Cluster Algorithms for the 2D Ising Model

lg653

27 April 2024

Abstract

The 2D Ising Model is simulated using the Metropolis algorithm and cluster algorithms. Measurements of the exponential correlation time of absolute magnetization are performed under a range of temperatures ($0.2 \leq \beta J \leq 0.6$) and lattice widths ($4 \leq L \leq 150$). The phenomenon of critical slowing down is examined by the measurement of dynamic exponents for each algorithm. All three algorithms have been tested under identical equilibration and sampling periods, so the dependence of correlation time on sampling period does not influence the comparison of their performances. We have obtained the dynamic exponents (1.86 ± 0.07), (0.06 ± 0.04) and (0.06 ± 0.07) for the Metropolis, Wolff and Swendsen-Wang algorithms respectively. This suggests that cluster algorithms are significantly better than single-flip algorithms for simulating the Ising Model near the critical point.

1 Introduction

The Ising Model is one of the most scrutinised problems in statistical mechanics. Analytically or computationally, the premise of the solution is simple: evaluate the partition function and all thermodynamic properties of the system follow. For the latter however, the computational effort required for this endeavour grows exponentially with system size. Monte Carlo techniques were invented to address this issue, the first and most famous being the Metropolis algorithm [1]. Since then, other algorithms such as cluster algorithms have been proposed.

These algorithms are grounded in the theoretical framework of Markov Chains. It can be shown that if a candidate algorithm satisfies certain conditions (ergodicity and detailed balance), then it is suitable for sampling the Boltzmann distribution and can thus be used to simulate the Ising Model. The reader is directed to [2] for a comprehensive review of Markov Chain Monte Carlo (MCMC) methods for simulating models in statistical mechanics.

The aim of this project is to investigate the performance of cluster algorithms in comparison to the Metropolis algorithm, by simulation of the 2D Ising Model. We first verify their implementation by observing the qualitative behaviour as the system settles to equilibrium. Thereafter, we run experiments to demonstrate and quantify a phenomenon known as critical slowing down, through the measurement of dynamic exponents.

The next Section gives a brief introduction to the 2D Ising Model, critical slowing down and cluster algorithms, followed by additional notes and notation. Section 3 details the code methodology and experiments to be performed. The results of these experiments are presented in Section 4, which are then discussed in Section 5. Section 6 summarises the conclusions of this project.

2 Theory

2.1 2D Ising Model

The 2D Ising model is a lattice model of a magnet. It is described as a square grid of side length L containing $N = L^2$ spins with binary values $s_i \in \{+1, -1\}$. The temperature T and exchange interaction energy[1] J of the lattice is parameterised together as inverse temperature $\beta J = \frac{J}{k_B T}$, where k_B is Boltzmann's constant. Using Monte Carlo algorithms, we simulate the dynamics of the lattice at a chosen temperature and zero external magnetic field, sampling quantities over a period and taking their time averages. Such quantities include the (relative) magnetisation $m = \frac{1}{N}\sum_i s_i$, the (relative) absolute magnetisation $|m|$ and the lattice energy:

$$\beta E = -\beta J \sum_{\langle ij \rangle} s_i s_j \tag{1}$$

where $\langle ij \rangle$ denotes nearest-neighbour interactions. As predicted by Onsager [3], the 2D Ising model exhibits a continuous phase transition between a disordered paramagnetic phase ($T > T_c$) and an ordered ferromagnetic phase ($T < T_c$) at a critical temperature T_c given by

$$(\beta J)_c = \ln\left(\frac{1+\sqrt{2}}{2}\right) \approx 0.441 \tag{2}$$

2.2 Critical Slowing Down and Dynamic Exponents

As the system approaches T_c, clusters of aligned spins begin to form and the errors of observables grow to considerable magnitudes. The first reason is due to critical fluctuations, an intrinsic property of the Ising model. All MCMC algorithms that sample the Boltzmann distribution will exhibit this effect [2].

The second issue is a property of the algorithm and is known as critical slowing down. It is described as the divergence of a quantity's correlation time τ, which is the typical time scale between statistically independent measurements. As a result, the algorithm becomes severely time inefficient at performing measurements with a level of precision comparable with those away from T_c [2]. Physically speaking, spin clusters take a long time to be destroyed when using single-flip algorithms due to low acceptance probabilities [4]. In the language of Markov chains, the simulation is unable to efficiently explore state space, generating long sequences of correlated samples.

The degree of critical slowing down is measured by the dynamic exponent z, defined by

$$\tau \sim \xi^z \approx L^z \tag{3}$$

where ξ is the correlation length, or the typical size of a spin cluster [2, 4, 5, 6]. In the thermodynamic limit $L \to \infty$, ξ diverges as $T \to T_c$, but caps off at L for finite systems. The higher the dynamic exponent, the faster τ diverges as $T \to T_c$, resulting in slower and inaccurate simulations [2].

2.3 Cluster Algorithms

The dynamic exponent can be reduced significantly by cluster algorithms, which differ chiefly from the Metropolis algorithm in that multiple spins can be flipped in one Monte Carlo (MC) step, a rather effective way of destroying spin clusters and critical correlations [5]. The first of

[1] More generally, the interaction energy can be different in each direction. Here, we consider the simple case $J_1 = J_2$.

these algorithms was introduced by Swendsen and Wang [7], which works by randomly partitioning the entire lattice into clusters and flipping each one with probability $\frac{1}{2}$. Shortly after, Wolff [8] designed a local variation which picks a random spin, grows a cluster by randomly adding neighbouring spins of the same value, and then flips the cluster. The pseudocode of these algorithms are listed in Sections 4.21 and 4.41 of [2], along with proofs of detailed balance and ergodicity being satisfied.

The **actual** dynamic exponent of the Wolff algorithm is related to (3) by an alternative expression (cf. Section 4.3.2 of [2])

$$z = z_{\text{steps}} + \frac{\gamma}{\nu} - d, \quad T \geq T_c \tag{4}$$

where z_{steps} refers to z in (3). For the 2D Ising model, $\gamma = \frac{7}{4}$, $\nu = 1$ are the critical exponents of magnetic susceptibility and correlation length respectively, while $d = 2$ is the dimension of the lattice.

2.4 Notation

2.4.1 Simulation time and exponential correlation time

For the Metropolis algorithm, it is common to measure time in units of MC steps per lattice site, a.k.a lattice sweeps, whereas Wolff and Swendsen-Wang time is measured in MC steps[2]. For this project, we seek to measure the exponential correlation time of absolute magnetization $\tau_{|m|}^{(A)}$, where $A \in \{M, W, SW\}$ denotes Metropolis, Wolff, and Swendsen-Wang respectively. This first requires calculating the time-displaced autocorrelation function $\rho(t)$ of $|m|$:

$$\rho(t) = \int dt' [|m|(t') - \langle m \rangle][|m|(t'+t) - \langle m \rangle] \tag{5}$$

We use the Fast Fourier Transform method (Wiener-Khinchin theorem) as described in Section 3.31 of [2]. Then, $\tau_{|m|}^{(A)}$ is estimated by the first crossing time $\rho(\tau_{|m|}^{(A)}) = \frac{1}{e} \approx 0.37$. Lastly, we denote the dynamic exponent corresponding[3] to $\tau_{|m|}^{(A)}$ using the symbol $z_{|m|}^{(A)}$.

2.4.2 Wolff cluster size and scaled Wolff time

The Wolff algorithm differs from the other two in that it considers only a fraction of the lattice per time step for spin flipping. For fair comparison of correlation times, it is useful to define a scaled Wolff time

$$\text{Scaled } \tau_{|m|}^{(W)} = \frac{\langle n \rangle}{N} \tau_{|m|}^{(W)} \tag{6}$$

where $\langle n \rangle$ is the Wolff cluster size in equilibrium. In practice however, this quantity is not used to calculate dynamic exponents [2] and not related to the actual Wolff dynamic exponent defined in (4). It will still be measured, as we will see that this quantity, rather than the unscaled version, appears to diverge at the critical temperature.

3 Methods

3.1 Implementation of simulation environment

Simulations using MCMC algorithms are very similar to physical experiments. We first initialize a set of simulation parameters and variables and define the data to be collected. The

[2]For brevity, I will collectively refer to time steps for both Metropolis and cluster algorithms in units of sweeps, unless otherwise stated.

[3]There exist other types of dynamic exponents in the literature [9, 10]. For example, there is a dynamic exponent corresponding to the integrated correlation time of energy $\tau_{E,int}^{(A)}$.

experiment begins with an equilibration phase, running the chosen algorithm and allowing the system to reach a steady state. Then, during the sampling phase, magnetization and energy are measured at each time step. At this point, we can either choose to rerun the experiment for error calculation, or proceed with the data analysis and visualization.

With this philosophy in mind, the code was designed with two main components: the `mcmc` and `plotting` modules. The functions defined in `mcmc` run the simulation according to the user's desired algorithm and experimental parameters, then store the raw data in timestamped files. Some manual processing of the data is required. Then, files can be loaded and plotted automatically using the functions defined in `plotting`. Wherever possible, the code was refactored in favour of clarity and elimination of redundancy. The Python modules are then imported into Jupyter notebooks where the following experiments are carried out.

3.2 Dynamics of MC algorithms

The goals of this experiment are

1. To observe the time evolution of the system and its quantities at three temperature regimes: $T \ll T_c$, $T \sim T_c$ and $T \gg T_c$, where T_c is given by (2).
2. To estimate equilibration time τ_{eq} for the subsequent experiments.

For each algorithm, we initialize a random lattice configuration with $L = 150$ and run the simulation for at least $N = 11250$ sweeps, sampling m, $|m|$, βE and the lattice configuration each time the algorithm is called. Thereafter, the time evolution of these observables are plotted. This procedure is performed for three temperatures $\beta J = \{0.01, 0.44, 10\}$.

A rough estimate of τ_{eq} was determined by visual inspection of the βE and $|m|$ traces. This is used to verify the suggestion that running the equilibration phase for $\geq N$ sweeps is sufficient to reach equilibrium [2, 11]. For reliable data, it is crucial that we achieve equilibrium before starting to sample. Unfortunately, there is no definite way (other than equilibrating for longer times) to ensure this, due to the existence of metastable states [2].

3.3 Critical slowing down

In this experiment, we demonstrate the phenomenon of critical slowing down by measuring $\tau^{(A)}_{|m|}$ for temperatures $0.2 \leq \beta J \leq 0.6$. For each algorithm, we initialize a random lattice configuration with $L = 25$, running the simulation for at least $N = 625$ sweeps. Two approaches to sampling correlation time are considered:

1. Averaging over runs: for a given temperature, we perform multiple short runs to sample a distribution of correlation times
2. Averaging over batches: for a given temperature, we perform one long run and split the sampling phase into batches, sampling a correlation time from each batch and hence a distribution of correlation times [6]

From the distribution, we report the sample mean and standard deviation of $\tau^{(A)}_{|m|}$. To balance between precision and simulation time, we used the first approach for smaller lattices $L \leq 25$, sampling for 10 - 50 runs. For larger lattices, the second approach was tested and the sampled data was split into 100 batches.

To increase the chances of sampling equilibrium values, the lattice is initialized once and reused between runs and temperatures throughout the simulation. This is justified by the argument that closely spaced temperatures have similar equilibrium states [2]. The data are saved in a comma separated value file (CSV) and a graph of $\tau^{(A)}_{|m|}$ against βJ is plotted.

3.3.1 Measuring $\langle n \rangle$ and $\tau_{|m|}^{(W)}$

The measurement of Wolff cluster size can actually be incorporated into the procedure described above, following identical steps as correlation time sampling. The size of the cluster for each Wolff step can be measured and thus a time average $\langle n \rangle$ can be computed. The scaled Wolff time immediately follows from (6) and can be plotted against βJ.

However, the main code presented does not have this functionality as priority was placed on code clarity. Separate functions which measure $\langle n \rangle$ and $\tau_{|m|}^{(W)}$ were defined to reduce the number of if/else statements in the main code, which would otherwise reduce code readability. Further work is needed to integrate these calculations into the main code more cleanly.

3.4 Dynamic exponent determination

For the last experiment, the dynamic exponents are calculated. We simulate lattices with widths $4 \leq L \leq 150$ at a temperature slightly above T_c, for at least N sweeps. The two approaches of sampling $\tau_{|m|}^{(A)}$ mentioned in Section 3.3 also apply here. Since L is now the independent variable, the lattice can only be reused between runs (when Approach 1 is used), and not between L values. The data is collected over multiple sessions and manually compiled into one CSV file for each algorithm. Then, a graph of $\ln(\tau_{|m|}^{(A)})$ against $\ln(L)$ is plotted and is least-squares fitted according to (3). Finally, $z_{|m|}^{(A)}$ and the actual Wolff dynamic exponent (4) are extracted from the fit.

4 Results

4.1 Time experiments

The lattice evolution, absolute magnetisation and energy traces for each algorithm are presented in Figures 1-6. By visually inspecting the plots, we verify that starting from a randomly initialized lattice, an equilibration period of N sweeps is a reasonable lower limit. The equilibrium state at each temperature regime also agrees with predictions from the Ising model. In the high temperature limit $\beta J \ll (\beta J)_c$, spins orient themselves randomly as thermal fluctuations dominate. Near the critical temperature $\beta J \sim (\beta J)_c$, the equilibrium state favours cluster formation as the interaction energy becomes comparable to thermal fluctuations. In the low temperature limit $\beta J \gg (\beta J)_c$, spins tend to align into an ordered lattice as the interaction energy dominates. These observations give us confidence in the implementation of the algorithms for the subsequent experiments.

4.2 Temperature experiments

The two methods for sampling correlation time were thoroughly tested and compared to literature values. As will be discussed in Section 5.1, it was determined that Approach 2 is not a rigorous method for sampling and error calculation. Henceforth, the results presented all use Approach 1 for sampling correlation time. Figures 7-9 depict the graphs of autocorrelation times against temperature. Although all three algorithms exhibit critical slowing down, the magnitude of $\tau_{|m|}^{(M)}$ was not expected. The calculation of correlation time is examined more carefully in Section 5.1. After addressing this problem, the performance of the algorithms is compared in Section 5.2. Nevertheless, the qualitative behaviour of correlation time and variance rising near the critical temperature is consistent with theory.

4.3 Lattice width experiments

Figures 10-12 depict the graphs of correlation time against lattice widths in logarithmic scale. Each data point was sampled for 50 runs. The equilibration and sampling periods were $0.6N$ and $0.4N$ respectively. However, due to the sheer computation time required, lattice sizes were reduced to $L \leq 60$ for the Metropolis and Swendsen-Wang algorithms. The Wolff algorithm was able to compute the range $4 \leq L \leq 150$ in a reasonable time (11 data points in 16 hours on a commercial laptop). Table 1 summarises the computed dynamic exponents of each algorithm.

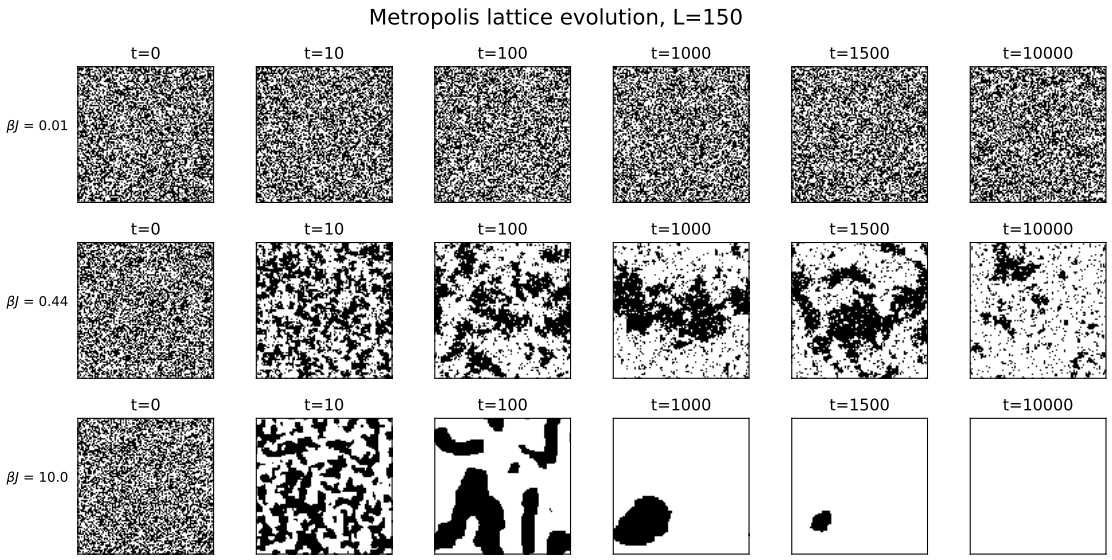


Figure 1: Time evolution of a 150×150 randomly initialized lattice in three temperature regimes, using the Metropolis algorithm. Total simulation time was 11250 sweeps.

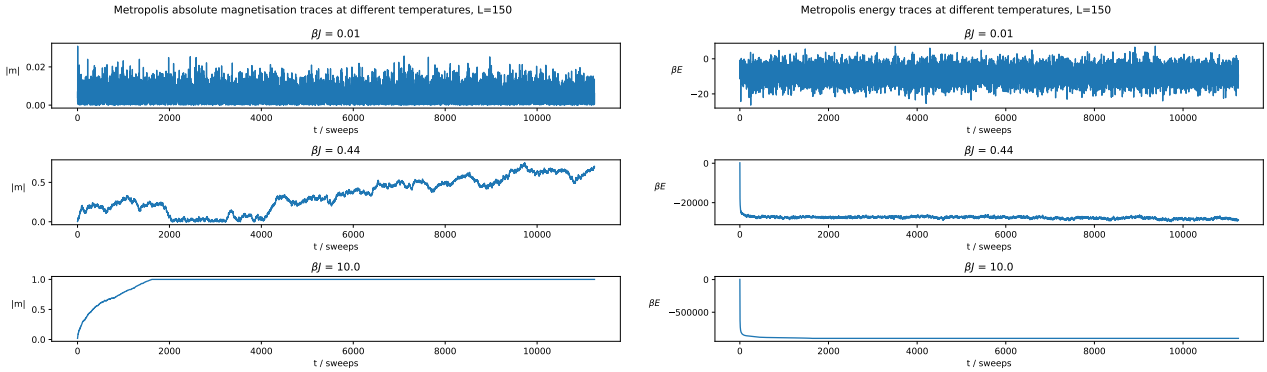


Figure 2: Measurement traces of (a) absolute magnetisation (b) energy for the Metropolis algorithm. The equilibration times for all three temperatures, starting from a randomly initialized lattice, was much less than $N = 22500$ sweeps. The maximum τ_{eq} was about 1800 sweeps, as deduced by $|m|(t)$ for $\beta J = 10$.

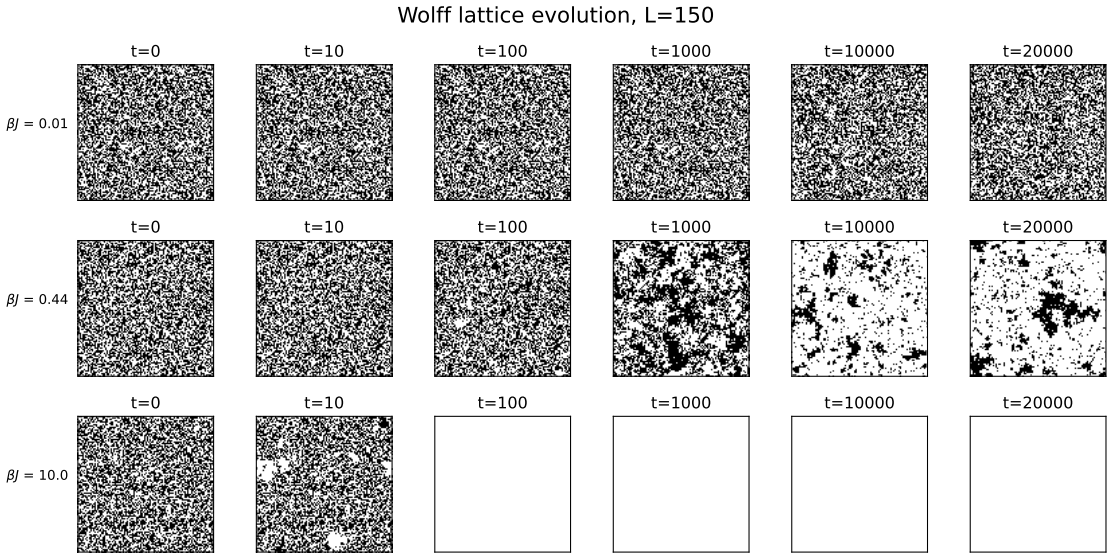
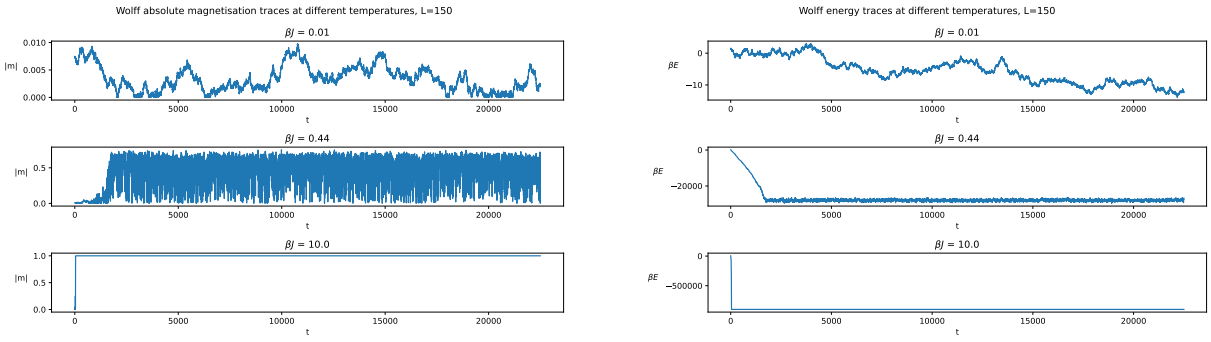


Figure 3: Time evolution of a 150×150 randomly initialized lattice in three temperature regimes, using the Wolff algorithm. Total simulation time was 22500 MC steps



(a) (b)

Figure 4: Measurement traces of (a) absolute magnetisation (b) energy for the Wolff algorithm. The equilibration time was much less than $N = 22500$ MC steps, with a maximum value of 2400 MC steps as seen by the traces for $\beta J = 0.44$.

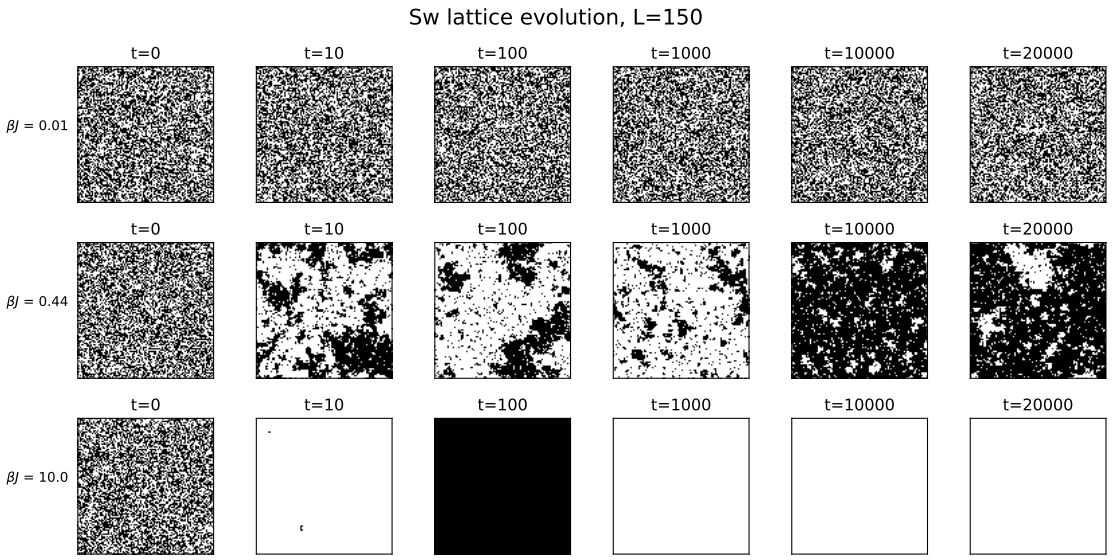
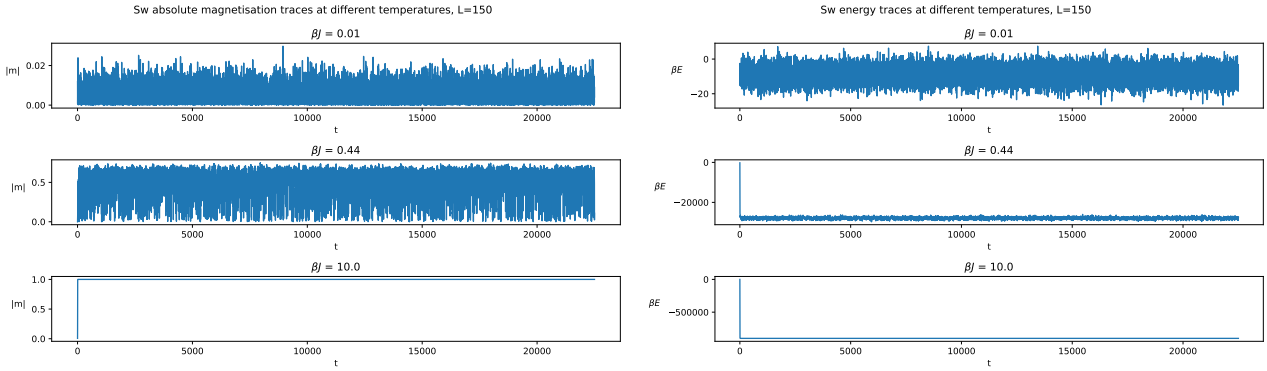


Figure 5: Time evolution of a 150×150 randomly initialized lattice in three temperature regimes, using the Swendsen-Wang algorithm. Total simulation time was 22500 MC steps.



(a) (b)

Figure 6: Measurement traces of (a) absolute magnetisation (b) energy for the Swendsen-Wang algorithm. From a random configuration, the system settles into equilibrium almost immediately in less than 50 MC steps.

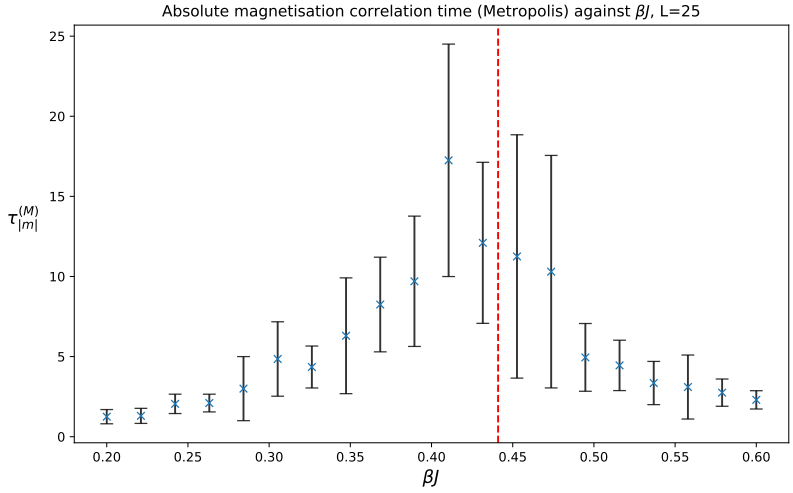
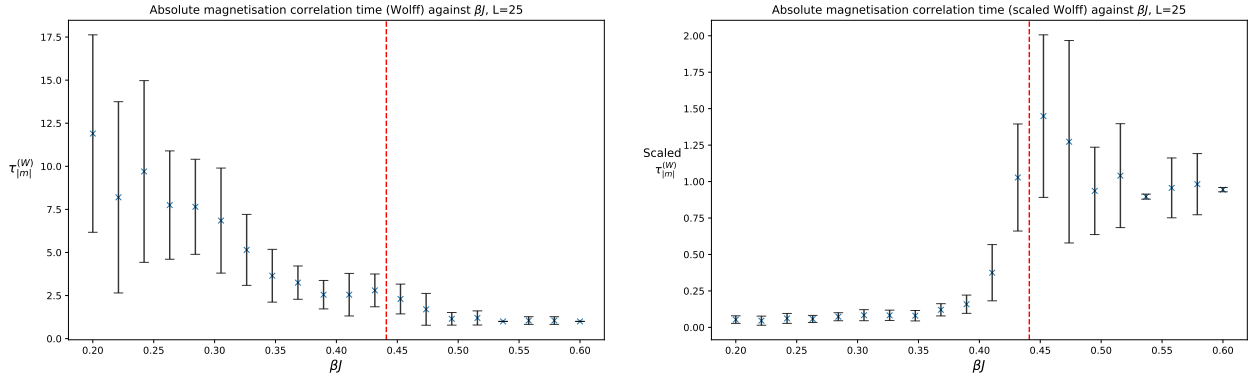


Figure 7: Critical slowing down of the Metropolis algorithm, using a randomly initialized $L = 25$ lattice. The red dotted line denotes the critical point. Each data point was sampled for 20 runs, with equilibration and sampling periods lasting 500 and 125 sweeps respectively. Then, the sample mean and standard deviation of $\tau^{(M)}_{|m|}$ is computed. The magnitudes of $\tau^{(M)}_{|m|}$ is not expected when compared to results elsewhere [2, 11].



(a) (b)

Figure 8: Critical slowing down of the Wolff algorithm, using a randomly initialized $L = 25$ lattice. The red dotted line denotes the critical point. Each data point was sampled for 20 runs, with equilibration and sampling periods lasting 500 and 125 sweeps respectively. Then, the sample mean and standard deviation of $\tau^{(W)}_{|m|}$ and $\langle n \rangle$ are computed. (a) Unscaled correlation time does not peak at the critical point. (b) Scaled correlation time exhibits critical slowing down.

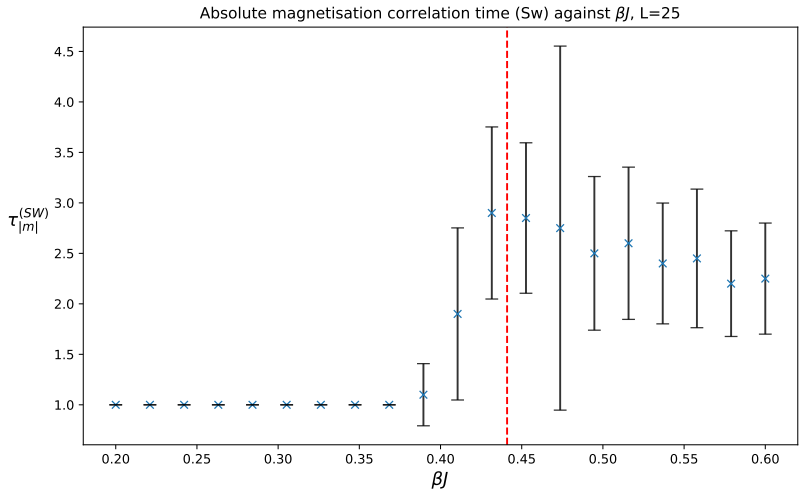


Figure 9: Critical slowing down of the Swendsen-Wang algorithm, using a randomly initialized $L = 25$ lattice. The red dotted line denotes the critical point. Each data point was sampled for 20 runs, with equilibration and sampling periods lasting 500 and 125 sweeps respectively. Then, the sample mean and standard deviation of $\tau^{(SW)}_{|m|}$ is computed.

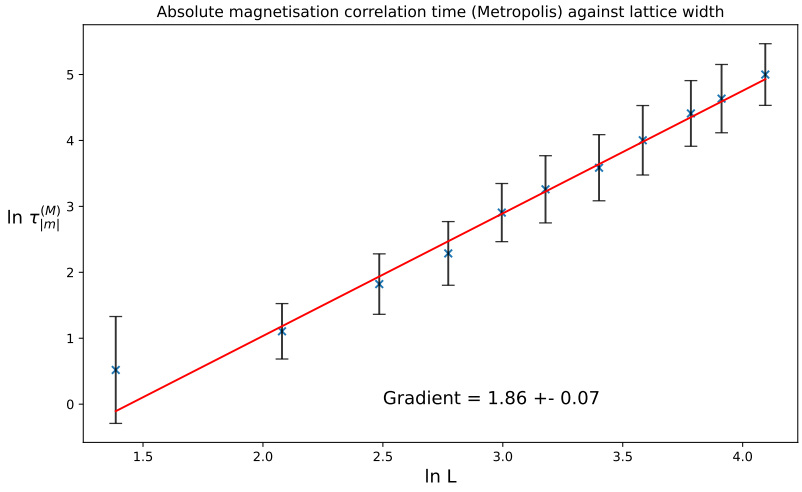


Figure 10: Estimating $z^{(M)}_{|m|}$ by fitting correlation time against lattice width to a power law, for a range of lattice widths $4 \leq L \leq 60$. The fitting procedure was done using `scipy.optimize.curve_fit`, where the errors in correlation time are passed into the `sigma` argument.

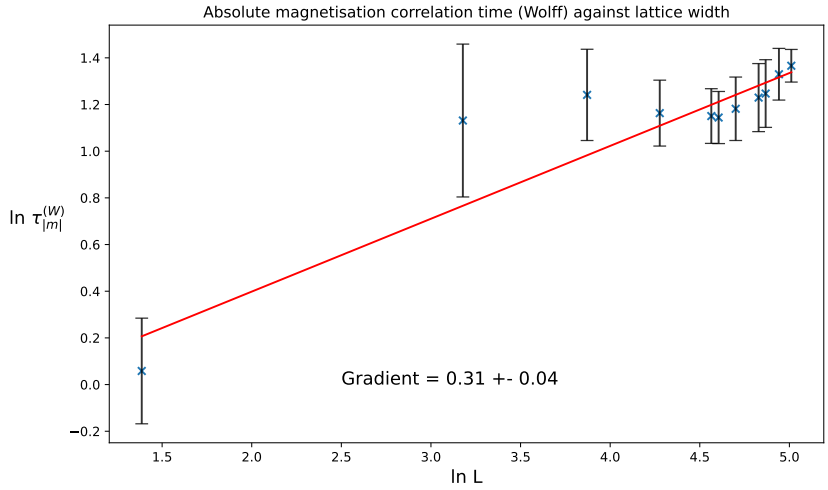


Figure 11: Estimating $z^{(W)}_{|m|}$ by fitting correlation time against lattice width to a power law, for a range of lattice widths $4 \leq L \leq 150$. The fitting procedure was done using `scipy.optimize.curve_fit`, where the errors in correlation time are passed into the `sigma` argument. Then, the actual dynamic exponent of (4) is calculated.

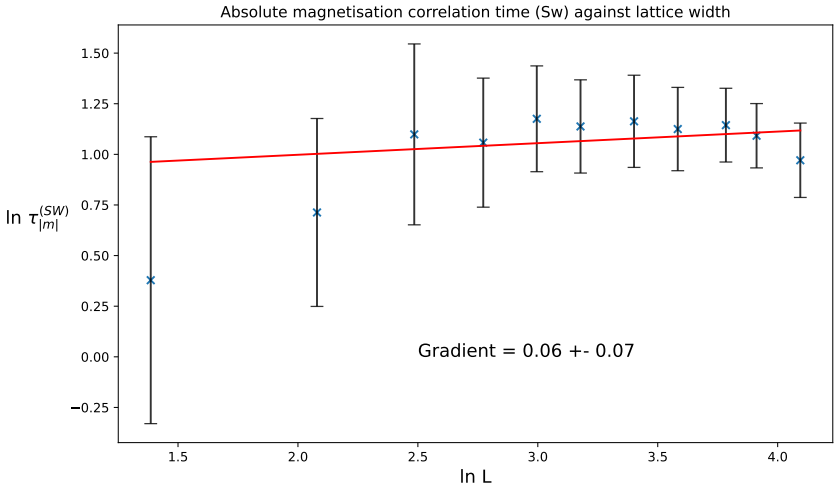


Figure 12: Estimating $z^{(SW)}_{|m|}$ by fitting correlation time against lattice width to a power law, for a range of lattice widths $4 \leq L \leq 60$. The fitting procedure was done using `scipy.optimize.curve_fit`, where the errors in correlation time are passed into the `sigma` argument.

	Dynamic exponent $z^{(A)}_{\lvert m\rvert}$
Metropolis	1.86 ± 0.07
Wolff	0.06 ± 0.04
Swendsen-Wang	0.06 ± 0.07

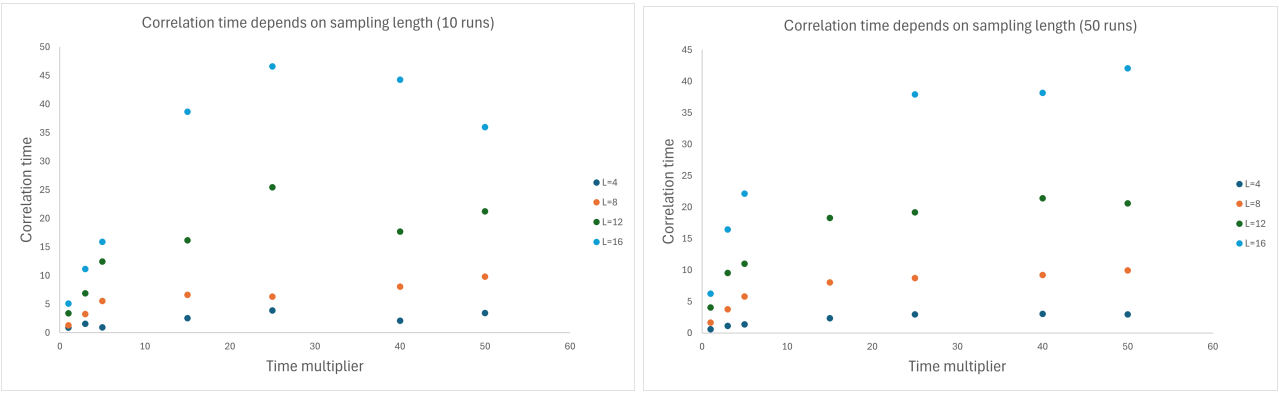
Table 1: Summary of dynamic exponents derived from exponential correlation time of absolute magnetisation.

5 Discussion

5.1 Computation of correlation time

One major issue that was observed when performing the experiments described in Sections 3.3 and 3.4 was the dependence of correlation time on the sampling period t_{sample}. In Figure 13, the Metropolis correlation time is plotted as a function of t_{sample}. $\tau^{(M)}_{\lvert m\rvert}$ appears to grow significantly with t_{sample} and this effect is consistent across four different lattice widths. However, the percentage error of these points is around $50-60\%$ on average, though this could be reduced by increasing the number of runs.

Although we can still observe the qualitative behaviour of critical slowing down for each algorithm, this issue complicates the interpretation when comparing the performance of the algorithms. It also invalidates the batching method for computing $\tau^{(A)}_{\lvert m\rvert}$, since the number of batches alters t_{sample} arbitrarily, in turn causing $\tau^{(A)}_{\lvert m\rvert}$ to vary. In summary, all measurements of correlation time should have the same sampling period for a fair comparison between algorithms.



(a) (b)

Figure 13: Metropolis correlation time plotted against sampling length measured in time multipliers. A time multiplier of unity denotes an equilibration and sampling period of N and $0.25N$ sweeps respectively. The survey is performed for (a) 10 runs (b) 50 runs, for lattice widths $L = 4, 8, 12, 16$. The error bars have been omitted because they are quite large ($\sim 50 - 60\%$ percentage error on average).

5.2 Performance of algorithms at different temperature regimes

Noting the final point of Section 5.1, we can fairly compare the performance of the algorithms, as all three have been tested under identical conditions. Referring to Figures 7, 8*b* and 9,

we observe that the cluster algorithms perform better than the Metropolis algorithm near the critical point due to their smaller correlation times. In the high and low temperature limits, the Metropolis algorithm has a comparable correlation time, but should take less CPU time overall because it is simpler to execute than cluster algorithms. This highlights the collective value of single-flip and cluster algorithms in simulating statistical models across all temperatures.

5.3 Evaluation of dynamic exponents

Monte Carlo simulations in statistical mechanics are notorious for being data and computationally intensive, and many of them are carried out in high performance computing clusters [10, 12]. For example, recent work by Kanbur [12] uses $10^4 - 10^5$ runs per data point with a sampling period of 50000 sweeps, up to lattice sizes of $L = 512$. In comparison, the present work samples a factor of $10^3 - 10^4$ times less, with $\tau^{(A)}_{|m|}$ percentage errors of 50% on average. Hence, the dynamic exponents presented are merely rough estimates. Many more runs and larger lattice sizes need to be sampled to obtain a result in agreement with literature values.

Historically, the value of the dynamic exponents is widely debated. Studies have asserted $z \approx 2.15$ for the Metropolis algorithm [2, 12] and $z \approx 0 - 0.25$ for the Swendsen-Wang and Wolff algorithms [10, 12, 13]. When fitting the data of cluster algorithms, lattice widths only above a certain threshold (e.g. $L \geq 48$ for [12]) are used because of the non-linearity of small lattices. The location at which data is truncated seems arbitrary, so the present work has used all the data points for fitting. Lastly, although we have not explored this, the relationship between dynamic exponents of different observables is not well understood.

5.4 Pre-equilibrating larger lattices for longer sampling periods

As noted in Section 4.3, the computation time for the Metropolis and Swendsen-Wang algorithms grow more quickly with lattice size than the Wolff algorithm. For instance, the Metropolis algorithm required 2.5 and 4.25 hours for $L = 50$ and $L = 60$ respectively. Meanwhile, the Wolff algorithm samples $L = 110$ in about 2 hours.

In thermal simulations, the equilibrium state corresponds to the lowest energy configuration and depends solely on the temperature, irrespective of the algorithm chosen. Moreover, because we are only interested with sampling equilibrium quantities, the question of how the lattice reaches equilibrium is not particularly important.

Hence, a potential improvement could be to first generate the equilibrium states at $T \sim T_c$ with the faster Wolff algorithm, saving it in storage. Then, we can retrieve the pre-equilibrated lattice and proceed directly to sampling correlation times with the other algorithms. This reallocation of the total simulation time towards sampling could improve the precision of $\tau^{(A)}_{|m|}$ and $z^{(A)}_{|m|}$.

6 Conclusion

In summary, we have compared the performance of cluster algorithms with the Metropolis algorithm in simulating the 2D Ising Model. Cluster algorithms are computationally more efficient at generating independent configurations for temperatures near the critical point, as seen by their smaller correlation times. The problem of critical slowing down is also less severe because of their smaller dynamic exponents. A rough measurement of these exponents has been performed and we have obtained the values (1.86 ± 0.07), (0.06 ± 0.04) and (0.06 ± 0.07) for the Metropolis, Wolff and Swendsen-Wang algorithms respectively. Although the sample sizes are $10^3 - 10^4$ times smaller than advanced studies, we have obtained results that are within an order of magnitude of literature values.

A Appendix

A.1 Code listing

The following is a description of the code repository /project

- /project
  - /bin: Contains modules which are imported in Jupyter notebooks
    - mcmc.py: Monte Carlo algorithms (Metropolis, Wolff, Swendsen-Wang), as well as experiment functions (e.g. `time_experiment` and `temperature_experiment_runs`) which are called upon for simulation and data collection and storage
    - plotting.py: Functions used to load and plot the data collected from simulations
    - helpers.py: Helper functions used by `mcmc` and `plotting`
  - /src: Contains the interactive environments in which the `mcmc` and `plotting` modules are imported. After defining parameters e.g. range of independent variable, choice of algorithm, etc, the functions are called to generate data or results.
    - Time Experiments.ipynb /.py: Simulates the 2D Ising model for a range of temperatures and simulation times; plots the time evolution of observables and lattice.
    - Temperature Experiments.ipynb /.py: Simulates the 2D Ising model, measuring correlation time for a range of temperatures. Also measures Wolff cluster sizes and scaled Wolff time.
    - Width Experiments.ipynb /.py: Simulates the 2D Ising model at the critical temperature, measuring correlation time for a range of lattice widths
    - Data Analysis and Plotting.ipynb /.py: Interactive environment for analysing and plotting the saved data from the three experiments above.
  - /data: Contains the data collected from the simulations defined in mcmc.py.
  - /results: Contains the plots produced by plotting.py, as well as from Excel analysis.

References

[1] Nicholas Metropolis et al. "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[2] Mark EJ Newman and Gerard T Barkema. *Monte Carlo methods in statistical physics*. Clarendon Press, 1999. Chap. 1,2,3,4.

[3] Lars Onsager. "Crystal statistics. I. A two-dimensional model with an order-disorder transition". In: *Physical Review* 65.3-4 (1944), p. 117.

[4] J-C Walter and GT Barkema. "An introduction to Monte Carlo methods". In: *Physica A: Statistical Mechanics and its Applications* 418 (2015), pp. 78–87.

[5] Erik Luijten. "Introduction to cluster Monte Carlo algorithms". In: *Computer Simulations in Condensed Matter Systems: From Materials to Chemical Biology Volume 1*. Springer, 2006, pp. 13–38.

[6] Timothy Budd. *Monte Carlo Techniques*. 2023.

[7] Robert H Swendsen and Jian-Sheng Wang. "Nonuniversal critical dynamics in Monte Carlo simulations". In: *Physical Review Letters* 58.2 (1987), p. 86.

[8] Ulli Wolff. "Collective Monte Carlo updating for spin systems". In: *Physical Review Letters* 62.4 (1989), p. 361.

[9] Paul D Coddington and Clive F Baillie. "Empirical relations between static and dynamic exponents for Ising model cluster algorithms". In: *Physical Review Letters* 68.7 (1992), p. 962.

[10] Jianqing Du, Bo Zheng, and Jian-Sheng Wang. "Dynamic critical exponents for Swendsen–Wang and Wolff algorithms obtained by a nonequilibrium relaxation method". In: *Journal of Statistical Mechanics: Theory and Experiment* 2006.05 (2006), P05004.

[11] Robert Knegjens. *Simulation of the 2D Ising Model*. 2008.

[12] Ulvi Kanbur and Zeynep Demir Vatansever. "Critical dynamics of cluster algorithms in the random-bond Ising model". In: *Physical Review E* 109.2 (2024), p. 024140.

[13] Semra Gündüç et al. "A study of dynamic finite size scaling behavior of the scaling functions—calculation of dynamic critical index of Wolff algorithm". In: *Computer physics communications* 166.1 (2005), pp. 1–7.