Many Body Physics

Kishan D
24B3306

Mentor: Arnav Jain

Abstract

The aim of this project is to get introduced to some basic methods of computation for simulating interacting systems. While it may seem that whatever cannot be done analytically should be possible numerically, it turns out that computational physics is no free lunch. We shall see the issues that arise and how to work around them.

Contents

Introduction

This report presents the computational work carried out as part of the Many Body Physics project. The primary aim of the project was to build practical familiarity with numerical methods used to study interacting many-body systems, and to understand the limitations that arise when working with finite-size simulations and correlated data.

In the first part, we studied site percolation on a two-dimensional square lattice. Clusters were identified using the Hoshen–Kopelman algorithm, and observables such as percolation strength and percolation probability were used to locate the percolation transition. In the second part, we simulated the two-dimensional Ising model using Monte Carlo techniques. We implemented both local (single spin-flip) and cluster updates, and analyzed the behavior of magnetisation, specific heat, and autocorrelation time across a range of temperatures.

All simulations were performed on finite lattices, and the results were interpreted while accounting for statistical fluctuations, finite-size effects, and autocorrelation in Monte Carlo time series.

1 Percolation Theory

1.1 Overview

In the first week of the project, we mainly focused on site percolation. We simulated a two-dimensional percolation system on a finite square lattice with dimensions $L \times L$ where each site was independently occupied with probability p.

The goal of this study was to investigate the percolation phase transition using computational methods. We identified and labeled clusters using the Hoshen–Kopelman algorithm, and computed observables such as the weighted average cluster size, percolation strength, and percolation probability in order to analyze their dependence on the occupation probability p. By studying the behavior of these quantities, we aimed to characterize the percolation transition and estimate the critical probability p_c.

1.2 Site Percolation Model

Consider a two-dimensional finite lattice with dimensions $L \times L$. Each site in the lattice can be in one of two states, occupied or unoccupied, with probabilities p and $1 - p$, respectively, and is occupied independently of all other sites. The parameter p is referred to as the occupation probability or concentration. A group of neighbouring occupied sites forms a cluster.

In this work, only nearest-neighbour connectivity is considered, where neighbours are defined as sites adjacent above, below, to the left, or to the right of a given site. Diagonally adjacent sites are not treated as neighbours. A system is said to be percolating if there exists at least one cluster that spans across opposite sides of the lattice.

1.3 System Initialization

A total of N independent lattice configurations of a two-dimensional square lattice of size $L \times L$ were generated. For each configuration, every lattice site was assigned a random number drawn from a uniform distribution in the interval $[0, 1]$. The lattice was then initialized by comparing these random values with the occupation probability p: sites with values less than or equal to p were marked as occupied, while sites with values greater than p were marked as unoccupied.

Open boundary conditions were used for the lattice, i.e., sites at the edges of the system did not interact with sites beyond the lattice boundaries. Percolation was defined in terms of the existence of a spanning cluster connecting two opposite sides of the lattice.

After labeling all clusters using the Hoshen–Kopelman algorithm, a system was identified as percolating if at least one cluster label appeared on both of the opposite boundaries under consideration. This criterion was checked by examining the cluster labels present on the corresponding boundary sites.

1.4 Cluster Labeling: Hoshen–Kopelman Algorithm

To identify connected clusters of occupied sites on the lattice, we employed the Hoshen–Kopelman (HK) algorithm. This algorithm provides an efficient single-pass method for cluster labeling by combining a raster scan of the lattice with a union–find data structure to handle label equivalences. The lattice is scanned in a row-major order, and cluster labels are assigned based on nearest-neighbour connectivity.

During the raster scan, only the left and upper neighbours of a lattice site are examined, as these are the only sites that have already been visited. If neither neighbour is occupied, a new cluster label is assigned. If exactly one neighbouring site is occupied, the current site inherits the label of that neighbour. If both neighbours are occupied but belong to different clusters, the corresponding cluster labels are merged using a union operation.

Cluster equivalences are maintained using a union–find structure with union by rank and path compression. Union by rank ensures that the shallower tree is attached to the deeper one during a merge operation, while path compression flattens the structure during find operations. These optimizations significantly reduce the computational cost of repeated label lookups.

After the initial raster scan, cluster labels stored on the lattice may still represent intermediate labels due to recorded equivalences. Therefore, a final scan of the lattice is performed to replace each label by its root label obtained from the union–find structure. This ensures that all sites belonging to the same cluster share a unique and consistent cluster label, which is essential for accurately computing cluster-based observables.

The complete procedure used in this work is summarized in the pseudocode below.

```
Raster scan and provisional labeling:
largest_label = 0
labels[x, y] = 0 for all sites
parent[i] = i for i = 0 to L*L
rank[i] = 0   for i = 0 to L*L

for x = 1 to L:
    for y = 1 to L:
        if site (x, y) is occupied:
            left  = labels[x-1, y]
            above = labels[x, y-1]

            if left == 0 and above == 0:
                largest_label += 1
                labels[x, y] = largest_label

            else if left != 0 and above == 0:
                labels[x, y] = find(left)

            else if left == 0 and above != 0:
                labels[x, y] = find(above)

            else:
                union(left, above)
                labels[x, y] = find(left)

Final relabeling scan:
for x = 1 to L:
    for y = 1 to L:
        if labels[x, y] != 0:
            labels[x, y] = find(labels[x, y])
```

```
Union(x, y):
    root_x = find(x)
    root_y = find(y)

    if root_x == root_y:
        return

    if rank[root_x] < rank[root_y]:
        parent[root_x] = root_y
    else if rank[root_x] > rank[root_y]:
        parent[root_y] = root_x
    else:
        parent[root_y] = root_x
        rank[root_x] += 1

Find(x):
    if parent[x] != x:
        parent[x] = find(parent[x])
    return parent[x]
```

Example lattice configuration

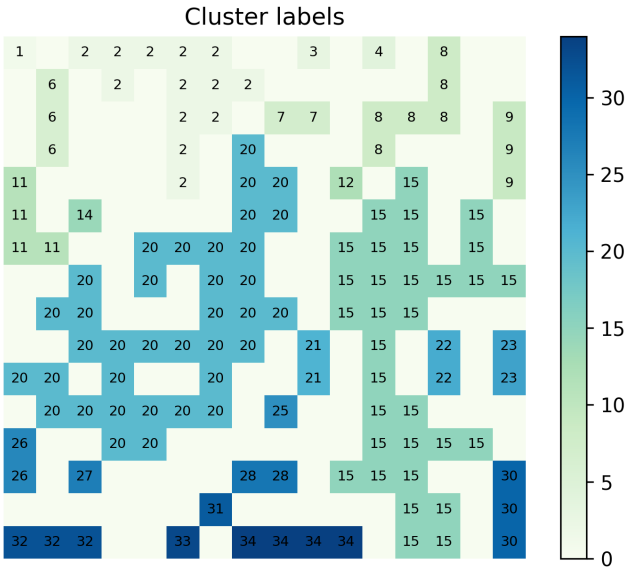


Figure 1: Example lattice configuration (occupied sites shown) used for illustration of the Hoshen–Kopelman labeling. Filled points indicate occupied sites.

1.5 Observables and Definitions

After labeling all clusters in the lattice, we computed a set of observables to characterize the percolation transition as a function of the occupation probability p. Each quantity was

evaluated for every lattice configuration and then averaged over the set of configurations.

Weighted average cluster size

The weighted average cluster size (excluding any percolating cluster) was computed as

$$S_w(p) = \frac{\sum_s s^2 n_s(p)}{\sum_s s\, n_s(p)},$$

where $n_s(p)$ denotes the number of clusters of size s at occupation probability p, and the sums run over all non-percolating clusters.

Percolation strength

The percolation strength measures the fraction of sites contained in the largest cluster:

$$P_\infty(p) = \frac{S_{\max}(p)}{L^2},$$

where $S_{\max}(p)$ is the size of the largest cluster in the lattice (zero if the system does not percolate).

Percolation probability

The percolation probability quantifies the fraction of independent lattice configurations that percolate at a given p. It was computed as

$$\Pi(p) = \frac{N_{perc}(p)}{N},$$

where $N_{perc}(p)$ is the number of configurations that show a spanning cluster and N is the total number of configurations used.

1.6 Results

Simulation parameters (used for the figures)

Parameter	Value
Number of independent configurations	$N = 5000$
Lattice sizes	$L = 8,\ 16,\ 32$
Occupation probability sampling	values used in notebook (full sweep across [0,1])
Boundary conditions	Open (free)
Cluster detection	Hoshen–Kopelman with union-by-rank and path compression
Percolation test	Spanning cluster touching opposite boundaries

Table 1: Key simulation parameters used to produce the plotted results.

Weighted average cluster size

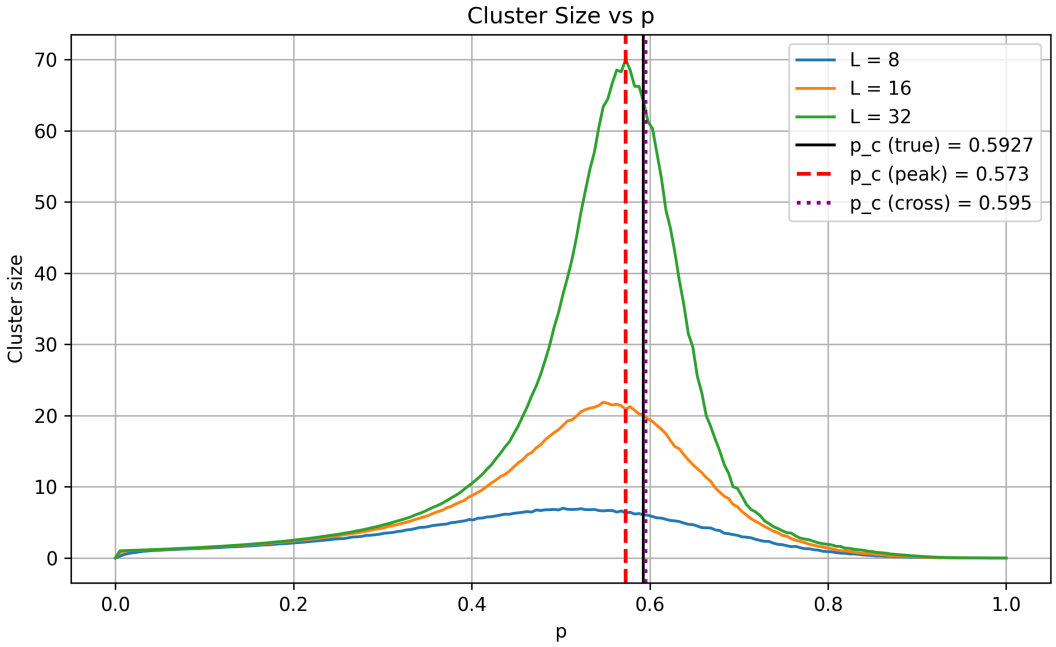


Figure 2: Weighted average cluster size $S_w(p)$ as a function of occupation probability p for lattice sizes $L = 8, 16, 32$. Vertical lines indicate different estimates of the critical probability.

Observations. The weighted average cluster size exhibits a pronounced peak near the percolation transition. The peak height increases with system size while its position shifts due to finite-size effects.

Percolation probability

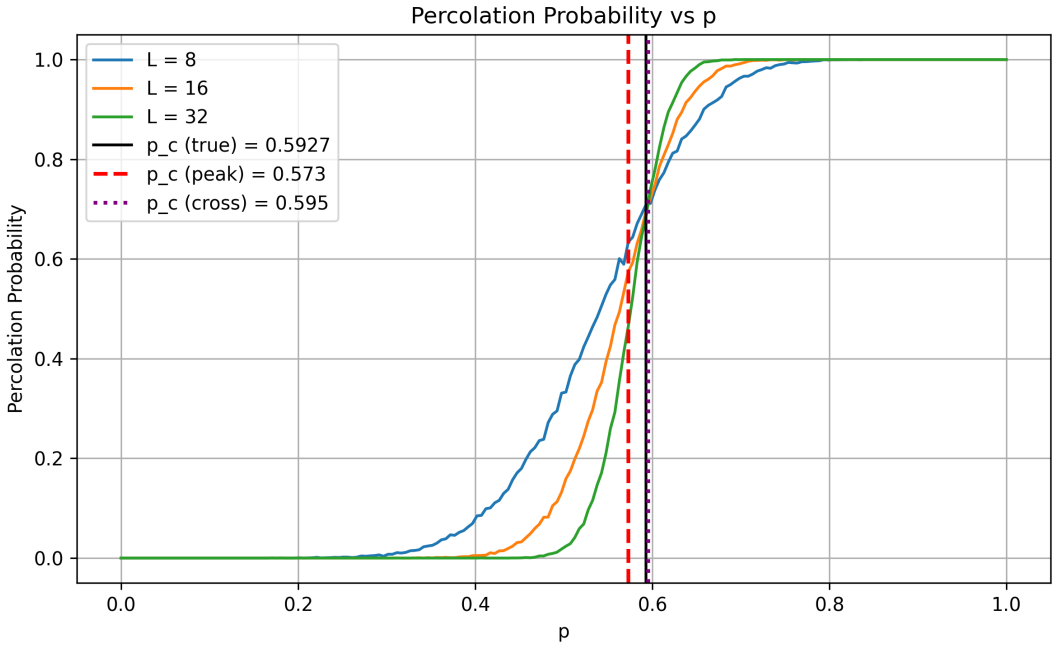


Figure 3: Percolation probability $\Pi(p)$ as a function of occupation probability p for lattice sizes $L = 8, 16, 32$.

Observations. The percolation probability curves show a sigmoidal transition from zero to one. As system size increases, the transition becomes sharper and shifts slightly, reflecting reduced finite-size rounding.

Percolation strength

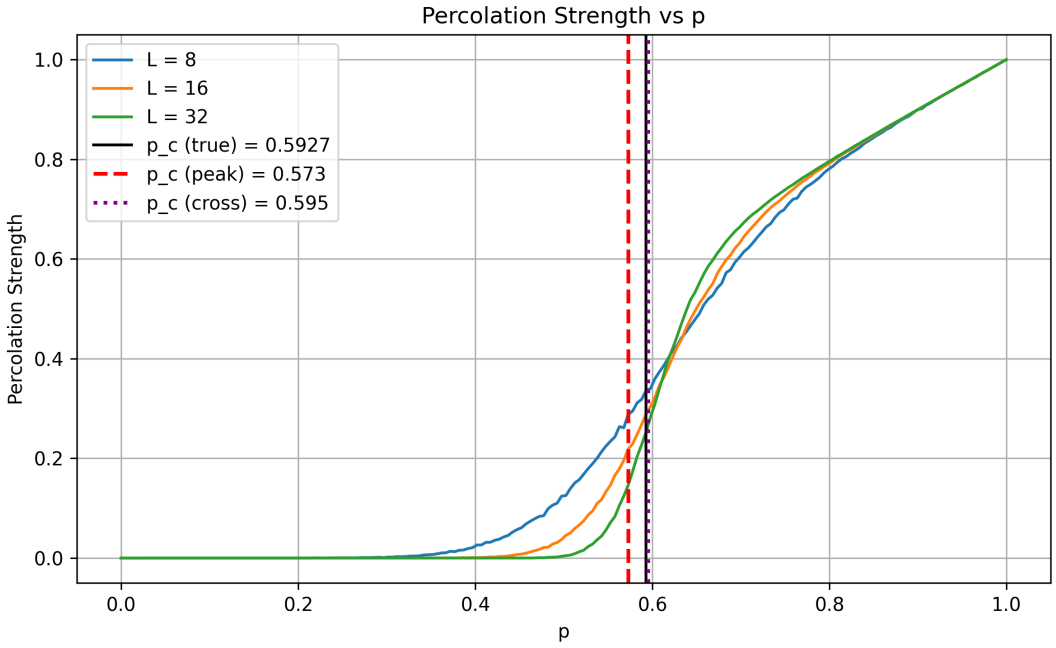


Figure 4: Percolation strength $P_\infty(p)$ as a function of occupation probability p for lattice sizes $L = 8, 16, 32$.

Observations. The percolation strength increases continuously beyond the transition region. For larger lattices the increase is steeper, whereas for smaller lattices the onset is smoothed by finite-size effects.

Estimation of the critical probability

Finite-size estimates of the critical probability were obtained using different criteria. The peak position of the weighted average cluster size yields $p_c(peak) \approx 0.573$, while a crossing-based estimate obtained from the percolation probability and percolation strength curves gives $p_c(cross) \approx 0.595$. For comparison, the known infinite-lattice value for site percolation on the square lattice, $p_c \approx 0.5927$, is also indicated in the figures.

The discrepancies between these estimates are expected given the limited system sizes and sampling. A systematic finite-size scaling analysis or measurements at larger L and denser p-sampling would be required to obtain a more accurate extrapolation of p_c.

1.7 Conclusion

In this phase of the project, we studied site percolation on a finite two-dimensional square lattice using computational methods. A percolation system was initialized for different

occupation probabilities, clusters were identified using the Hoshen–Kopelman algorithm, and several observables were computed to characterize the percolation transition.

The weighted average cluster size, percolation probability, and percolation strength all exhibited clear signatures of a phase transition as the occupation probability was varied. Finite-size effects were evident in all observables, with the transition becoming sharper and shifting toward higher values of p as the lattice size increased. Estimates of the critical probability obtained from different criteria showed reasonable agreement with the known infinite-lattice value, with deviations attributable to finite lattice sizes and limited sampling.

These results provide a clear numerical demonstration of a phase transition in a simple lattice system. As the occupation probability increases, the system undergoes a transition from a non-percolating phase, where clusters remain finite, to a percolating phase characterized by the emergence of a system-spanning cluster. The consistent behavior observed across different observables confirms the critical nature of this transition and highlights how collective, large-scale connectivity arises from local site occupation rules.

Code availability: The source code used to generate the results and figures in this report is available at the project GitHub repository.

References

- Hoshen–Kopelman algorithm (Wikipedia).
- K. Christensen, *Percolation Theory* (lecture notes).
- Bethe lattice (Wikipedia).

2 Ising Model

2.1 Overview

In the second week of the project, we studied the two-dimensional Ising model as a simple lattice model of an interacting spin system. The system was simulated on a finite square lattice of size $L \times L$, where each site carries a spin variable $s_i \in \{-1, +1\}$ and interactions are restricted to nearest neighbours.

The objective of this study was to investigate the thermal phase transition between an ordered (ferromagnetic) low-temperature phase and a disordered high-temperature phase by performing Monte Carlo simulations over a range of temperatures. Two different update schemes were implemented and compared: the single spin-flip Metropolis algorithm and the Swendsen–Wang cluster algorithm.

To characterize the system, we measured observables such as the magnetisation, energy, specific heat, and the integrated autocorrelation time. In particular, we used binning analysis to account for correlated Monte Carlo samples and to estimate statistical uncertainties in magnetisation measurements. By comparing magnetisation and autocorrelation time as functions of temperature for multiple lattice sizes, we aimed to locate the critical temperature and identify clear signatures of the phase transition.

2.2 Two-Dimensional Ising Model

The two-dimensional Ising model is a simple lattice model of interacting spins. Each lattice site i carries a discrete spin variable $s_i \in \{-1, +1\}$. We consider a square lattice of size $L \times L$ and restrict interactions to nearest neighbours only.

The energy of a spin configuration is given by the Ising Hamiltonian

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

where J is the coupling strength and the sum runs over all nearest-neighbour pairs $\langle i, j \rangle$. For $J > 0$, the interaction is ferromagnetic and the energetically preferred state is one in which neighbouring spins align. Thermal fluctuations are introduced through the temperature T, or equivalently $\beta = 1/T$ (setting $k_B = 1$).

A key feature of the two-dimensional Ising model on the square lattice is the existence of a continuous phase transition at a finite critical temperature. In the thermodynamic limit, the exact critical temperature (for $J = 1$) is given by

$$T_c = \frac{2}{\ln(1 + \sqrt{2})} \approx 2.269.$$

Below T_c the system is magnetically ordered and develops a non-zero spontaneous magnetisation, while above T_c the system is disordered and the magnetisation vanishes in the long-time average.

2.3 System Initialization and Boundary Conditions

For each simulation run, the Ising spin lattice was initialized as a random configuration of spins on a finite square lattice of size $L \times L$, where each site was assigned a value

$s_i \in \{-1, +1\}$ with equal probability. This choice provides an unbiased starting point and does not favour either of the magnetised phases.

Periodic boundary conditions were imposed in both spatial directions. In this setting, spins located at the edges of the lattice interact with spins at the opposite boundary, effectively wrapping the lattice into a torus. This reduces finite-size edge effects and provides a better approximation of an infinite system compared to open boundaries.

For a given temperature T (or inverse temperature $\beta = 1/T$, with $k_B = 1$), the system was evolved using Monte Carlo updates. Since the initial configuration is generally far from equilibrium, an initial number of Monte Carlo iterations were discarded in order to allow the system to reach a steady state. After this thermalization period, measurements of observables such as magnetisation and energy were collected over a subsequent measurement window and averaged to obtain equilibrium estimates.

2.4 Algorithms Implemented

To simulate the Ising model at a given temperature, Monte Carlo updates were used to generate spin configurations distributed according to the Boltzmann weight. In this work, we implemented two different update schemes and compared their performance: a local single spin-flip Metropolis update and a non-local cluster update using the Swendsen–Wang algorithm.

2.4.1 Single spin-flip Metropolis update

The Metropolis algorithm updates the lattice locally by selecting spins at random and attempting to flip them. The proposed flip is accepted depending on the energy difference ΔE between the trial and current configurations. If the energy decreases, the move is always accepted. Otherwise, it is accepted with probability $\exp(-\beta \Delta E)$. This ensures that the update satisfies detailed balance and converges to the correct equilibrium distribution.

For a site (i, j) on a square lattice with periodic boundary conditions, the energy difference for flipping a spin is

$$\Delta E = 2J s_{ij} \left(s_{i+1,j} + s_{i-1,j} + s_{i,j+1} + s_{i,j-1} \right).$$

The algorithm below summarizes one Monte Carlo sweep, defined as L^2 attempted spin flips.

```
Metropolis single spin-flip sweep:
Input: spins s[L,L], inverse temperature beta, coupling J

for attempt = 1 to L*L:
    choose random site (i, j)
    compute DeltaE = 2*J*s[i,j]*(s[i+1,j] + s[i-1,j] + s[i,j+1] + s[i,j-1])

    if DeltaE <= 0:
        s[i,j] <- -s[i,j]
    else:
        draw r uniformly from [0,1]
        if r < exp(-beta*DeltaE):
```

```
            s[i,j] <- -s[i,j]

return updated spins
```

2.4.2 Cluster update: Swendsen–Wang algorithm

While the Metropolis algorithm updates spins locally, cluster algorithms attempt to reduce slow relaxation by flipping groups of correlated spins together. The Swendsen–Wang (SW) update works by introducing probabilistic bonds between neighbouring spins that are aligned. Connected components of these bonds form clusters, and each cluster is flipped independently with probability $1/2$.

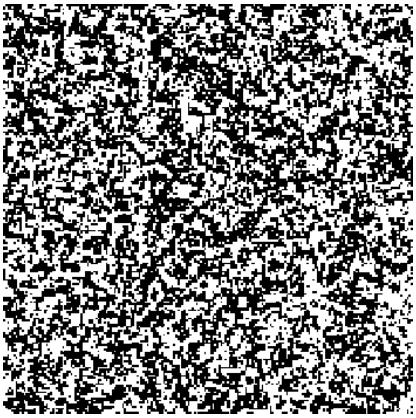
For the ferromagnetic Ising model, a bond between two equal neighbouring spins is activated with probability

$$p = 1 - \exp(-2\beta J),$$

and no bond is placed if the spins are opposite.

Clusters were identified efficiently using a union–find data structure with path compression and union by rank, similar to the approach used in the Hoshen–Kopelman algorithm in the percolation section.

```
Swendsen--Wang cluster update:
Input: spins s[L,L], inverse temperature beta, coupling J
p = 1 - exp(-2*beta*J)

Initialize union-find structure over all lattice sites

# Step 1: activate bonds
for each site (i, j):
    if s[i,j] == s[i+1,j]:
        with probability p: union((i,j), (i+1,j))
    if s[i,j] == s[i,j+1]:
        with probability p: union((i,j), (i,j+1))

# Step 2: flip clusters
for each cluster root:
    choose flip = +1 or -1 with probability 1/2

for each site (i, j):
    root = find((i,j))
    if flip[root] == -1:
        s[i,j] <- -s[i,j]

return updated spins
```



(a) Random initial configuration ($t = 0$).

(b) After equilibration at low T, showing ferromagnetic domains.

Figure 5: Representative Ising spin configurations on a $L \times L$ lattice.

Performance optimisation using Cython

Both the Metropolis and Swendsen–Wang updates involve repeated sweeps over the lattice and therefore require a large number of elementary operations. In pure Python, these updates become slow due to the overhead of nested loops and repeated indexing. To improve performance, the core update steps were also implemented using `Cython`, where static typing and compiled code allow the tight loops to run close to C speed.

In particular, typed memory views were used for fast access to NumPy arrays, and runtime checks such as bounds checking and negative index wraparound were disabled where appropriate. The Cython implementations preserve the same Monte Carlo logic as the Python versions, but run significantly faster, making it feasible to generate long time series required for binning analysis, autocorrelation time estimation, and temperature sweeps.

2.5 Autocorrelation error and binning analysis

A Monte Carlo simulation generates a sequence of configurations $\{C_1, C_2, \ldots, C_N\}$ using a Markov chain. Since each new configuration is constructed from the previous one, successive measurements of an observable are generally not statistically independent. As a consequence, the time series $A_1, A_2, \ldots, A_N$ (for example the magnetisation) is correlated, and the true statistical error on the mean is larger than what would be obtained by assuming independent samples.

For an observable A, the Monte Carlo estimator of the mean is

$$\bar{A} = \frac{1}{N} \sum_{t=1}^{N} A_t.$$

To quantify correlations, we introduce the autocovariance function

$$C(\Delta) = \langle A_t A_{t+\Delta} \rangle - \langle A \rangle^2,$$

and the normalized autocorrelation function

$$\rho(\Delta) = \frac{C(\Delta)}{C(0)} = \frac{\langle A_t A_{t+\Delta} \rangle - \langle A \rangle^2}{\langle A^2 \rangle - \langle A \rangle^2}.$$

The integrated autocorrelation time is defined as

$$\tau_{\text{int}} = \frac{1}{2} + \sum_{\Delta=1}^{\infty} \rho(\Delta),$$

and it measures how many Monte Carlo iterations are required before the system produces an effectively independent sample. In terms of τ_{int}, the standard error of the mean is enhanced compared to the uncorrelated case, and can be written as

$$\sigma_{\bar{A}} \approx \sqrt{\frac{2\tau_{\text{int}}}{N}}\,\sigma_A,$$

where $\sigma_A^2 = \langle A^2 \rangle - \langle A \rangle^2$. This motivates the need for an autocorrelation-aware error analysis.

In practice, the autocorrelation function becomes noisy at large time lags, making direct summation unreliable. Instead, we estimate statistical errors using binning (blocking) analysis. The time series is repeatedly coarse-grained by averaging neighbouring pairs, producing binned series of size 2^l at binning level l. The standard error of the mean is computed at each level and plotted as a function of l. As the bin size increases, correlations are reduced and the estimated error increases until it saturates. The plateau value corresponds to a reliable error estimate, while very large binning levels become unstable due to the small number of remaining bins.

This procedure was applied to the magnetisation time series for both the single spin-flip Metropolis algorithm and the Swendsen–Wang cluster algorithm.

Binning analysis results

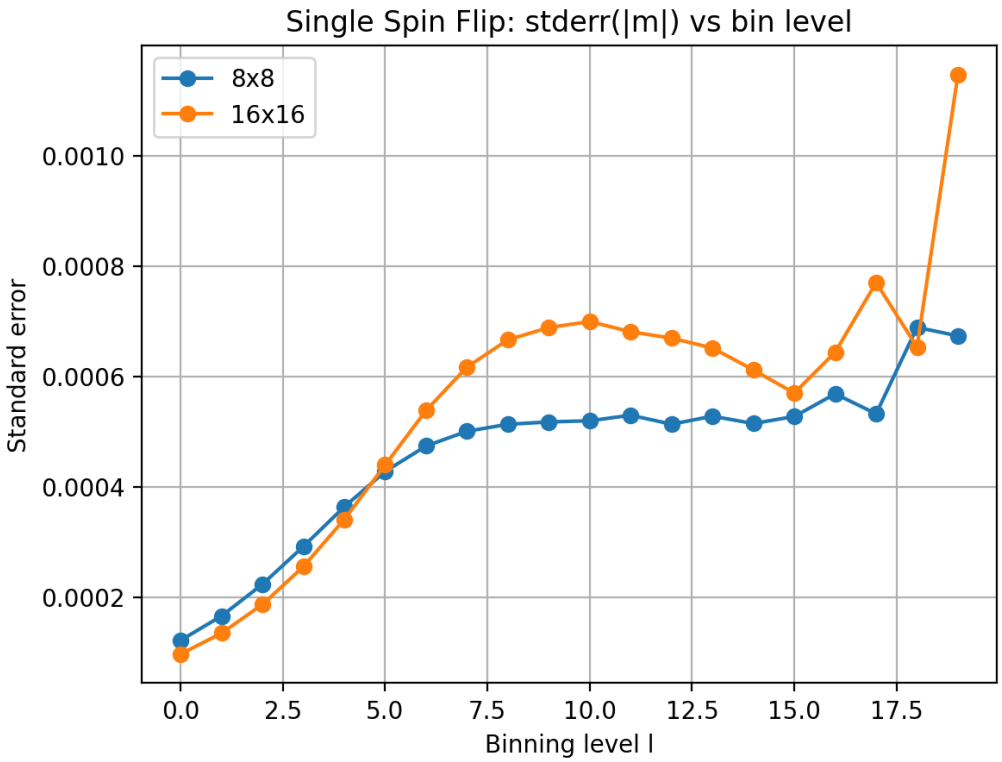


Figure 6: Binning analysis for the single spin-flip Metropolis update. The estimated standard error of the mean absolute magnetisation increases with bin size and eventually saturates, indicating the presence of strong autocorrelations.

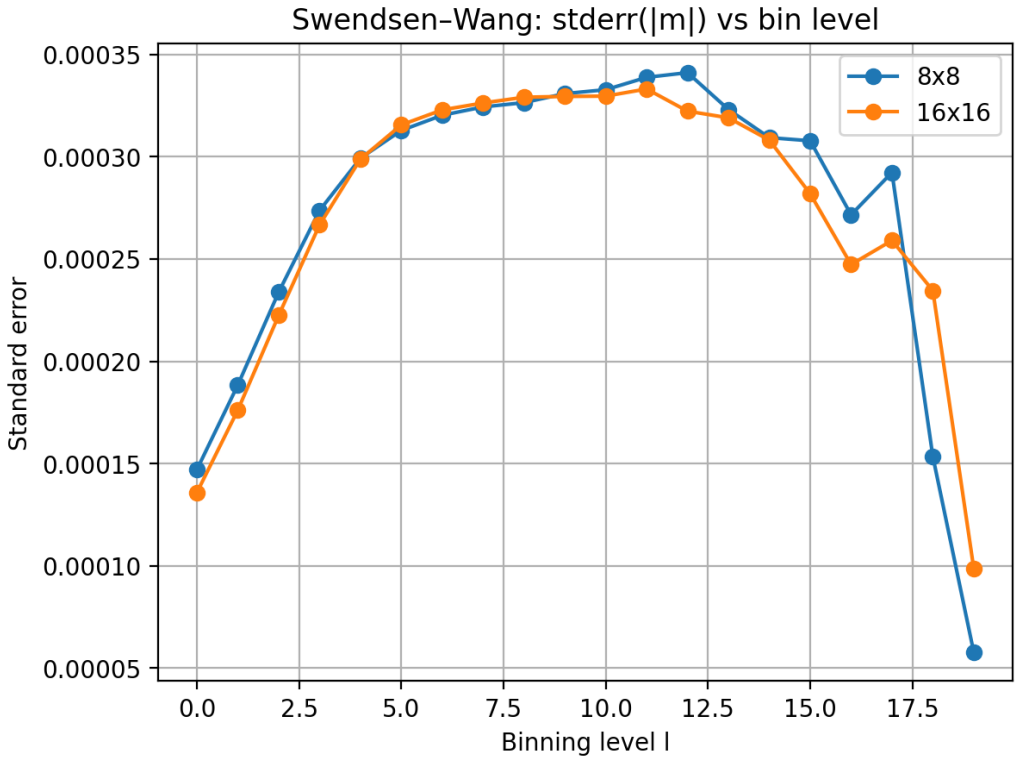


Figure 7: Binning analysis for the Swendsen–Wang cluster update. Saturation is reached at much smaller binning levels compared to Metropolis, showing that cluster updates significantly reduce autocorrelation effects.

Observations. For the single spin-flip Metropolis algorithm, the estimated error grows steadily with the binning level and saturates only at relatively large bin sizes, which indicates long autocorrelation times. The effect becomes stronger for the larger lattice size 16×16, where the saturation plateau is reached later. For the Swendsen–Wang cluster algorithm, saturation occurs much earlier and the plateau is reached at smaller binning levels, implying that consecutive samples decorrelate more rapidly.

At very large binning levels, the error estimate may decrease or fluctuate strongly. This behaviour is expected, since excessive binning leaves only a small number of bins in the coarse-grained data set. The variance estimate becomes statistically unreliable in this regime, and the apparent error can behave erratically.

2.6 Comparison of convergence at low temperature

To compare the efficiency of the two update schemes, we studied the evolution of the magnetisation as a function of Monte Carlo iterations at a low temperature ($\beta = 1.0$, $J = 1$). Starting from a random initial configuration, the system was evolved using either the single spin-flip Metropolis update or the Swendsen–Wang cluster update, and the absolute magnetisation per spin $\langle |m(t)| \rangle$ was recorded as a function of iteration number.

The absolute value of the magnetisation was used, since in the absence of an external field the Ising Hamiltonian is symmetric under a global spin flip. As a result, the magnetisation can change sign during the simulation even in the ordered phase, which can obscure the equilibration behaviour when plotting $m(t)$ directly.

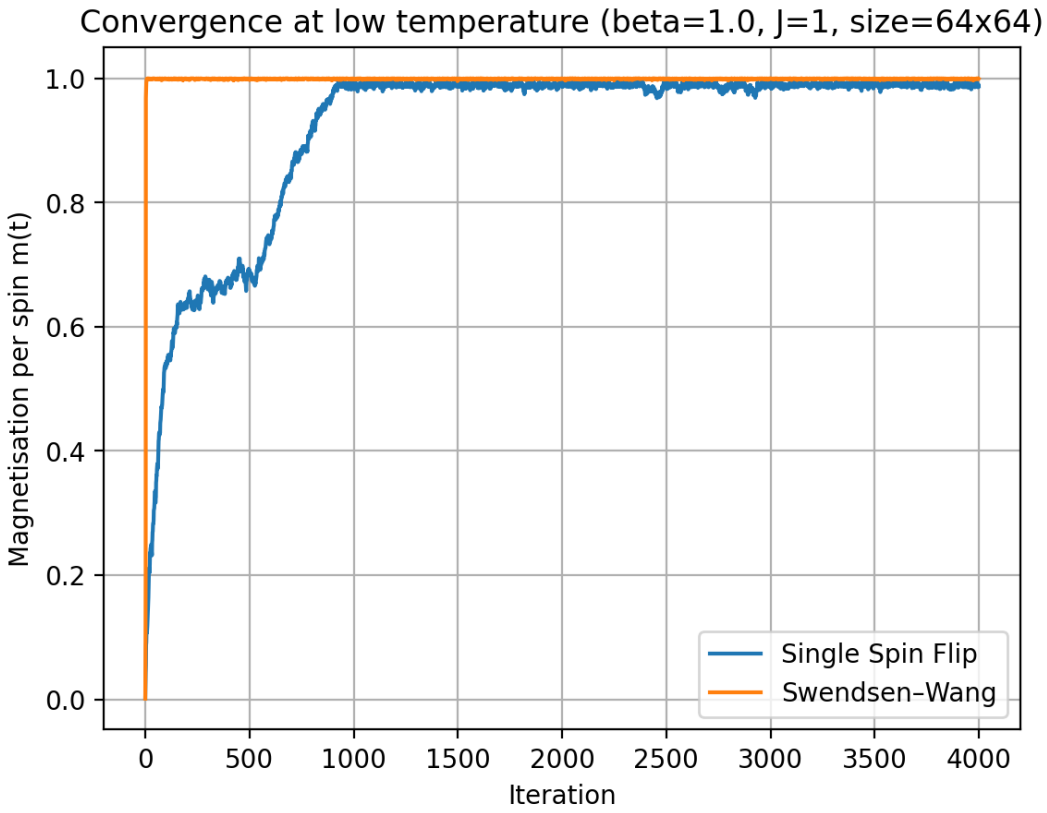


Figure 8: Convergence of the absolute magnetisation per spin $|m(t)|$ as a function of Monte Carlo iteration at low temperature ($\beta = 1.0$). The Swendsen–Wang update reaches an ordered state significantly faster than the single spin-flip Metropolis update.

Observations. The Swendsen–Wang algorithm reaches $|m| \approx 1$ within only a few iterations, indicating that it rapidly produces large ordered domains in the low-temperature phase. In contrast, the single spin-flip Metropolis update converges much more slowly and requires a significantly larger number of iterations to reach the same level of ordering. This behaviour is expected, since cluster algorithms perform non-local updates by flipping correlated groups of spins together, whereas local spin-flip updates modify the configuration only one spin at a time.

2.7 Observables and Definitions

To characterize the thermal phase transition of the Ising model, we measured a set of observables as a function of temperature. Since Monte Carlo samples are generated through a Markov chain, successive measurements are correlated. Therefore, for quantities involving statistical uncertainty (such as the magnetisation), autocorrelation effects must be taken into account when interpreting the results.

Magnetisation

The magnetisation per spin was computed as

$$m = \frac{1}{N} \sum_{i=1}^{N} s_i, \qquad N = L^2.$$

In practice, we often use the mean absolute magnetisation $\langle |m| \rangle$, since the Ising Hamiltonian is symmetric under global spin reversal, and thus the sign of m may fluctuate over long simulation times even in the ordered phase.

Autocorrelation time

Monte Carlo measurements A_t are generally correlated due to the local nature of the update steps. This correlation can be quantified using the normalized autocorrelation function

$$\rho(\Delta) = \frac{\langle A_t A_{t+\Delta} \rangle - \langle A \rangle^2}{\langle A^2 \rangle - \langle A \rangle^2},$$

and the integrated autocorrelation time

$$\tau_{\text{int}} = \frac{1}{2} + \sum_{\Delta=1}^{\infty} \rho(\Delta).$$

A larger τ_{int} implies slower decorrelation and larger statistical errors for a fixed number of Monte Carlo samples. In this work, τ_{int} was estimated from the magnetisation time series using a windowed summation in order to avoid contamination by noise at large lag times.

Specific heat

The specific heat per spin was computed from energy fluctuations using

$$C_V = \frac{\langle E^2 \rangle - \langle E \rangle^2}{NT^2},$$

where E is the total energy of a spin configuration and $N = L^2$. This relation follows from the fluctuation–dissipation framework in the canonical ensemble and provides a convenient estimator of $C_V(T)$ from Monte Carlo samples.

Magnetisation as a function of temperature

To study the thermal phase transition, we computed the equilibrium magnetisation as a function of temperature for multiple lattice sizes. For each temperature, the system was first thermalized by discarding an initial set of Monte Carlo iterations, after which measurements were collected over a measurement window. The mean absolute magnetisation $\langle |m| \rangle$ was then obtained by averaging over the recorded values.

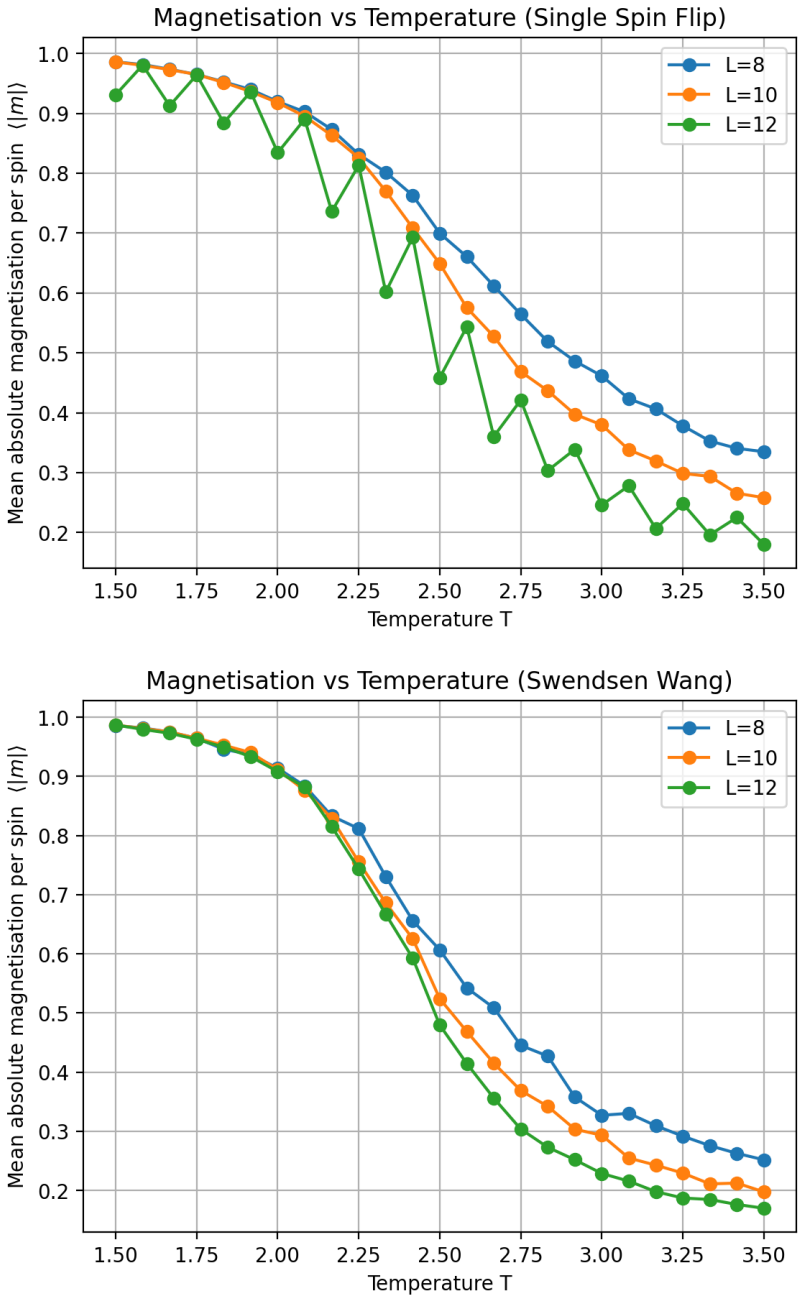


Figure 9: Mean absolute magnetisation per spin $\langle |m| \rangle$ as a function of temperature for different lattice sizes using the Metropolis single spin-flip update (top) and the Swendsen–Wang cluster update (bottom).

Observations. At low temperatures the system is magnetically ordered and $\langle |m| \rangle$ approaches unity, while at high temperatures the system becomes disordered and $\langle |m| \rangle$ decreases towards zero. As the lattice size increases, the drop in $\langle |m| \rangle$ becomes sharper, consistent with reduced finite-size rounding near the transition.

Estimate of the critical temperature. A finite-size estimate of the critical temperature was obtained by identifying the temperature at which $\langle |m| \rangle(T)$ exhibits the steepest decrease. This was implemented numerically by computing the discrete slope $\mathrm{d}\langle |m| \rangle / \mathrm{d}T$ and selecting the temperature of the most negative slope. The resulting estimate is expected to deviate slightly from the thermodynamic limit due to finite-size effects.

Autocorrelation time as a function of temperature

The integrated autocorrelation time $\tau_{\text{int}}(T)$ was computed from the magnetisation time series for both update schemes. The autocorrelation time provides a quantitative measure of how many Monte Carlo iterations are required before successive samples become approximately independent. Large values of τ_{int} indicate slow dynamics and enhanced statistical uncertainties.

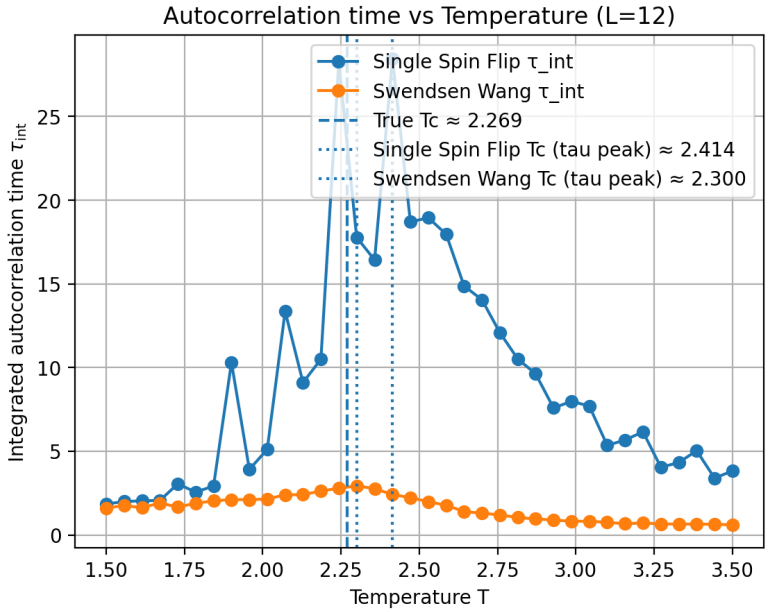


Figure 10: Integrated autocorrelation time τ_{int} as a function of temperature for the Metropolis and Swendsen–Wang updates. Vertical lines indicate the exact critical temperature and finite-size estimates obtained from the peak position.

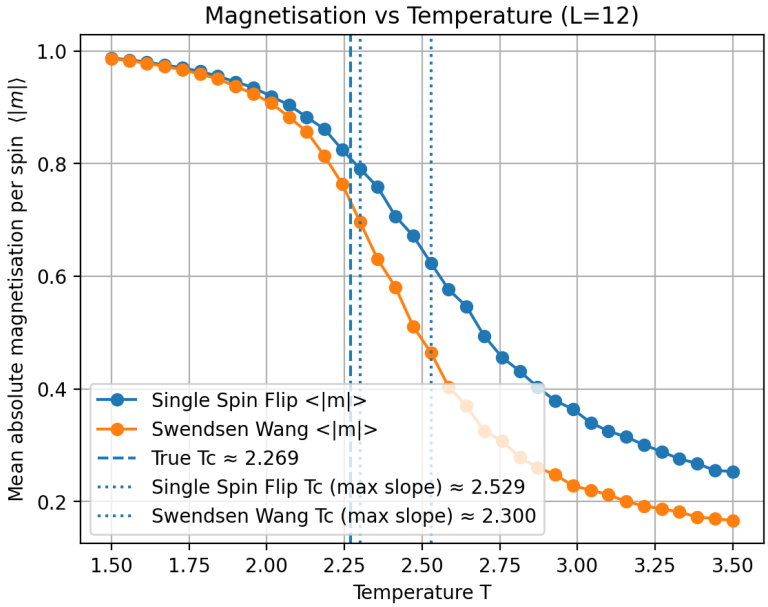


Figure 11: Mean absolute magnetisation per spin $\langle |m| \rangle$ as a function of temperature for $L = 12$, showing the true critical temperature and finite-size estimates obtained from the steepest slope criterion for both the single spin-flip and Swendsen–Wang updates.

Observations. The Metropolis update exhibits significantly larger autocorrelation times than the Swendsen–Wang update, especially in the vicinity of the phase transition. This behaviour is expected, since local spin flips lead to slow relaxation near criticality (critical slowing down), whereas cluster updates reduce these correlations by flipping large correlated domains in a single update.

Estimate of the critical temperature. A second estimate of the critical temperature was obtained by identifying the temperature at which $\tau_{\text{int}}(T)$ reaches its maximum. For a finite lattice, the peak is broadened and shifted, but it still provides a clear signature of critical behaviour.

Specific heat as a function of temperature

To further identify the phase transition, we computed the specific heat per spin $C_V(T)$ from energy fluctuations at equilibrium. For each temperature, measurements of the total energy E were collected after thermalization, and the averages $\langle E \rangle$ and $\langle E^2 \rangle$ were used in the fluctuation formula for C_V.

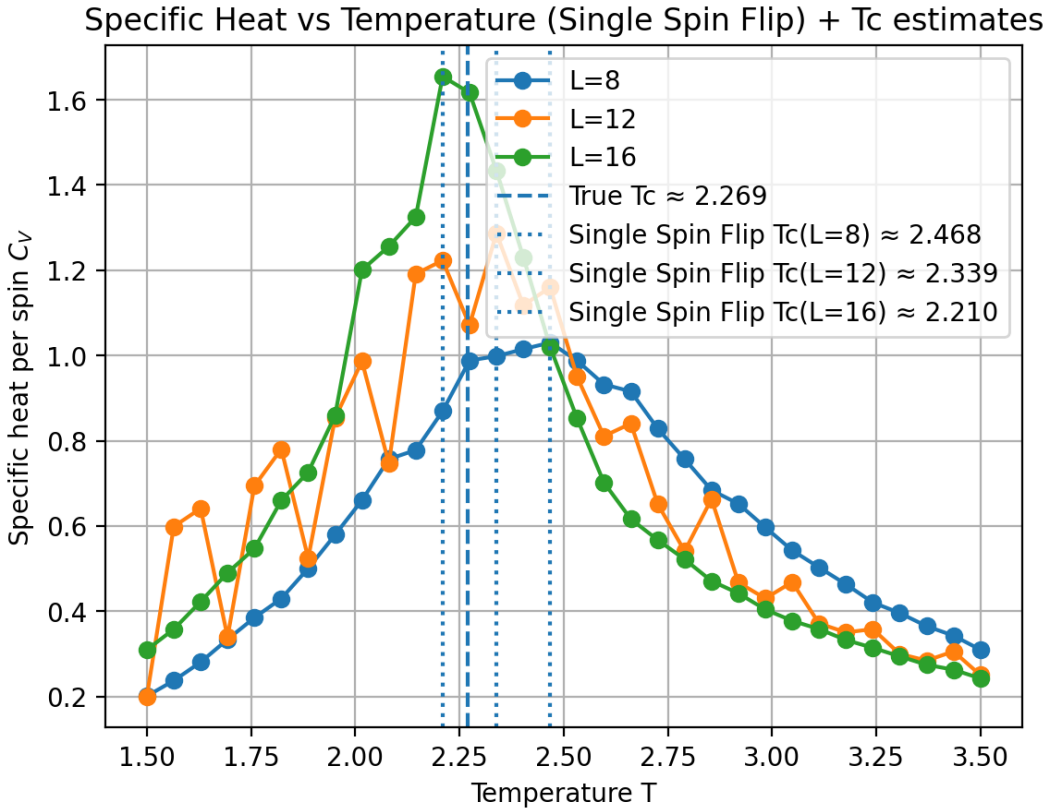


Figure 12: Specific heat per spin C_V as a function of temperature for different lattice sizes using the Metropolis update.

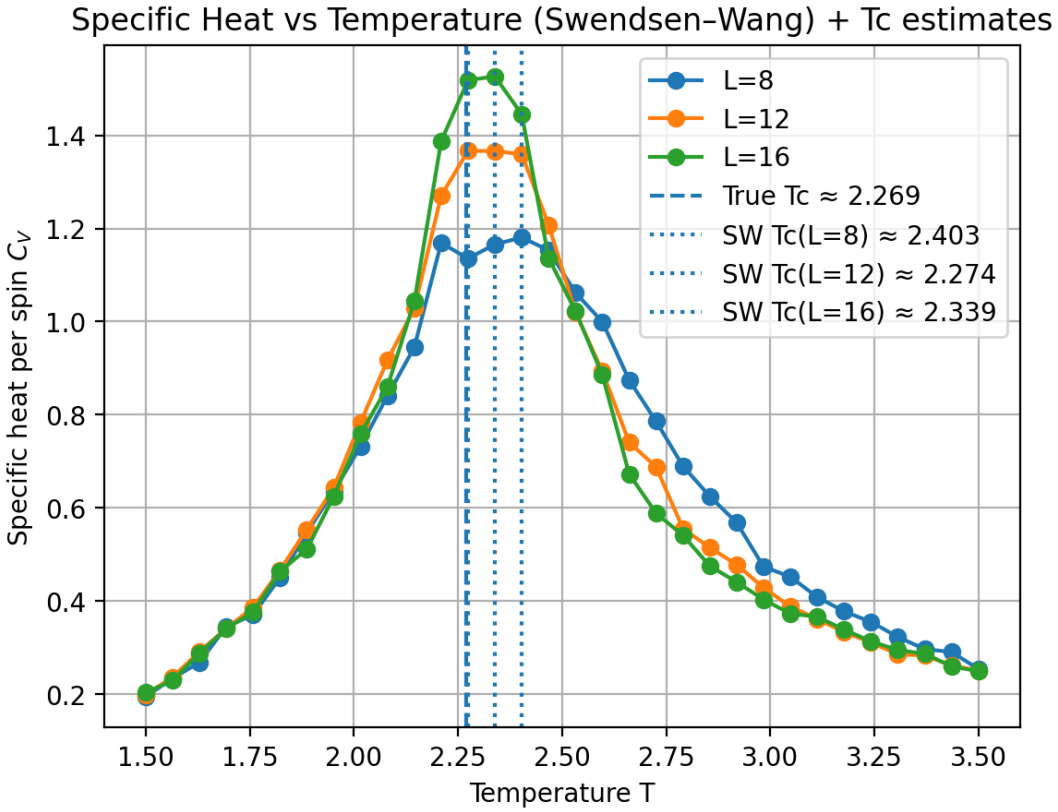


Figure 13: Specific heat per spin C_V as a function of temperature for different lattice sizes using the Swendsen–Wang update.

Observations. In both algorithms, the specific heat exhibits a pronounced peak near the transition region. As the lattice size increases, the peak becomes sharper and higher, providing a clear numerical signature of the phase transition. The peak location offers a further finite-size estimate of the critical temperature.

Estimate of the critical temperature. A third estimate of the critical temperature was obtained by locating the temperature at which $C_V(T)$ reaches its maximum. For a finite lattice, this peak position provides an approximate critical temperature which approaches the thermodynamic limit value as L increases. For reference, the exact critical temperature of the two-dimensional square lattice Ising model with $J = 1$ is $T_c \approx 2.269$.

2.8 Results summary

Estimated critical temperature

Finite-size estimates of the critical temperature were obtained using multiple criteria. In particular, T_c was estimated from the position of the peak in the autocorrelation time $\tau_{\text{int}}(T)$, the steepest drop in $\langle|m|\rangle(T)$, and the peak position of the specific heat $C_V(T)$. The resulting estimates are summarized in Table 2.

Method	Algorithm	Lattice size	Estimated T_c
$\tau_{\text{int}}(T)$ peak	Single spin-flip	$L = 12$	2.4143
$\tau_{\text{int}}(T)$ peak	Swendsen–Wang	$L = 12$	2.3000
Max slope of $\langle\|m\|\rangle(T)$	Single spin-flip	$L = 12$	2.5286
Max slope of $\langle\|m\|\rangle(T)$	Swendsen–Wang	$L = 12$	2.3000
$C_V(T)$ peak	Single spin-flip	$L = 8$	2.4677
$C_V(T)$ peak	Single spin-flip	$L = 12$	2.3387
$C_V(T)$ peak	Single spin-flip	$L = 16$	2.2097
$C_V(T)$ peak	Swendsen–Wang	$L = 8$	2.4032
$C_V(T)$ peak	Swendsen–Wang	$L = 12$	2.2742
$C_V(T)$ peak	Swendsen–Wang	$L = 16$	2.3387

Table 2: Summary of critical temperature estimates obtained from different observables. The exact thermodynamic limit value for the square lattice Ising model (with $J = 1$, $k_B = 1$) is $T_c \approx 2.269$.

Specific heat peak summary

The specific heat peak position and height provide another finite-size indicator of the phase transition. Table 3 summarizes the temperature T_{peak} at which $C_V(T)$ reaches its maximum and the corresponding peak value $C_V^{\max}$, obtained for both update schemes.

Algorithm	Lattice size	T_{peak}	$C_V^{\max}$
Single spin-flip	$L = 8$	2.4677	1.0303
Single spin-flip	$L = 12$	2.3387	1.2868
Single spin-flip	$L = 16$	2.2097	1.6547
Swendsen–Wang	$L = 8$	2.4032	1.1807
Swendsen–Wang	$L = 12$	2.2742	1.3668
Swendsen–Wang	$L = 16$	2.3387	1.5264

Table 3: Peak position and peak height of the specific heat $C_V(T)$ for different lattice sizes and update algorithms.

Observation. The peak in $C_V(T)$ becomes more pronounced with increasing system size, providing a clear signature of critical behaviour in the finite lattice simulations.

2.9 Conclusion

In this phase of the project, we studied the two-dimensional ferromagnetic Ising model on a finite $L \times L$ square lattice using Monte Carlo methods. We implemented both a local single spin-flip Metropolis update and a non-local Swendsen–Wang cluster update, and compared their behaviour across a range of temperatures.

The magnetisation curves $\langle|m|\rangle(T)$ showed clear signatures of a thermal phase transition. At low temperatures the system evolved into an ordered phase with a large magnetisation, while at high temperatures it became disordered and the magnetisation decreased towards zero. As the lattice size increased, the transition region became sharper, consistent with reduced finite-size rounding.

Autocorrelation effects were studied using binning analysis and by estimating the integrated autocorrelation time $\tau_{\mathrm{int}}(T)$. The single spin-flip Metropolis algorithm exhibited significantly larger autocorrelation times, especially near the transition region, indicating slow relaxation due to critical slowing down. In contrast, the Swendsen–Wang update produced much shorter autocorrelation times and reached equilibrium substantially faster, as expected for a cluster algorithm.

Finally, the specific heat $C_V(T)$, computed from energy fluctuations, displayed a pronounced peak near the transition region. The height and sharpness of this peak increased with system size, providing an additional indicator of critical behaviour. Estimates of the critical temperature obtained from magnetisation, autocorrelation time, and specific heat were found to be reasonably consistent with the known value $T_c \approx 2.269$ for the two-dimensional square lattice Ising model (with $J = 1$ and $k_B = 1$), with deviations attributable to finite system sizes and statistical noise. :contentReference[oaicite:0]index=0

Code availability: The source code used to generate the Ising model results and figures in this report is available at the project GitHub repository.

References

- YouTube video: Optimization using Cython.
- Cython Documentation (official).
- E. Luijten, *Introduction to Cluster Monte Carlo Algorithms* (review notes).
- S. Trebst, *Computational Many-Body Physics* lecture series (2025), including autocorrelation and binning analysis (A6).