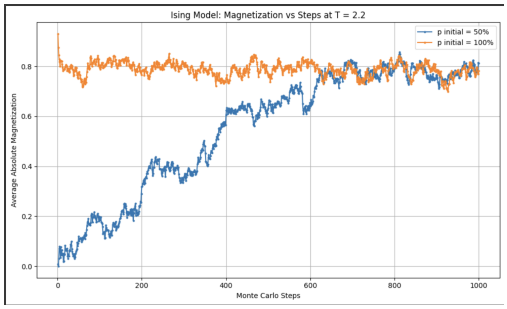
5.2 Results

To see this effect in action, and to see how many thermalization steps are necessary for measurements of a L sized lattice a T temperature we plotted average magnetization vs steps as in figure 17. Here, we became aware of a difference in lattice initialization.Two potential choices for lattice initialization are either one that is initialized with a probability of 50% up spin and 50% down spin at each site on the lattice, or one that has all the same spin across the lattice. Looking at our figures, we decided that initializing the lattice with all the same spin- referred to as p = 100% and in orange on our figures- was more efficient because it generally needed fewer steps to thermalize.

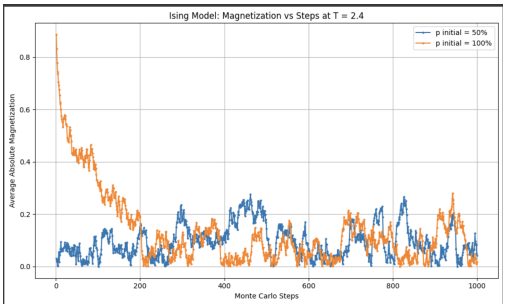
It is important to note that as temperature approaches critical temperature, the number of thermalization steps dramatically increases. Therefore, by plotting thermalization times near the critical temperature it allowed us to vary thermalization steps in our measurements so that, around the critical temperature we took many steps, but as we moved farther away we were able to take much fewer steps. This whole methodology served as one factor in making our code more efficient, as we had to take fewer total step measurements. The code for these figures can be found here [9].



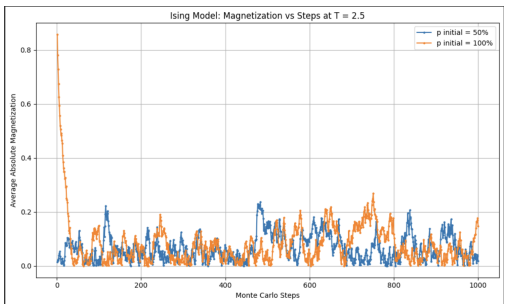
(a)



(b)



(c)



(d)

Figure 17: Magnetization vs Temps for T = 2.1 , 2.2, 2.4,2.5. Orange line is p = 1 initial lattice and blue is p =0.5

6 Autocorrelation (Metropolis)

Before we start sampling thermodynamic quantities such as energy, magnetization, specific heat, etc., we need to make sure that two consecutive measurements are statistically independent. Here comes the auto-correlation function. In Monte Carlo simulations, the *autocorrelation function* quantifies the statistical dependence between measurements separated by steps (equivalent to time for real world measurement). For instance, it assesses how strongly a measurement of magnetization $m(t)$ is correlated with its value at a later step $t + \Delta t$.

- **High autocorrelation:** Indicates that the system evolves slowly, requiring more steps to generate statistically independent configurations.
- **Low autocorrelation:** Suggests efficient exploration of phase space and rapid decorrelation between configurations.

Mathematical Definition

Given a time series $m(t)$, such as magnetization at each Monte Carlo step, the autocorrelation function $C(t)$ is defined as:

$$C(t) = \frac{\langle m(t') \cdot m(t + t') \rangle - \langle m \rangle^2}{\langle m^2 \rangle - \langle m \rangle^2} \tag{5}$$

where $\langle \cdot \rangle$ denotes an average over time t'.

In many systems, $C(t)$ exhibits exponential decay:

$$C(t) = e^{-t/\tau} \tag{6}$$

where τ is the *exponential autocorrelation time*, characterizing how quickly correlations diminish.

6.1 Algorithm

The implementation for this section is available at [8].

Generating the Magnetization Time Series

The function `get_m_t(lattice)` computes magnetization after each Monte Carlo step, generating a time series:

$$\{m(t)\} = [m(1), m(2), \ldots, m(\texttt{mcs\_max})]$$

Each magnetization value is defined as:

$$m = \frac{1}{N} \sum_{i=1}^{N} s_i, \quad m \in [-1, 1]$$

Computing the Autocorrelation Function

Using the magnetization time series, the function `autocor_direct(m_t, t_max=100)` computes $C(t)$ using Equation 5 and returns an array of length `t_max`. An alternative and more efficient implementation using the Fast Fourier Transform is available through the function `autocor_fft(m_t, t_max=100)`.

Estimating the Autocorrelation Time

The integrated autocorrelation time τ is estimated by integrating the computed autocorrelation function:

$$\tau_{\text{int}} = \frac{1}{2} + \sum_{\tau=1}^{\tau_{\max}} C(\tau) \tag{7}$$

This is implemented using the function `integrated_autocorr_time(autocorr)`, described in detail in Section 6.3.

6.2 Results

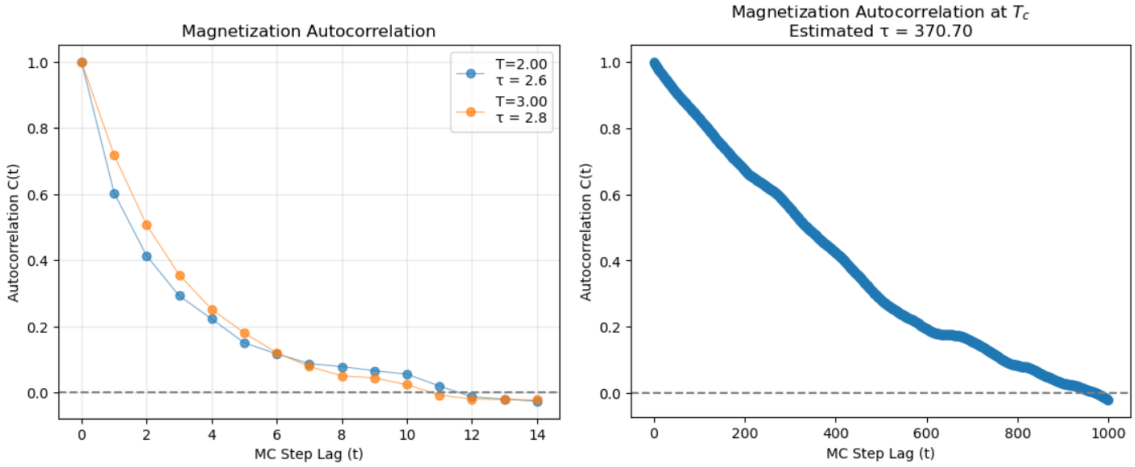
Figure 18 (left) shows the autocorrelation function for a 50×50 lattice at two temperatures: $T = 2.0$ (below critical temperature) and $T = 3.0$ (above critical temperature). Simulations were performed using 5000 Monte Carlo steps with 1000 steps for thermalization. In both cases, the autocorrelation decays exponentially, yielding autocorrelation times $\tau \approx 2.8$ for $T = 2.0$ and $\tau \approx 2.6$ for $T = 3.0$.

These estimates can be cross-verified using the half-life method. From Equation 6, setting $C(t_{1/2}) = \frac{1}{2}$ leads to:

$$\frac{1}{2} = e^{-t_{1/2}/\tau} \quad \Rightarrow \quad \tau = \frac{t_{1/2}}{\ln 2}$$

Using this relation, $t_{1/2} \approx 2$ for $T = 2.0$ implies $\tau \approx 2.8$, and $t_{1/2} \approx 1.8$ for $T = 3.0$ yields $\tau \approx 2.6$, consistent with previous estimates.

Figure 18 (right) shows the behavior at $T = T_c \approx 2.296$, where the decay no longer follows an exponential trend, suggesting a slower decay — potentially power-law. This behavior corresponds to a significantly increased autocorrelation time, reflecting the phenomenon of critical slowing down.



(a) (b)

Figure 18: Autocorrelation function for $L = 50$ at $T = 2.0$ and $T = 3.0$ (a) and at $T = 2.296$ (b).

6.3 Integrated Autocorrelation Time

The *integrated autocorrelation time* τ_{int} serves as a practical metric for determining an appropriate sampling interval. To ensure statistical independence between measurements, one should sample every $2\tau_{\text{int}}$ steps. Integrated auto-correlation time is mathematically defined in equation 7 and the implementation can be found here [9].

Furthermore, τ_{int} provides insight into the system's behavior near criticality. As the temperature approaches the critical point, the autocorrelation time increases significantly — a phenomenon known as *critical slowing down*. This is particularly evident in systems evolving under local update rules, such as the Metropolis algorithm.

Figure 19 illustrates τ_{int} as a function of temperature for a 64×64 lattice. The peak near T_c highlights the sharp increase in correlation time at criticality.

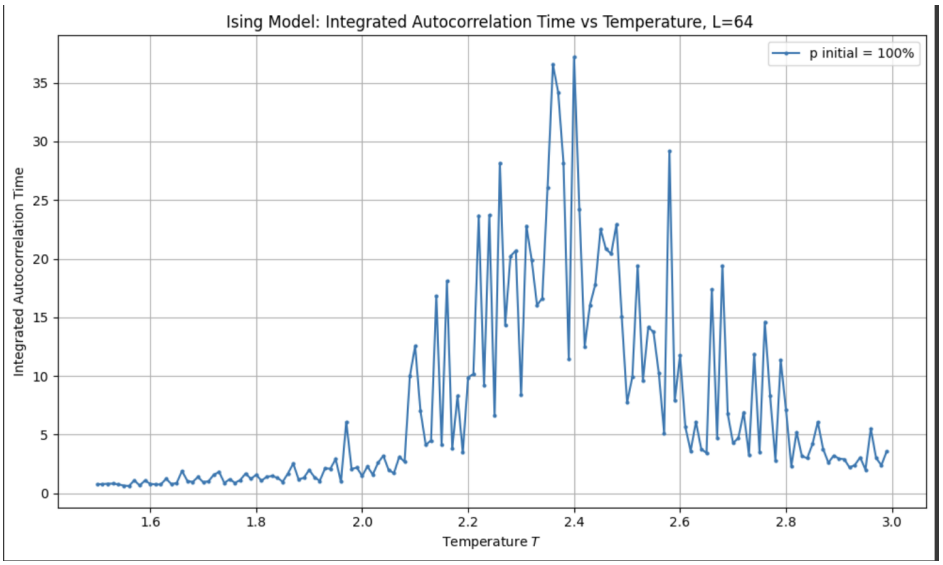


Figure 19: Integrated autocorrelation time τ_{int} vs. temperature for $L = 64$.

7 Energy (Metropolis)

Finding the energy of the system is a key component of the Metropolis algorithm and also allows for an estimation of the critical temperature. Specifically, the Metropolis Algorithm uses change in energy see equation 3. Furthermore energy or the Ising Model Hamiltonian is seen in equation 4. From that we can derive an equation for average energy per site

$$\langle \mathcal{H} \rangle = \frac{1}{2L^2} \sum_{i}^{L^2} (-J \sum_{\langle i,j \rangle} s_i s_j) \tag{8}$$

Using this equation to graph average energy per site vs temperature, as in figure 20, we see a relationship very similar to magnetization, where at the critical temperature there is an inflection point. Then, to be able to graph this relationship, for each temperature we let a lattice thermalize for 100 steps and then took 1000 steps with sample intervals every 10 steps.

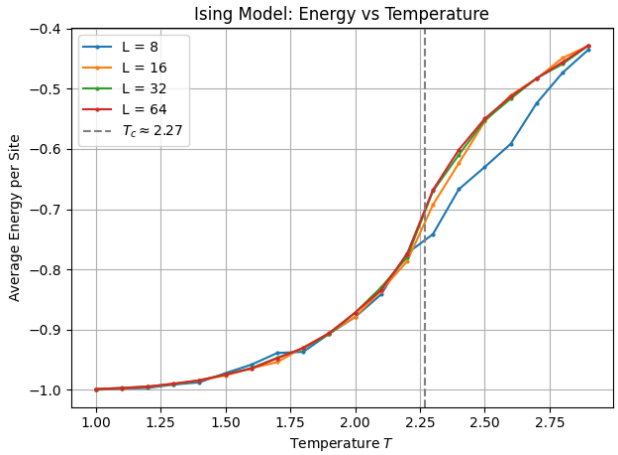


Figure 20: Energy vs Temperature

8 Magnetization (Metropolis)

In the two-dimensional Ising Model, magnetization describes the overall alignment of spins the the system. Each spin on the lattice can either be in the states up (+1) or down (-1). The average magnetization per spin is given by:

$$\langle m \rangle = \frac{1}{L^2} \langle |M| \rangle = \frac{1}{L^2} \langle | \sum_{i=1}^{N} s_i | \rangle \tag{9}$$

Where $s_i = \pm 1$ is the spin at a site i and N is the total spins.

If most spins are aligned in the same direction, the value of m will be near $+1$ or -1, which will indicate a high degree of order (a magnetized state). If the spins are randomly arranged (like in high temperatures), the magnetization tends to fluctuate around zero, meaning the system is in a disordered, non-magnetized state. Magnetization is therefore an important quantity for understanding how the system behaves, especially near the critical temperature, where a phase transition occurs between ordered and disordered phases.

8.1 Results

The trends, which can be observed in Figure 21, reflect the predicted behavior of the Ising model near the critical temperature. At low temperatures ($T < T_c$), the spins are in a ferromagnetic phase, where spins align and m fluctuates near one. As the temperatures increase beyond the critical temperature ($T > T_c$), thermal fluctuations dominate, causing the spins to become more randomly oriented. In this high-temperature, disordered phase, the magnetization approaches zero. At critical temperature ($T = T_c$), the system goes through a phase transition, where magnetization drops sharply. Near this point, the system is highly sensitive to fluctuations and can shift between ordered and disordered states.

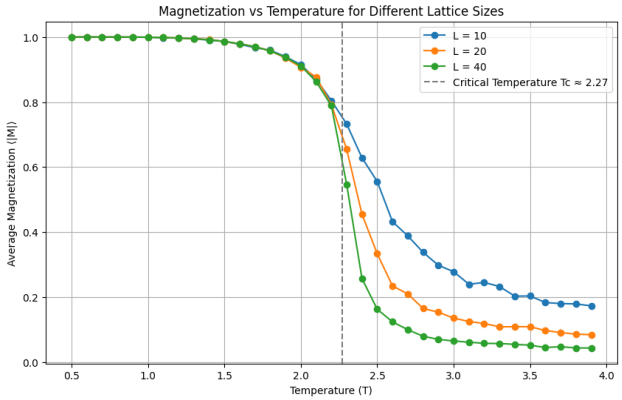


Figure 21: Magnetization vs Temperature for Different Lattice Sizes

8.2 Algorithm

The figure in this section is from the Ising Model magnetization program[10]. The parameters are: 300 Steps, 5 Runs, 20 Magnetization Frequency (Samples).

9 Binder Cumulant (Metropolis)

In Monte Carlo simulations of magnetic systems, calculating the average magnetization can be misleading near the critical temperature due to spontaneous symmetry breaking. As the system transitions between magnetization states (e.g., from $+m$ to $-m$), the average magnetization tends to zero, obscuring the presence

of order. To address this, Binder introduced a dimensionless quantity, *Binder cumulant*, which provides a more reliable indicator of critical behavior [14]. The Binder cumulant is defined as:

$$U_L = 1 - \frac{\langle m^4 \rangle_L}{3\langle m^2 \rangle_L^2} \tag{10}$$

where $\langle \cdot \rangle_L$ denotes ensemble averages over a system of size $L \times L$.

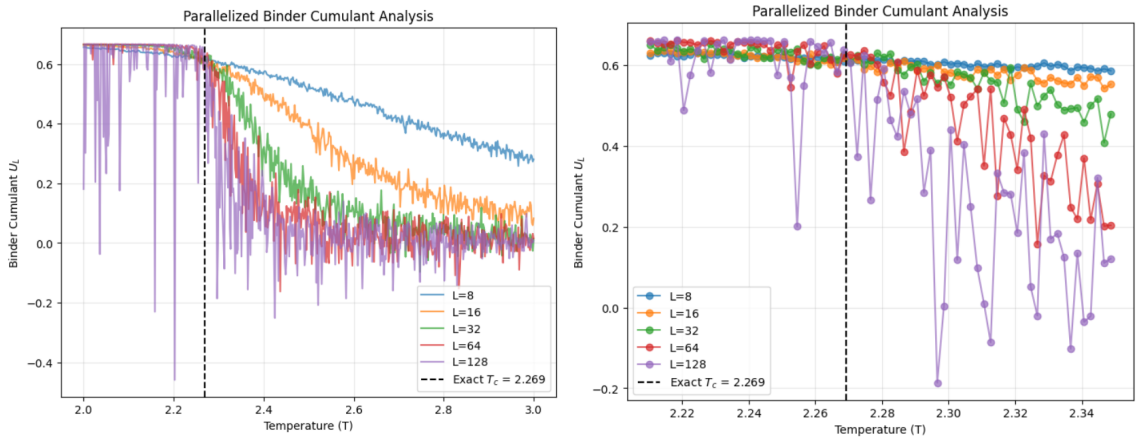
9.1 Algorithm

All simulations and numerical analyses in this section were implemented using the code available at [8]. The function `simulate_mag(L, T)` performs Metropolis sweeps over a square lattice of size $L \times L$ for `mcs_max` Monte Carlo steps. After discarding the first `n_0` steps for thermalization, it computes the Binder cumulant U_L as defined in Equation 10.

9.2 Results and Discussion

We computed the Binder cumulant U_L for various lattice sizes $L = 8, 16, 32, 64, 128$ across a temperature range from $T = 2.0$ to $T = 3.0$. A total of 500 temperature points were sampled for each lattice size. Simulations were run on a high-performance computing (HPC) cluster with parameters `mcs_max` $= 5000$ and `n_0` $= 1000$.

The resulting Binder cumulant curves are shown in Figure 22. The left panel displays the full temperature range, while the right panel provides a close-up near the critical region.



(a) (b)

Figure 22: Binder cumulant as a function of temperature for various lattice sizes (a), and a magnified view near the critical region (b).

As shown in Figure 22, the curves for different lattice sizes initially overlap in the low-temperature regime, where the system is ordered. Near the critical temperature, the curves cross each other at a single point, corresponding to the phase transition. This crossing point provides an estimate of the critical temperature T_c, independent of system size. Above T_c, the cumulant values diverge again, with smaller lattices exhibiting a less pronounced drop in U_L, while larger lattices display a steeper descent.

The right panel highlights these behaviors more clearly. Initially, larger lattices show slightly higher cumulant values than smaller ones. However, as the temperature approaches criticality, all curves intersect, after which the cumulants for larger lattices dip more sharply, resulting in a reversal of their order.

This behavior—curve crossing at criticality and size-dependent divergence away from it—is a hallmark of finite-size scaling and confirms the efficacy of the Binder cumulant in locating the critical point in spin systems.

10 Cluster Size (Metropolis)

10.1 Algorithm

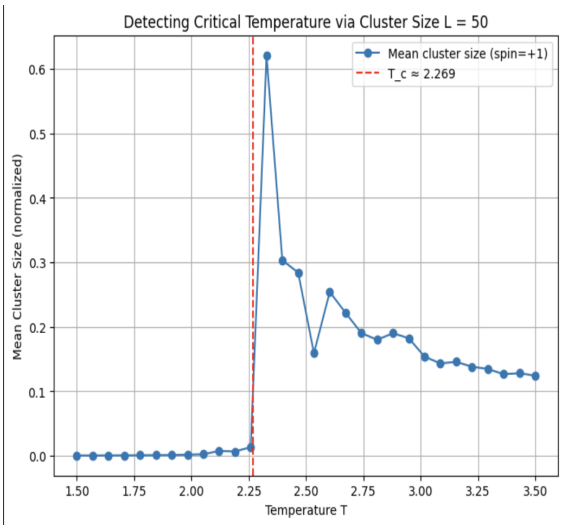
Measuring the average size of the clusters is another useful way to deduce the critical temperature. In our situation, we defined a cluster as any group of one or more spins next to each other. To compute this value we defined *mean cluster size* as

$$S = \frac{\sum_s s^2 n_s}{\sum_s s n_s} \tag{11}$$

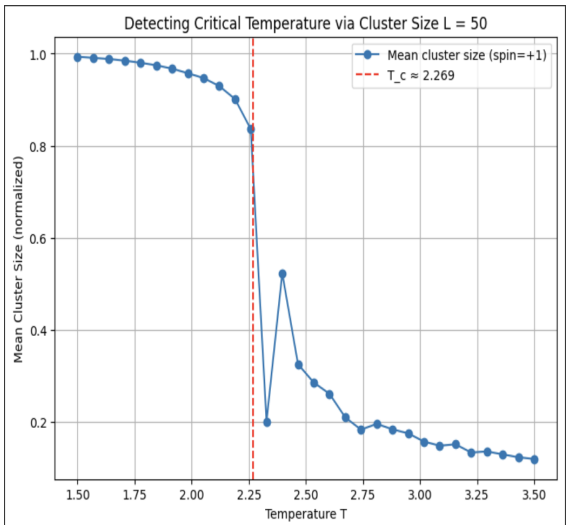
and all computations were done with the code found here [9]. The function `mean_cluster_size` takes an input lattice and uses the function label() from the import `scipy_ndimage` to label and count different clusters based on the array, `structure`, which determines that connectivity is by adjacent neighbors.

10.2 Results

To find the critical temperature we plotted mean cluster size vs temperature as seen in figures 23a and 23b. The only difference between the two depends on the direction of spin in the initialized lattice. Either way the result can still be used to deduce where the critical temperature is. The only difference is that when the initialized spin direction is the same as the type of spin we are measuring such as in figure 23a, the critical temperature appears as a peak. Then when the initialized spin type and the measured type differ, as in figure 23a, the critical temperature occurs as an inflection point. For both of the graphs below at each temperature, we took 1000 thermalization steps then simulated 500 sweeps and took samples every 25 sweeps.



(a) Average Cluster Size vs Temperature for L=50 and initialized lattice of all down spin



(b) Average Cluster Size vs Temperature for L=50 and initialized lattice of all up spin

Figure 23

11 Two-Point Correlation (Metropolis)

11.1 Definition

The *two-point correlation function* provides a measure of how spin orientations at two distinct sites on a lattice are statistically related. The decay of this function with distance reflects how rapidly the influence of local spin configurations diminishes. For spins s_i and s_j at positions i and j, separated by a distance $r = |i - j|$, two-point correlation is defined as:

$$C(r) = \langle s_i \cdot s_j \rangle - \langle s_i \rangle \langle s_j \rangle \tag{12}$$

This expression quantifies the deviation from statistical independence between the two sites. The behavior of $C(r)$ varies significantly with temperature:

- At $r = 0$: $C(0) = \langle s_i^2 \rangle - \langle s_i \rangle^2 = 1 - m^2$
- At large distances r:
  - $T < T_c$: $C(r) \to 1 - m^2 \to 0$, indicating long-range order.
  - $T > T_c$: $C(r) \propto e^{-r/\xi}$, decaying exponentially with correlation length ξ.
  - $T \approx T_c$: $C(r) \propto r^{-1/4}$, showing algebraic decay at the critical point.

11.2 Algorithm

To numerically compute the two-point correlation function, we define the function `two_point_correlation(lattice, r_max)`, which calculates the spin correlation between all spin pairs separated by distances up to `r_max` using equation 12. A more computationally efficient version, `two_point_correlation_fft(lattice, r_max)`, uses the Fast Fourier Transform (FFT) to perform this calculation.

Then `metropolis_swipe(lattice, T)` is run on a randomly generated lattice for `n_0` steps (thermalization period), which allows the lattice to equilibrate. The `two_point_correlation(lattice, r_max)` function is then run over the ensemble of evolved spin configurations.

The code used for this analysis is available at [8] and [9].

11.3 Results and Discussion

The algorithm described in Section 11.2 was applied to a 100×100 lattice after a thermalization period of `n_0` Monte Carlo steps. The two-point correlation function was evaluated for five different temperatures, and the results are presented in Figure 25.

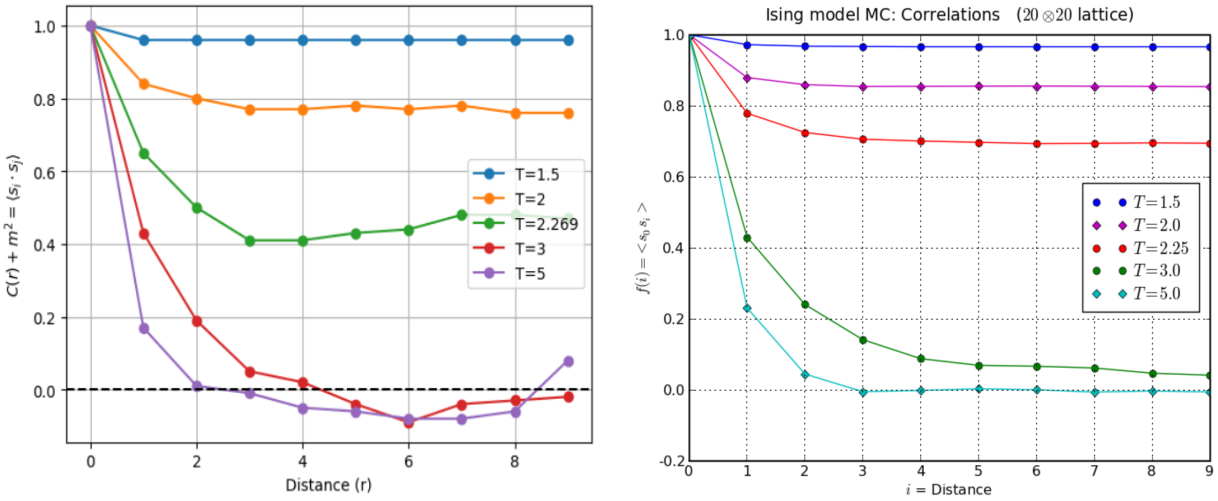
The observed trends in the correlation functions across temperatures are consistent with theoretical predictions:

- **For $T = 1.5$ and $T = 2.0$ ($T < T_C$):** The correlation function $C(r)$ exhibits a rapid decay and stabilizes near zero. This behavior is expected in the ordered phase, where the spontaneous magnetization m is close to unity, and hence $C(r) = 1 - m^2 \approx 0$.
- **For $T = 2.5$ and $T = 3.0$ ($T > T_C$):** The correlation function displays clear exponential decay, consistent with the disordered phase where spin correlations vanish over long distances. The exponential form aligns well with the theoretical prediction $C(r) \propto e^{-r/\xi}$, where ξ is the finite correlation length.
- **For $T = 2.266 \approx T_C$:** The decay of $C(r)$ is noticeably slower compared to the high-temperature cases. This slower, non-exponential decay is characteristic of the critical regime.

To further validate our results, we compared them with reference data from the literature [19]. For this purpose, we considered a modified version of the correlation function by omitting the disconnected term $\langle s_i \rangle \langle s_j \rangle$, and defined:

$$f(r) = \langle s_i \cdot s_j \rangle \tag{13}$$

This function, $f(r)$, was computed for various temperatures and plotted in Figure 24a. The resulting curves exhibit strong qualitative agreement with those reported in Figure 24b, reproduced from [19]. The resemblance confirms the robustness and correctness of our simulation methodology.



(a) (b)

Figure 24: $\langle s_i \cdot s_j \rangle$ vs Distance from (a) our code and (b) literature [19]

11.4 Correlation Length

A key physical parameter that can be extracted from $C(r)$ is the *correlation length*, denoted by ξ. It characterizes the typical distance over which spin configurations are correlated. In statistical mechanics, the correlation length diverges as the system approaches the critical temperature from either side in the thermodynamic limit. For finite systems, this divergence manifests as a pronounced peak.

To estimate ξ, we fit the decay of $C(r)$ to an exponential form:

$$C(r) = A \cdot e^{-r/\xi} \tag{14}$$

By performing this fit at various temperatures, we extract $\xi(T)$.

As illustrated in Figure 26, the correlation length increases sharply as the system approaches T_c, providing another quantitative signal of the critical point.

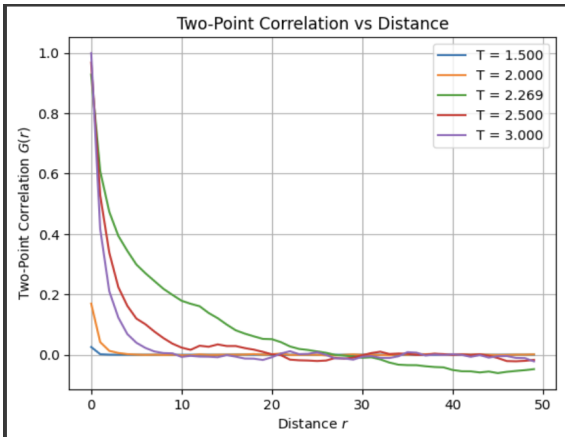


Figure 25: Two-point correlation $C(r)$ as a function of distance r

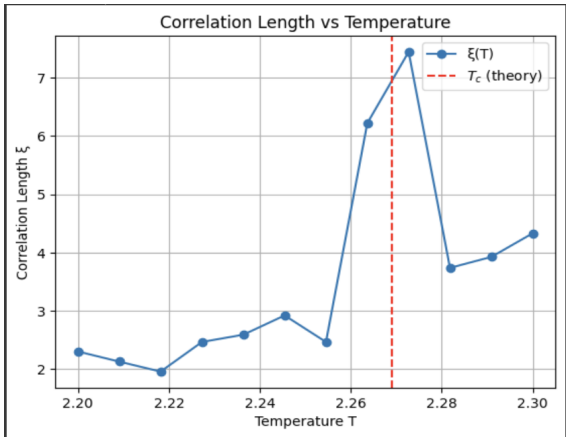


Figure 26: Correlation length ξ as a function of temperature for $L = 65$

To obtain these results, we averaged over 8 independent simulation samples. Each simulation included 1000 thermalization steps to ensure equilibration before data collection. The agreement of these results with theoretical expectations [13] supports the reliability of the implementation.

12 Specific Heat (Metropolis)

Specific heat is a thermodynamic quantity that characterizes how a material responds to changes in energy input. Specifically, it describes the change in temperature resulting from the addition or removal of a unit amount of heat. A material with higher specific heat exhibits smaller temperature changes under thermal perturbations, whereas one with lower specific heat experiences larger fluctuations.

In the context of the Ising model, the specific heat of a lattice can be computed via fluctuations in energy using the following expression:

$$C_V = \frac{1}{k_B T^2} \left[\langle E^2 \rangle - \langle E \rangle^2 \right] \text{[20]} \tag{15}$$

12.1 Algorithm

The function `simulate_cv(L, T)` implements Equation 15 to calculate the specific heat of a square lattice of size $L \times L$ at a given temperature T. The simulation performs `mcs_max` Metropolis sweeps, with the first `n_0` steps discarded for thermalization.

To mitigate correlations between successive measurements, energy values are sampled every `n_s` steps using the function `calculate_energy(lattice)` (discussed previously in Section 3.1). The time series of sampled energies is then used to compute the ensemble averages $\langle E \rangle$ and $\langle E^2 \rangle$, which are substituted into Equation 15 to obtain the specific heat.

12.2 Results and Discussion

Simulations were performed for the following parameters:
Lattice sizes: $L = 10, 20, 50, 100$
Monte Carlo steps: `mcs_max` $= 1000$
Thermalization steps: `n_0` $= 500$
Sampling interval: `n_s` $= 8$
Temperature range: $T \in [1.5, 3.0]$ with step size 0.05, and a denser sampling of 0.01 near the critical temperature ($T_C \approx 2.269$) within the range $T \in [2.2, 2.4]$

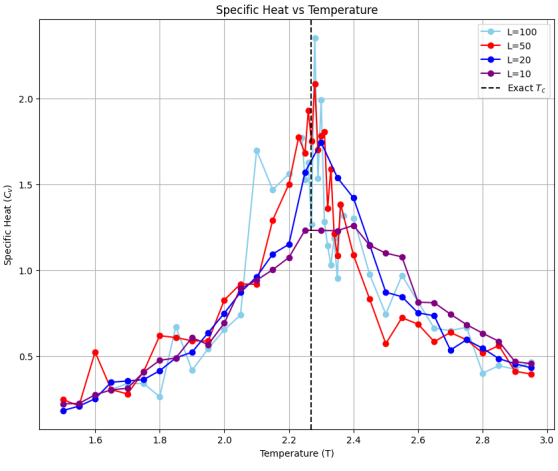


Figure 27: Specific heat vs. temperature for various lattice sizes

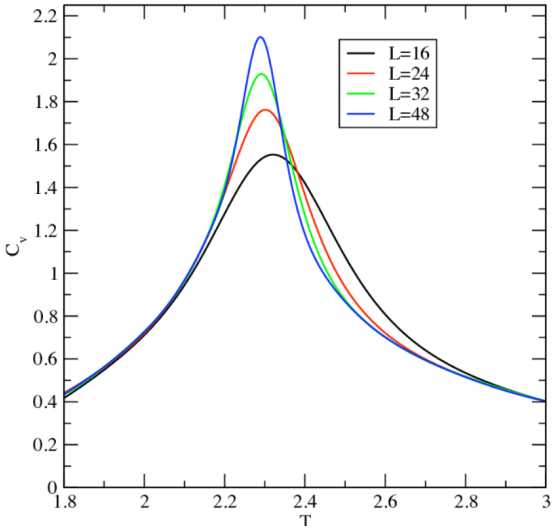


Figure 28: Reference plot from Viot, Pascal (2011)

The computed specific heat values were normalized by multiplying by the number of lattice sites L^2 and are plotted in Figure 27. The following key features are evident:

- The specific heat curves exhibit a prominent peak near the critical temperature.

- The height and sharpness of the peak increase with system size.
- The peak position converges toward $T_C = 2.269$ as L increases.

Figure 28 presents a reference plot adapted from Viot, Pascal (2011) in the lecture notes titled "Numerical Simulation in Statistical Physics". Although the lattice sizes differ slightly, a striking agreement can be observed between the two figures. Notable comparisons include:

- For $L = 20$ (our result) and $L = 24$ (reference), the peak is located at $T = 2.30 \pm 0.05$ with height $C_V = 1.7 \pm 0.1$.
- For $L = 50$ (our result) and $L = 48$ (reference), the peak occurs near $T = 2.28 \pm 0.01$ with height $C_V = 2.1 \pm 0.1$.
- For both small and large lattice sizes, the low- and high-temperature limits of C_V stabilize around 0.5 and 0.4, respectively.

These comparisons confirm that our simulation accurately reproduces the specific heat behavior known from theoretical and numerical studies of the two-dimensional Ising model. The emergence of sharper peaks with increasing lattice size is a hallmark of finite-size scaling near continuous phase transitions.

13 Susceptibility (Metropolis)

Magnetic susceptibility (χ) quantifies the degree to which a system responds to an external magnetic field. In the case of the Ising model, we consider the system in the absence of any external field; nonetheless, χ remains a well-defined and physically meaningful quantity. It captures the fluctuations in the magnetization that arise from thermal effects at equilibrium. These fluctuations become particularly pronounced near the critical temperature and serve as a key indicator of phase transitions, as will be discussed in more detail in Section 14.

Due to the $\mathbb{Z}_2$ symmetry of the Ising model without an external field, the total magnetization M averages to zero in finite systems, especially below the critical temperature where spontaneous symmetry breaking occurs. To accurately characterize the magnetization fluctuations and avoid cancellations due to sign changes, we define susceptibility using the square of the magnetization and the square of the absolute magnetization:

$$\chi = \frac{1}{k_B T} \left(\langle M^2 \rangle - \langle |M| \rangle^2 \right), \tag{16}$$

where M denotes the total magnetization per spin, T is the temperature, and k_B is the Boltzmann constant (set to unity throughout this work) [20].

13.1 Algorithm

Susceptibility is computed using the function `calculate_susceptibility(L, T)`, which implements Eq. 16 by sampling magnetization values from successive Monte Carlo sweeps. The function `run_susceptibility_analysis(L_range, T_range)` extends this computation over a range of lattice sizes and temperatures to analyze critical behavior. The full implementation can be found in the accompanying code repository [8].

13.2 Results & Discussion

The simulations were conducted using `mcs_max = 4000` Monte Carlo steps per site, with an initial thermalization period of `n_0 = 500` steps and a sampling interval of `n_s = 10` to reduce autocorrelation. We computed susceptibility for lattice sizes $L = 8$, 16, 32, 64, and 128 across 20 temperature points in the range $T \in [2.1, 2.7]$.

The resulting susceptibility curves are shown in Figure 29. These curves qualitatively reproduce the reference behavior shown in Figure 49, adapted from Binder [14]. Several characteristic features are evident:

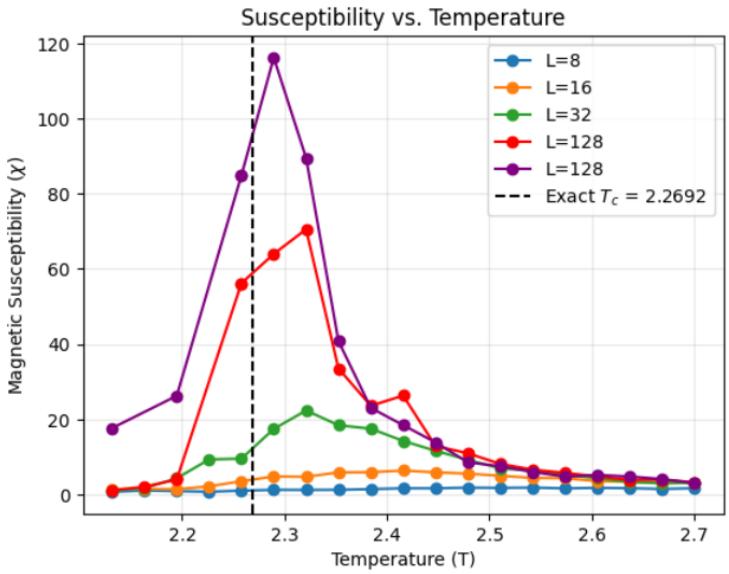


Figure 29: Susceptibility VS Temperature

1. The peaks of the susceptibility curves lie just to the right of the theoretical critical temperature (indicated by a dashed vertical line).
2. As the lattice size increases, the peaks shift leftward toward the critical temperature, consistent with the expected finite-size scaling behavior that suggests divergence of χ in the thermodynamic limit.
3. The susceptibility rises sharply near T_c and decays more gradually on the high-temperature side, producing a skewed bell-shaped curve.

14 Finite Size Scaling (Metropolis)

14.1 Introduction

Finite-size scaling is used to estimate physical quantities in the thermodynamic limit ($L \to \infty$), which better represents many real-world systems. The method involves computing values for increasingly larger lattices and then fitting the data to a scaling relation, allowing for extrapolation to an infinite lattice.

14.2 Magnetization vs Temperature

In Figure 30, this graph represents a finite-size scaling collapse of magnetization (The parameters are L = 10-40, 300 Steps, and 5 Runs). By plotting the rescahled magnetization $\langle |M| \rangle \cdot L^{\frac{\beta}{\nu}}$ against the rescahled tempreature distance $(T - T_c) \cdot L^{(\frac{1}{\nu})}$ (where β is magnetization and ν is the correlation length), which reveals a universal behavior near critical temperature T_c. In finite systems, phase transitions are more rounded and smooth, but through finite-size scaling, based on critical exponents, magnetization data for different lattice sizes become one single curve. In Figure 30, the collapses of curves for lattice sizes ten through forty is consistent with the finite-size scaling in Figure 47.

14.3 Susceptibility vs Temperature

As discussed in section 13, we calculated susceptibility for different lattice sizes for 20 temperatures between 2.1 and 2.8 (shown in figure 31a).
$T_c(L)$ is determined by taking the temperature at which the susceptibility peaks for each lattice. Since the interval of T is $(2.8 - 2.1)/20 = 0.035$, the uncertainty of $T_c(L)$ is 0.035. Then we plotted the $T_c(L)$ against

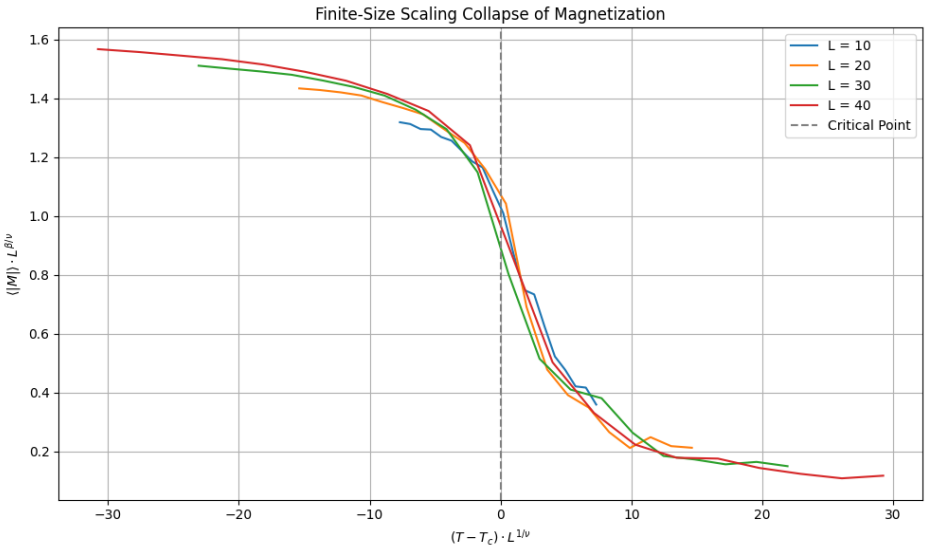
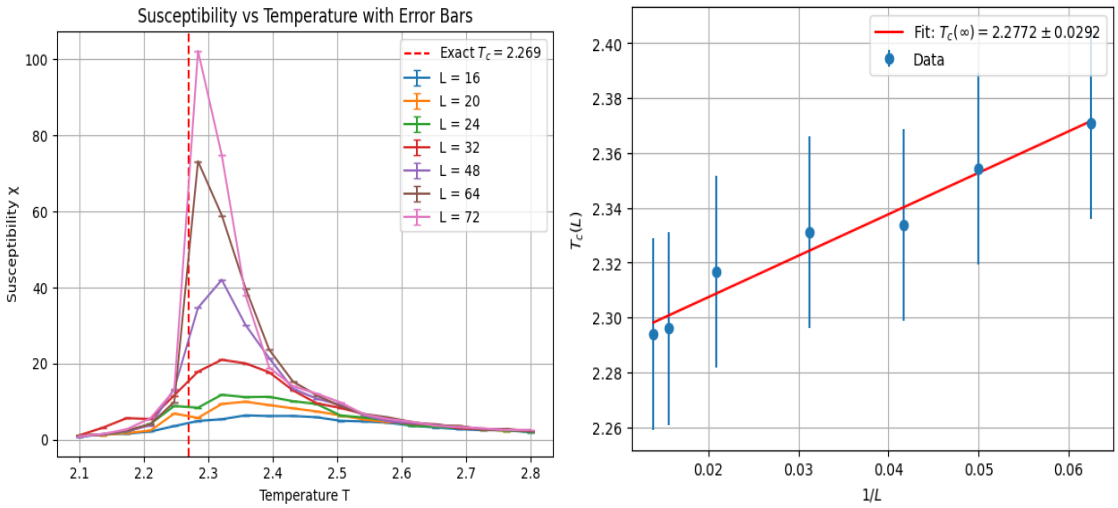


Figure 30: L = 10, 20, 30, 40; 300 Steps, 5 Runs

$1/L$ and fitted a line to it (shown in figure 31b).
The y-intercept represents $T_c(1/L = 0) = T_c(L \to \infty)$. According to our analysis, $T_c(L \to \infty) = 2.28 \pm 0.03$. The theoretical value of $T_c = 2.269$ falls within the uncertainty limit of our measurement.



(a) (b)

Figure 31: (a) Susceptibility vs Temperature (b) Extrapolated Critical Temperature

15 Worm Algorithm Introduction

15.1 What is the Worm Algorithm?

The worm algorithm serves the same general purpose as the metropolis algorithm of 'sweeping' through 2D Ising Model lattices and taking measurements then calculating observables. Many of these observables are the same: 2-pt correlation, susceptibility, energy, and specific heat. Although the Metropolis algorithm works well at low and high temperatures, the autocorrelation time for this algorithm diverges at critical temperature. The Worm algorithm is more efficient because the autocorrelation time for this algorithm stays relatively constant throughout all temperatures.

15.2 Theory Behind Worm Algorithm

A worm has 2 ends i and m, which stand for "Ira" and "Masha". The stationary end, i starts at a random point on the lattice, with initial length 0, while the other end m moves through the lattice by creating or erasing *bonds* between the sites. *Bonds* are formed between 2 neighboring sites when they have aligning spins.

Bonds are represented by n_b which can take 2 values: $n_b = 0$ means no bond is present and $n_b = 1$ means a bond is present. A worm propagates by flipping bonds, which is dictated by the following probability.

$$P = min[1, tanh(\beta)^b] \tag{17}$$

where $\beta = \frac{1}{T}$ and

$$b(m_1 \to m_2) = \begin{cases} +1, & n_b = n_b + 1 \\ -1, & n_b = n_b - 1 \end{cases} \tag{18}$$

The worm continues to increment following equations 17 and 18. Once the m end reaches the i end, the worm closes and a new worm is initiated [21].

15.3 Algorithm

This theory is implemented in the function `move_worm(x, y, bonds, T, L)`, which can be found at [11]. Here a $L \times L \times 4$ 3D array is used to keep track of all the bonds of an $L \times L$ lattice. A direction is randomly picked from where the m end of the worm is, and if flipping that bond is probability-wise preferable, the position of m and the bond from both sites are updated.

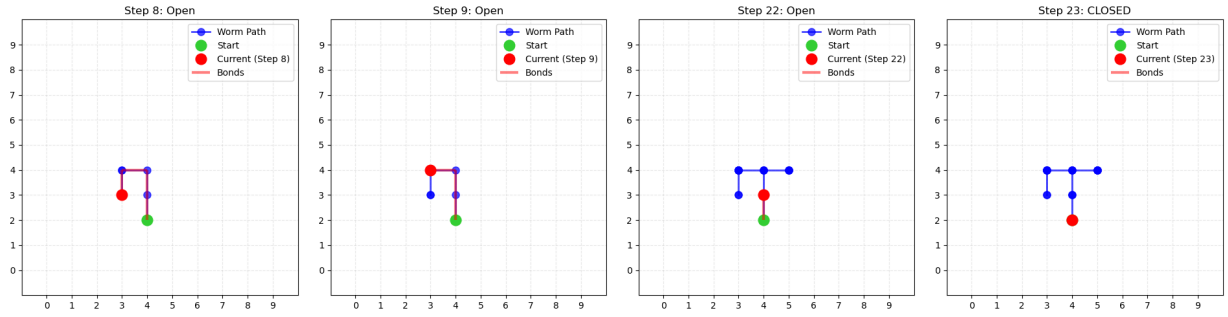
Figure 32 shows visualizations of one worm's motion on a 10×10 lattice at $T = 2.27$. Key observations:

- Worms only move when they can create or remove a bond. In (a), blue lines overlap with red lines, meaning the path only grows when a bond is formed.
- In (b), (c), and (d), the worm retraces steps by removing previously created bonds.
- Bond deletions are always accepted, confirming $P_{\text{flip}} = 1$ when $\Delta n_b = -1$.

We tested bond creation probability by running the algorithm at different temperatures and calculating the empirical acceptance rate:

$$\text{Bond acceptance rate} = \frac{\#\text{Accepted bond insertions}(\texttt{random.random()<P\_accepted})}{\#\text{Attempted insertions}(\Delta n_b = +1)}$$

Figure 33 shows that the empirical acceptance rate matches the theoretical expectation, $P_{\text{accept}} = \tanh(\beta)$.



(a) (b) (c) (d)

Figure 32: Worm Movement on a 10×10 Lattice at $T = 2.27$ (a) Worm progresses only through bond creation, (b) Bond erased on retracing path, (c) Retracing again shows $P_{\text{flip}} = 1$ for deletion, (d) Loop closes when worm head meets tail

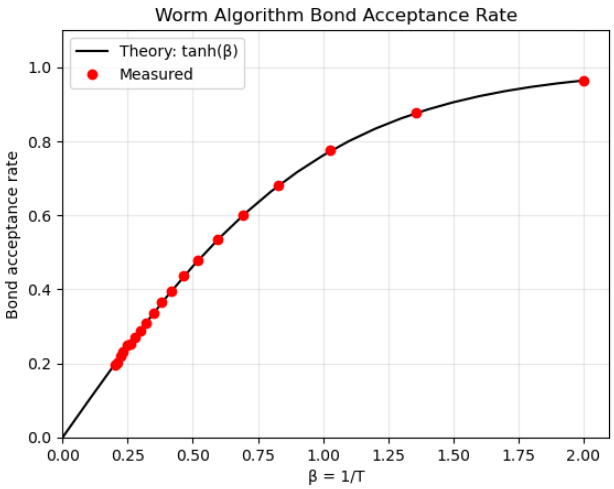


Figure 33: Empirical vs. Theoretical Bond Acceptance Probability

16 Two-Point Correlation (Worm)

The worm algorithm allows us to compute the two-point correlation function:

$$G(r) = \frac{\text{Weight of open loops}}{\text{Weight of closed loops}} = \frac{G_r[r]}{G_0 \times \text{degeneracy}[r]}$$

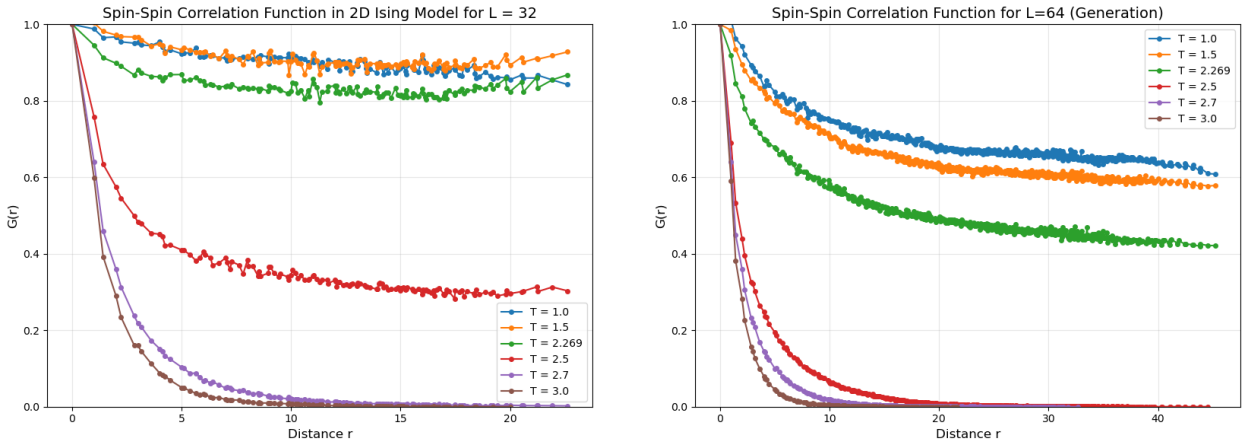
Figure 34 shows results for $L = 32$ and $L = 64$, using $n_{\text{worms}} = 10^4$ and steps per worm $= 10^4$. Key observations:

- For $T > T_c$, the correlation decays exponentially with distance, i.e., $G(r) \to 0$.
- For $T < T_c$, the correlation function plateaus to a non-zero number, indicating long-range order.

These results are consistent with those obtained using the Metropolis algorithm.

Why This Works

Worms propagate through the lattice by creating and erasing bonds. The probability of bond creation is given by $P = \tanh(1/T)$, which increases at low temperatures. Bond erasure is also more frequent at low T, since more bonds are present. As a result, worms can travel longer distances, meaning stronger correlations between distant sites at low temperatures compared to high temperatures.



(a) (b)

Figure 34: Spin-Spin Correlation Function for (a) $L = 32$, (b) $L = 64$

16.1 Correlation Length

To find the correlation length, ξ at any temperature, we can work with the Worm algorithm in the same way as Metropolis. Using equation 14, we fit a power law to our two-point correlation curve and are can extract ξ. An example of this in action is 35 where we worked on a lattice L = 16 at T = 2.3 and ran 10,000 worms.

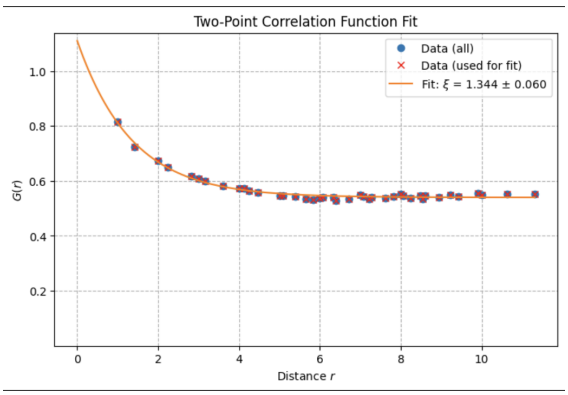


Figure 35: two-point correlation vs r with fit $G(r) = Ae^{\frac{-r}{\xi}}$

Then, if we measure correlation length across multiple temperatures, it acts very similar to how susceptibility will in the following section:

17 Susceptibility (Worm)

To compare a single worm algorithm with the Metropolis algorithm, we first plot the expectation value of the magnetic susceptibility per spin as a function of temperature (Figure 37a). Because we lack direct estimators for most observables in the Metropolis algorithm, *susceptibility* is really the only quantity we can directly compare between the two methods. Also note that our worm algorithm operates within a high temperature expansion of the Ising model–meaning using a reformulated version of the Ising model that is only accurate above the critical temperature where the system is disordered and the expansion converges well–where the average magnetization $\langle M \rangle$ is zero due to system being in the disordered phase. This will allow for a direct estimate of *susceptibility*; however, values computed at below critical temperature fall outside the valid domain of the expansion and therefore are not physically meaningful (According to Cleary

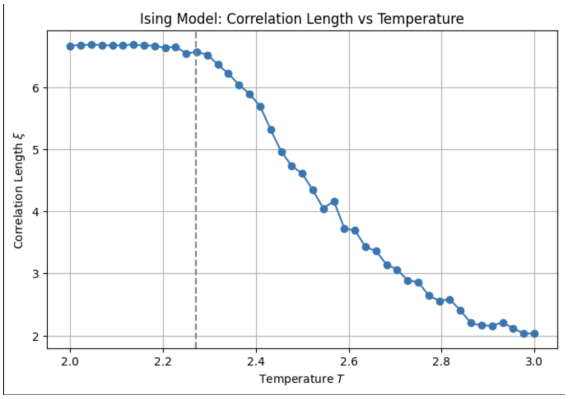


Figure 36: Correlation length vs Temperature for L=32

in his thesis[16]). In addition, we see a prominent peak at the critical temperature (β_c), which sharpens as lattice size increases. This indicates reduced finite-size effects (results become closer to what we'd expect in an infinite system). This essentially confirms the worm algorithm's ability to make physically consistent results. It also allows us to simulate significantly larger lattice sizes ($L = 128$), which would have taken longer with the Metropolis algorithm.

17.1 Algorithm

17.1.1 Susceptibility vs β

- The susceptibility program for Worm Algorithm is similar to the program for Metropolis (Section 13). They both rely on a main, core algorithm and a function that computes susceptibility. The main difference differences between the two is that the Worm Algorithm samples the two-point correlation function and uses "degeneracy" function. In simple terms, shell degeneracy counts how many lattice pairs correspond to each squared distance r^2. In the Worm Algorithm susceptibility program, we multiplied the shell degeneracy by the normalized correlation function to compute magnetic susceptibility χ:

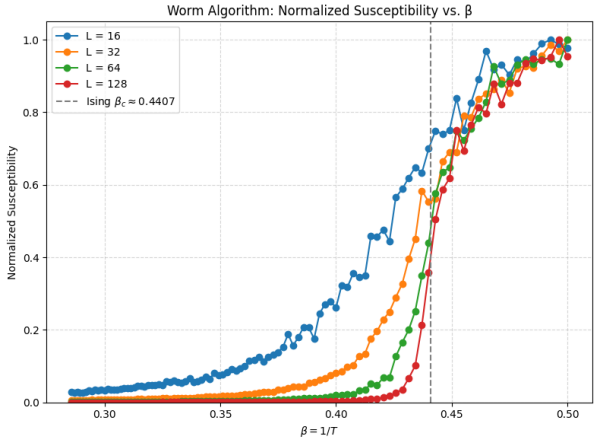
$$\chi = \sum_r G(r) \cdot D(r) \tag{19}$$

Where $G(r)$ is the normalized correlation function and $D(r)$ is the shell degeneracy. We also normalize susceptibility per site: $\chi_{norm} = \frac{\chi}{L^2}$. In Figure 37a, it can be seen that there is an inflection point, instead of a prominent peak (Figure 31a at the critical temperature, which Cleary explains is caused by high temperature expansion. This is consistent with Cleary's results (Figure 50). Our parameters: 3000 equilibration steps (thermalization), lattice sizes 16, 32, 64, 128 (we recommend adding more "in-betweens"), one run (or more), and the worm number depends on lattice size.

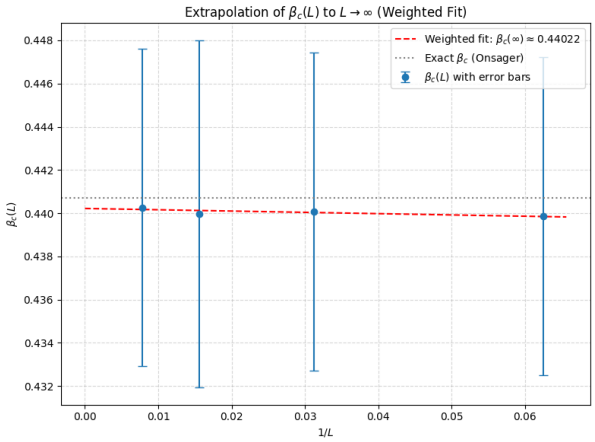
17.1.2 Extrapolate program:

This program estimate the true critical inverse temperature $\beta_c(\infty) = \frac{1}{T_c(\infty)}$ when $L \to \infty$ using measured critical temperatures $\beta_c(L)$ from different finite size lattices. It first loops over each lattice size L and takes the susceptibility χ, then finds the peak of susceptibility and does a local quadratic fit around the peak to get a refined estimate of $\beta_c(L)$. We do this because at finite size L, the susceptibility peaks at a pseudo-critical temperature $\beta_c(L)$, which shifts with L. This collects a list lattice sizes, the fitted β_c values, and their uncertainties. Similar to the Metropolis algorithm, we fit a weighted least squares line to extrapolate to $L \to \infty$ (when $\frac{1}{L} = 0$). Finally, Figure 37b shows the $\beta_c(L)$ plotted for different lattice sizes with a weighted fit, which is used to extrapolate data from susceptibility vs beta. It has large error bars because we only gave it one run, instead of multiple, which reduces accuracy. aa

17.2 Results



(a) Susceptibility vs Temperature for Multiple Lattices



(b) Extrapolated β_c vs $1/L$

Figure 37: Magnetic susceptibility plots using the worm algorithm.

18 Energy (Worm)

18.1 Derivation

The average energy of the system is given by:

$$\langle E \rangle = -J \tanh(\beta) \left[\frac{\langle N_b \rangle}{\sinh^2(\beta)} - dN \right] [15], \tag{20}$$

where d is the number of spatial dimensions (in our case, $d = 2$), and $N = L^2$ is the total number of lattice sites. Consequently, dN represents the total number of bonds, and $\langle N_b \rangle$ is the average number of occupied bonds.

Using the hyperbolic trigonometric identity:

$$\frac{\tanh(\beta)}{\sinh^2(\beta)} = 1 + \tanh(\beta) = \tanh(\beta)(1 + \coth^2(\beta)), \tag{21}$$

we can rewrite equation 20 (setting $J = 1$ and $d = 2$) as:

$$\begin{aligned} \langle E \rangle &= -\tanh(\beta) \left[\langle N_b \rangle \cdot \frac{1 + \tanh(\beta)}{\tanh(\beta)} - 2N \right] \\ &= -\left[\langle N_b \rangle (1 + \tanh(\beta)) - 2N \tanh(\beta) \right] \\ &= -\left[\langle N_b \rangle - (2N - \langle N_b \rangle) \tanh(\beta) \right]. \end{aligned}$$

Recognizing that $\langle N_b \rangle$ is the number of occupied bonds and $2N$ is the total number of bonds, we can express the energy as:

$$\langle E \rangle = -J \left[N_{\text{occupied}} - N_{\text{empty}} \cdot \tanh(\beta) \right], \tag{22}$$

where $N_{\text{empty}} = 2N - N_{\text{occupied}}$.

18.2 Algorithm

Equation 22 is implemented in the function `calculate_energy(bonds, L, T)` (see full code in [11]). The total number of bonds is calculated as $2L^2$, since each site has 4 neighboring bonds but each bond is shared between two sites. The number of occupied bonds is given by the sum of the bond array, i.e., $\sum$ `bonds`. The average energy per site is then obtained by dividing the total energy by L^2.

18.3 Results

The algorithm was executed with the following parameters:

- Number of worms: `n_worm = 5000`
- Maximum number of steps per worm: `steps_per_worm = 5000`
- Temperature range: 50 values of $T \in [0.5, 5.0]$
- Thermalization steps: 1000
- Sampling interval: 10

The results for lattice sizes $L = 32$ and $L = 64$ are shown in Figure 38. A significant transition in energy is observed near the critical temperature, indicating the phase transition.

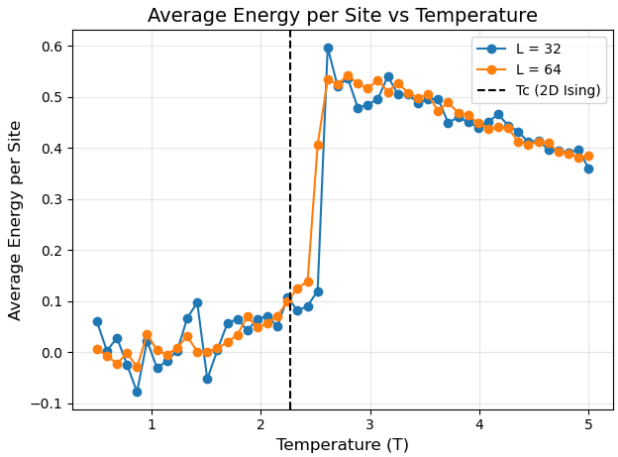


Figure 38: Average Energy per Site vs. Temperature

19 Specific Heat (Worm)

The specific heat quantifies the fluctuation in energy with respect to temperature. It is defined as:

$$C_V = \frac{\langle E^2 \rangle - \langle E \rangle^2}{T^2}. \tag{23}$$

19.1 Algorithm

The function `calculate_energy(bonds, L, T)` records the lattice energy at every 10th step and stores this data in `energy_history`. The specific heat is then computed using this history in the function `calculate_specific_heat(energy_history, L, T)`, based on Equation 23. The complete algorithm is available at [11].

19.2 Results

The simulation was run with the same parameters listed in section 18.3. Results for lattice sizes $L = 16$, 32, and 64 are presented in Figure 39. A clear peak is observed near the critical temperature, consistent with the expected behavior of specific heat regarding phase transition.

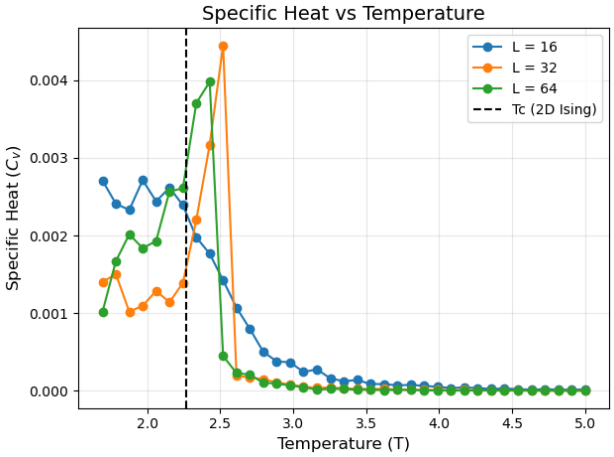


Figure 39: Specific Heat vs. Temperature

20 Autocorrelation (Worm)

The autocorrelation function for the Worm algorithm has the same fundamental form as for the Metropolis algorithm. The function shows the time in Monte Carlo steps t, that it takes for an observable $O(t)$ to become uncorrelated with a later value $t + \triangle t$. The primary difference between the Metropolis and Worm is that for the Worm algorithm a Monte Carlo step occurs after each worm closes, while for Metropolis we denoted a Monte Carlo step to be one "sweep" of the lattice.

20.1 Mathematical Definition

The mathematical definition for this worm iteration is the same as for Metropolis (see equation 5); however, in this instance we use the formula with number of *occupied bonds* as an observable rather than magnetization.

$$C(t) = \frac{\langle O(t') \cdot O(t+t') \rangle - \langle O \rangle^2}{\langle O^2 \rangle - \langle O \rangle^2} \tag{24}$$

20.2 Algorithm

The code for this section can be found here [12] and under the section labeled Autocorrelation.

20.2.1 Generating Bond Time Series

The main function to generate our bond time series is `bond_run`. This function runs a worm, following the rules of the normal worm algorithm and then after a worm finishes it sums the total number of *occupied bonds*. This creates a bond time series that measures bonds through time where time is measured in worms which can then be input into the autocorrelation function.

20.3 Results

Figure blank shows the autocorrelation function for various temperatures. Each line follows an exponential decay of $C(t) = Ae^{\frac{-t}{\tau}}$. Then, we can see that at low temperatures autocorrelation drops to 0 quickly, and then as temperature increases the decay is slower.

20.4 Integrated Autocorrelation time

Integrated autocorrelation for the Worm works the same as with the metropolis and is found directly from autocorrelation function using equation 7.

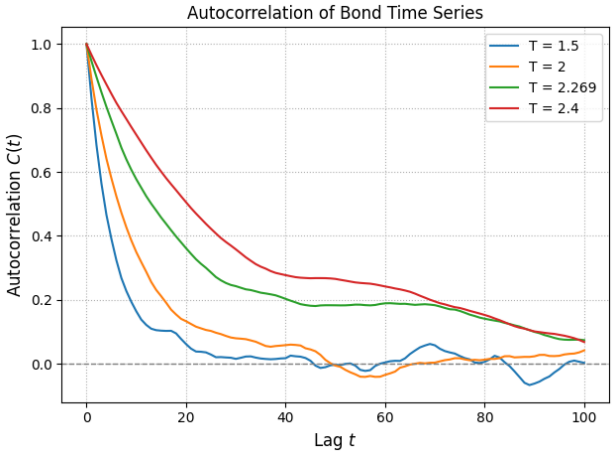


Figure 40: Autocorrelation function vs lag on L=32 for various temperatures

20.4.1 Results

When plotting integrated autocorrelation time vs temperature we see that an inflection point occurs at right around the critical temperature. Also, as lattice sizes increase, so do integrated autocorrelation times, which is expected.

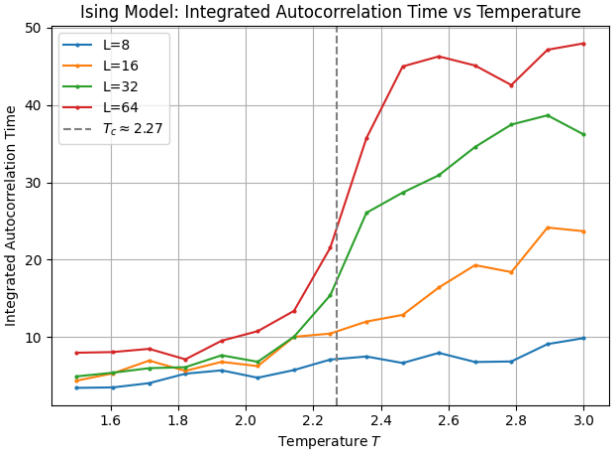


Figure 41: Integrated autocorrelation time vs temperature for various lattice sizes

21 Dynamical Critical Exponent (Worm & Metropolis)

The dynamical critical exponent z, quantifies the *critical slowing-down* of a system around its phase transition. Critical slowing-down describes the phenomenon that occurs when a system approaches its critical temperature, where things such as the *integrated autocorrelation time* τ and *correlation length* ξ diverge. This is because near critical temperature, the system both requires more Monte Carlo steps to get uncorrelated iterations (measured by τ) and the system remains correlated over longer distances (measured by ξ).

21.1 Mathematical Definition

The dynamical critical exponent appears as τ scales with ξ across various temperatures around criticality in the relationship

$$\tau_{int} \sim \xi^z$$

however, a notable quality that occurs at critical temperature is that correlation length can be approximated as lattice size such that (see Sokal's paper [18])

$$\tau_{int} \sim L^z$$

Lastly, to see this effect analytically we plot $\log \tau \sim z \log L$ and extract the critical exponent as the slope of the line.

21.2 Algorithm

The code for this section can be found in two parts, as different versions exist for the Metropolis vs Worm: the worm is here [12] and the metropolis version is here [9], they can both be found under dynamical critical exponent sections. For both versions, we reused the same functions and code used to calculate autocorrelation in its original section. For the metropolis algorithm, we used the functions `mag_timeskip_crit` and `int_auto_corr_length` to generate a time series that measures magnetization and with time measured in sweeps of the lattice. Then for the worm the primary functions are `bond_run` and `integrated_autocorrelation_time` which create a time series that measures number of occupied bonds and time is measured in worms.

21.3 Results

The main importance in our results is the comparison between the dynamical critical exponent measured in the Worm algorithm and the one measured for Metropolis algorithm. From our data we measured the Metropolis to have a $z_{metro} = 1.856 \pm 0.063$ and Worm to have a $z_{worm} = 0.833 \pm 0.030$. It is of note that these variables are also dependent in some part on the observable used to compute them, ie magnetization for Metropolis and occupied bonds for Worm. This also leaves open the possibility for measuring z in different ways, some of which can be found in Sokal's article [18] and also explains why our result may differ from measurements done in other potentially more precise with more computational power backing. On this note the theoretical results are $z = 2.13$ for the Worm and and $z = 0.25$ for the Metropolis.
However, when using our results and comparing the role of magnetization and bonds, they both exist as global variables within each of their systems, and therefore comparing the difference of scale between exponents of each systems could still indicate a difference in critical slowing down effects by each algorithm. Due to the worm's lower dynamical critical exponent we can see that as lattice size increase, while integrate autocorrelation time increases, it increase much slower that for the Metropolis.

22 Conclusion

In this study, we simulated the two-dimensional Ising model using the Metropolis algorithm and observed its behavior across a range of temperatures. As expected, the system exhibited an ordered phase at low temperatures, where spins tended to align, and a disordered phase at high temperatures, where spin alignment was lost. The transition between these two phases, known as the phase transition, occurs near a specific temperature—the critical temperature.

To identify and characterize this transition, we examined several physical quantities. The energy, magnetization, and Binder cumulant showed distinct changes around $T \approx 2.27$, with the energy displaying a sharp increase and the other two quantities showing a sharp decrease. Additionally, cluster size, two-point correlation length, specific heat, and magnetic susceptibility all exhibited pronounced peaks near the same temperature.

For specific heat and susceptibility, we observed that the peak locations shifted closer to the theoretical critical temperature $T_C = 2.27$ as the lattice size increased. A finite-size scaling analysis of the susceptibility

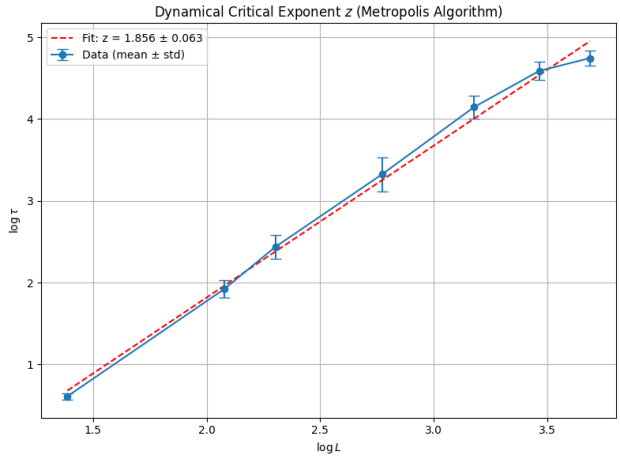


Figure 42: integrated autocorrelation time vs lattice size on a log scale for Metropolis algorithm

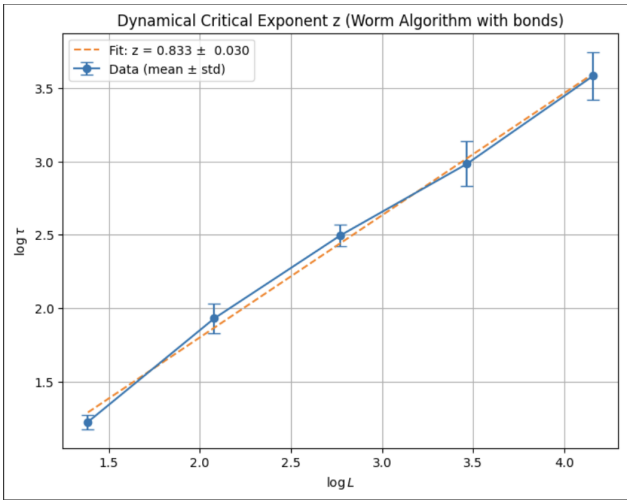


Figure 43: integrated autocorrelation time vs lattice size on a log scale for Worm algorithm

data further confirmed that the peak approaches $T \approx 2.265$ in the thermodynamic limit, providing strong numerical agreement with the known theoretical value ($T_C = 2.269$).

Overall, our results demonstrate the expected thermodynamic behavior of the Ising model and validate the effectiveness of the Metropolis algorithm in capturing critical phenomena.

23 Acknowledgments

We would first like to extend our greatest gratitude to Professor Loinaz for not only giving us this opportunity, but for his continued guidance, support, and patience in this endeavor. We would also like to thank Andrew Leaker (‘26) for all of the meaningful conversations and providing us with knowledge of his expertise. We thank Amherst College for providing us with the resources necessary for our research.

References

[1] Non-reversal and Self Avoiding Random Walk [Jakia].

[2] Random Walk [Jakia].

[3] Random Walk [Carter].

[4] Loops in Random Walk [Jakia].

[5] Basic, non-reversible and self-avoiding Random Walk [Owen].

[6] Lattice: percolation, cluster count and anomalous diffusion [Jakia].

[7] Lattice [Owen].

[8] Lattice: Ising Model [Jakia].

[9] Ising Model Code [Owen].

[10] Ising Model: Magnetization graph code [Ellis].

[11] Worm Algorithm [Jakia].

[12] Worm Algorithm Code [Owen].

[13] John Beggs and Nicholas Timme. Being critical of criticality in the brain. *Frontiers in Physiology*, 3:163, 06 2012.

[14] K. Binder and D.W. Heermann. *Monte Carlo Simulation in Statistical Physics*. Springer, 6th edition.

[15] Massimo Boninsegni. The worm algorithm and path integral monte carlo. https://pitp.phas.ubc.ca/confs/sherbrooke/archives/pitpcifar_sherbrooke2008_worm_boninsegni.pdf, 2008. PITP-CIFAR Sherbrooke Summer School.

[16] Andrew Cleary. Worm algorithms for the ising model. Undergraduate Final Year Project, Trinity College Dublin, 2020. https://www.maths.tcd.ie/~clearya1/ss_files/AndrewCleary_FYP.pdf.

[17] Wikipedia contributors. Square lattice ising model — wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Square_lattice_Ising_model.

[18] Youjin Deng, Timothy M. Garoni, and Alan D. Sokal. Dynamic critical behavior of the worm algorithm for the ising model. *Physical Review Letters*, 99(11):110601, 2007.

[19] Frank Krauss. Numerical methods: Phase transitions — lecture notes. https://www.ippp.dur.ac.uk/~krauss/Lectures/NumericalMethods/PhaseTransitions/Lecture/pt2.html.

[20] Frank Krauss. Numerical methods: Phase transitions — lecture notes. https://www.ippp.dur.ac.uk/~krauss/Lectures/NumericalMethods/PhaseTransitions/Lecture/pt1.html.

[21] Boris Prokof'ev, Nikolai;Svistunov. *A Study of Critical Behavior in the Ising Model Using Monte Carlo Methods*. PhD thesis, University of Massachusetts Amherst, 2001. Available from ScholarWorks@UMass Amherst: https://scholarworks.umass.edu/server/api/core/bitstreams/56c7e25e-a9a1-42de-81a6-4eae28db963d/content.

[22] David Schaich. Lattice simulations of nonperturbative quantum field theories. https://drive.google.com/drive/folders/1cocTP-6REOvygS4V4eHIgCvPncuUCkSX, 2020.

A Appendix

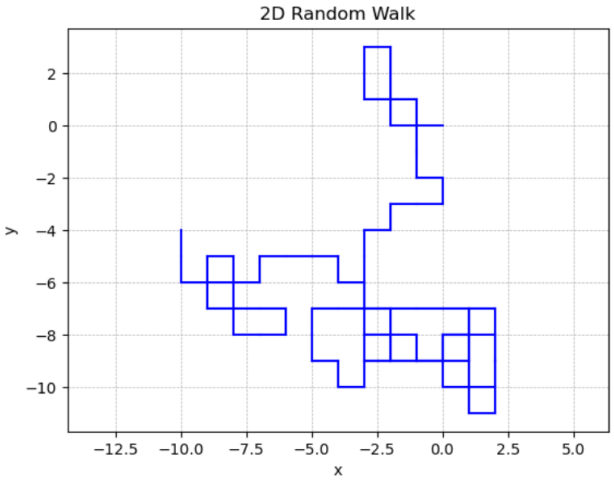


Figure 44: Non-reversible Random Walk

Dimension	Predicted Relationship	Results
2	$\langle x^2 \rangle = 2N$	$\langle x^2 \rangle = 1.9993(10)N$
3	$\langle x^2 \rangle = (3/2)N$	$\langle x^2 \rangle = 1.5018(6)N$
4	$\langle x^2 \rangle = (4/3)N$	$\langle x^2 \rangle = 1.3331(5)N$

Figure 45: Predictions and Results for Non-reversal Random Walks from David Schaich Thesis (Chapter-3)

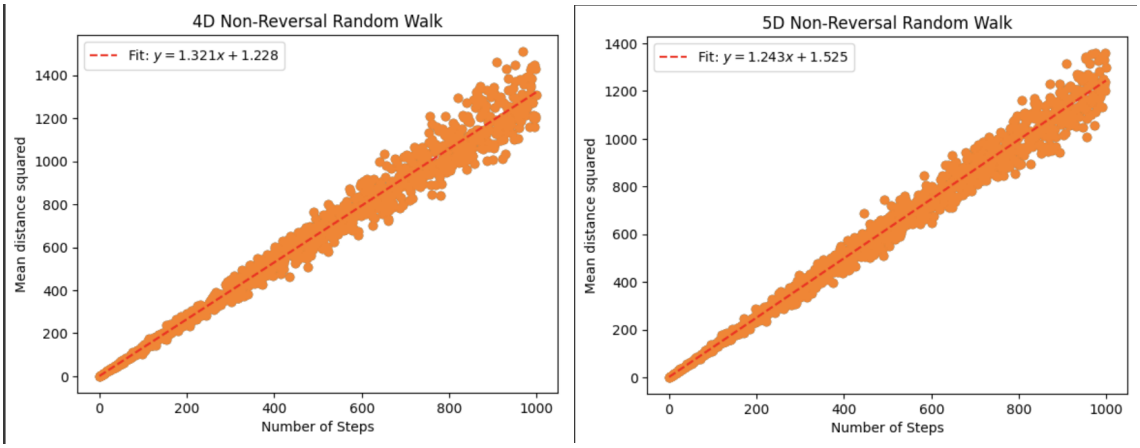


Figure 46: Mean end-to-end Distance Squared for Non-Reversal Walk

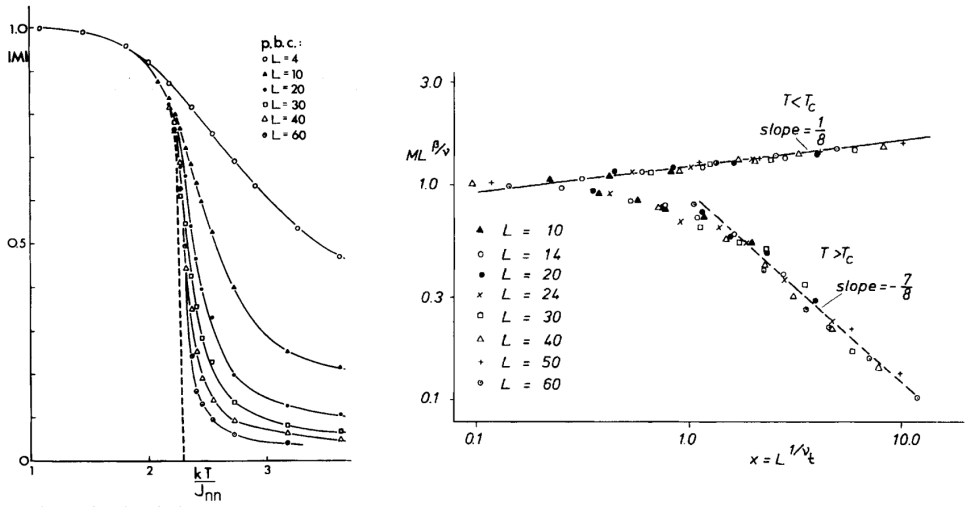


Figure 47: Magnetization of 2D Ising model above Tc and below Tc. (Finite Size Scaling)

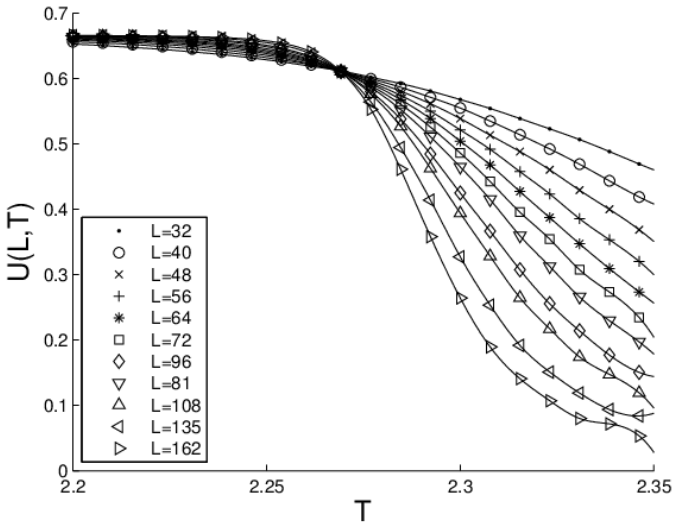


Figure 48: Cumulant vs Temperature from from Palma, Guillermo & Zambrano, Delsito. (2009)

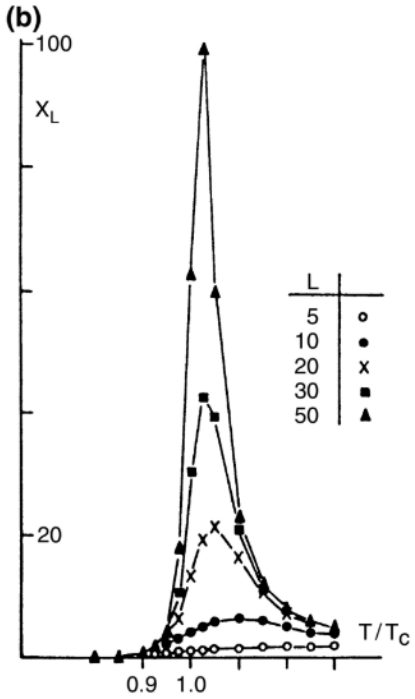


Figure 49: Susceptibility vs Temperature from Binder

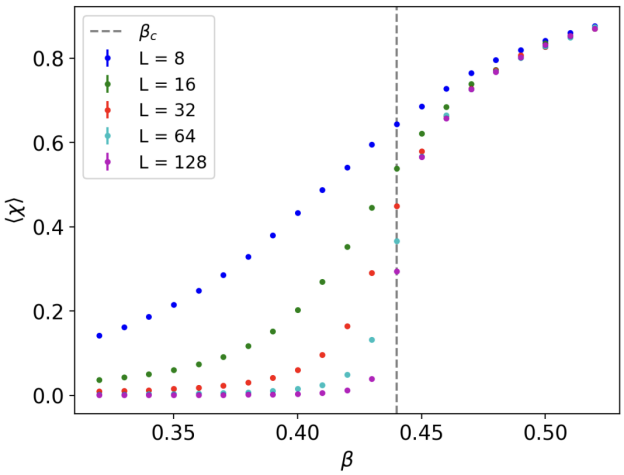


Figure 50: $\langle\chi\rangle$ per spin as a function of β, for $\beta < \beta_c$. The analytic value for c is shown as the dotted line, and the errors were calculated using the jackknife estimator. [16]

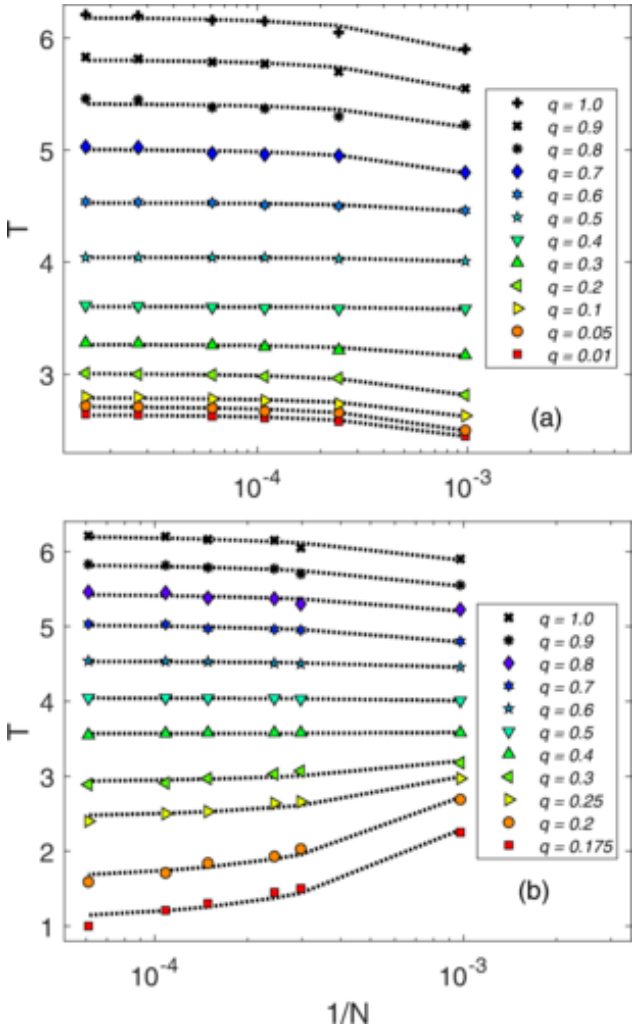


Figure 51: Linear adjustment of the temperature where we have the susceptibility peak, as a function of the inverse of the network size N, to estimate the critical point in the system. In the adjustments in panel (a), we have networks with sizes ranging from $N = (32)^2$ to $N = (256)^2$, and in panel (b), networks with sizes ranging from $N = (32)^2$ to $N = (128)^2$. These were used to find the critical points in Figs. 2(a) and 2(b), respectively. Here the x axis is on a logarithmic scale just to better visualize the points, and the error bars are smaller than the symbol sizes. [14]