TU Delft

Computational Physics

Continuous Phase Transition in the 2D Ising Model with Lattice Impurities: A Comparative Study of Metropolis and Wolff Algorithms

Authors:
Petra Navarcikova *(6245129)*
Lucio F. Vecchia *(5476852)*
Arianna Pasterkamp *(5450845)*

Teaching Assistant:
Cem Nurlu

May 13, 2025

Abstract

In this report, Monte Carlo simulations were employed to investigate the critical behaviour of a two-dimensional ferromagnetic system modeled by the Ising lattice. Spin configurations were propagated using both the Metropolis and Wolff algorithms under periodic boundary conditions. To improve numerical robustness and comparability, all observables were non-dimensionalized. Simulations were performed for lattice sizes of up to L=15 and across a wide range of temperatures to study phase transitions. Magnetization, internal energy, specific heat and susceptibility were selected as key observables, each evaluated using block bootstrapping to estimate uncertainties. The impact of diluted site impurities was further exploredby randomly removing spin contributions from the lattice. Results demonstrated a shift in the critical temperature with increasing impurity concentration, while comparisons between algorithms highlighted the superior efficiency of the Wolff method near phase transitions. Overall, the simulations qualitatively aligned with theoretical and numerical literature, confirming that Monte Carlo methods effectively capture thermodynamic behaviour in spin systems.

1 Introduction

Nowadays, ferromagnets are used in every aspect of life. They can be found in electromechanical technologies such as transformers, electrical motors and generators. Additionally, ferromagnetic materials are utilized for information storage because they have bistable states that can be seen as "1" and "0." In magnetic-based memory devices such as magnetic tapes, magnetic hard drives, and magnetic random access memory, information is stored by switching the magnetic state using a magnetic field that is generated by electric currents [1]. Unlike, paramagnetism, ferromagnetic materials can retain their magnetic properties when the magnetic field is removed. In other words, paramagnetic materials like aluminum or platinum become magnetized in a magnetic field but their magnetization disappears when the field is removed [2]. Nevertheless, the magnetic properties of a material depends on its temperature hence, a paramagnetic material can transition into a ferromagnetic and vice versa. This phenomena is referred to as a phase change.

In this investigation, the Ising model was used. Named for German physicist Ernst Ising, it was introduced to study magnetic dipole moments of atoms in statistical physics. The magnetic dipole moments of atomic spins can be in one of two states, $+1$ or -1. The spins, represented as discrete variables, are arranged in a graph, typically a lattice graph, and this model allows each spin to interact with the nearest neighbors [3]. Therefore, the model allows for the identification of phase transitions.

As a results, critical temperature i.e the temperature at which phase transition occurs is a widely studied phenomena. Moreover, the magnetization of a material is related to the spin of the particles that make up the material. As a consequence, the models utilized to simulate the magnetic object must be of a probabilistic nature due to the thermodynamical behaviour of the system. Hence, many models utilized to simulate this magnetic behavior rely on the Monte Carlo algorithm. The Monte Carlo methods are essential for studying the Ising model because they allow efficient sampling of spin configurations from the Boltzmann distribution. This is crucial since the number of possible states grows exponentially with system size, making exact calculations impractical for all but the smallest lattices.

In practice, materials contain impurities (imperfections) that can potentially affect the system. In other words, the Ising model simulates a perfect lattice meaning that the results obtained from the simulation may deviate slightly from practical measurements. As a consequence, this investigation aims to research the effect these impurities have in the magnetization of materials. More specifically, the effect of diluted site impurities will be researched, a common type of imperfection that can occur when atoms are missing from the lattice. Additionally, the number of impurities a material contains and their location can vary, meaning that the concentration of the impurity could potentially have an effect on observables such as magnetization. Therefore, this investigation aims to answer the following research question:

How does the diluted site impurity concentration affect the critical temperature of a magnetic material utilizing the Ising model?

2 Theoretical background

In this section, the necessary concepts to understand magnetic phase transitions dynamics will be explained. Firstly, the 2D Ising model is introduced together with definitions of relevant observales; average magnetization per spin, internal energy per spin, specific heat capacity and magnetic susceptobility. Next, the mathematical framework necessary for simulating thermodynamic quantities is explained; The Monte Carlo method and importance sampling. The section concludes with descriptions of a local update Metropolis algorithm and sigle cluster Wolff algorithm.

2.1 The 2D Ising Model and observables

Mention that $\mathcal{H}$ is the hamiltonian, and β is the inverse temp. please call the lattice size L

The Ising model is one of the most important tools used to describe magnetic systems at the microscopic level. It was originally introduced to model the behavior of ferromagnetic materials and is particularly useful for identifying when a phase transition occurs. In this model, each atomic spin is represented by a discrete variable that can take one of two values: $+1$ or -1 (see figure 1. These spins are placed on the sites of a lattice under periodic boundary conditions and are allowed to interact with their closest neighbors. The total energy of the system, known as the Hamiltonian, is calculated as

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} s_i s_j - H \sum_i s_i, \tag{1}$$

where J is the interaction strength between neighboring spins, H is the external magnetic field, and s_i is the spin at site i. The sum over $\langle i, j \rangle$ represents a sum over all neighboring pairs of spins. When $J > 0$, the interaction is ferromagnetic, meaning the system favors spins that align in the same direction.

An interesting aspect of this model is its ability to capture the concept of critical temperature, denoted T_c. Below this temperature, the system exhibits spontaneous magnetization, where most spins align without any external influence. Above T_c, thermal fluctuations dominate, and the system becomes disordered. For the 2D square lattice with no external magnetic field, the critical temperature can be calculated exactly as

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

At this point, the system undergoes a phase transition that is continuous (second-order), meaning properties like magnetization change smoothly but with infinite slope.

To understand this behavior in more detail, one must study physical observables that describe the macroscopic state of the system. One of the most important quantities is the average magnetization per spin, given by

$$\langle M \rangle = \frac{1}{N} \sum_i s_i. \tag{3}$$

This observable conveys how aligned the system is overall. At low temperatures, M is close to 1 or -1, while at high temperatures it approaches zero. Another important observable is the internal energy per spin,

$$E = \frac{1}{N}\langle \mathcal{H} \rangle, \tag{4}$$

which provides information on how much energy the system carries at a given temperature. As the temperature increases, the energy increases as well, but its rate of change contains critical information.

This leads to the heat capacity, which is a measure of how much the energy fluctuates. This can be easily inferred from eq 4 where the Energy is related to the Hamiltonian while the heat capacity is related to its variance.

$$C = \frac{1}{Nk_BT^2}\left(\langle \mathcal{H}^2 \rangle - \langle \mathcal{H} \rangle^2\right). \tag{5}$$

Near the critical temperature, the system experiences large fluctuations in energy, and this causes the heat capacity to peak. Similarly, the magnetic susceptibility tells us how responsive the system is to a magnetic field and is given by

$$\chi = \frac{1}{Nk_BT}\left(\langle M^2 \rangle - \langle M \rangle^2\right). \tag{6}$$

similarly to the heat capacity, susceptibility peaks at the critical temperature due to strong fluctuations in magnetization. These peaks are used to numerically estimate T_c in simulations. Note that the susceptibility is dependent on the variance of magnetization. Hence, the two observables are related.

By tracking how observables behave as temperature varies, it becomes possible to locate the critical point and explore how factors such as impurities affect the magnetic properties of the system.

2.2 Impurities

Real-world materials are rarely perfect. They contain various types of imperfections such as vacancies, substitutions and dislocations that can significantly influence their physical properties. In the context of magnetic systems modeled by the Ising model, one commonly studied imperfection is the diluted site impurity [4].

Diluted site impurities are introduced by modifying certain lattice sites such that they no longer contribute to the spin interactions. This is achieved by setting the spin variable to zero at those sites, i.e., $s_i = 0$. These sites are effectively removed from the Hamiltonian (see eq 1). In this modified system, the diluted sites do not participate in local ordering and break the connectivity of spin-spin interactions across the lattice [4].
From a statistical mechanics perspective, diluted systems represent a form of quenched disorder, where the random distribution of inactive sites is fixed in time. Unlike thermal fluctuations, this type of disorder does not evolve dynamically but instead alters the energy landscape of the system permanently. This disruption can influence the formation and stability of spin domains and alter critical phenomena such as phase transitions [5].

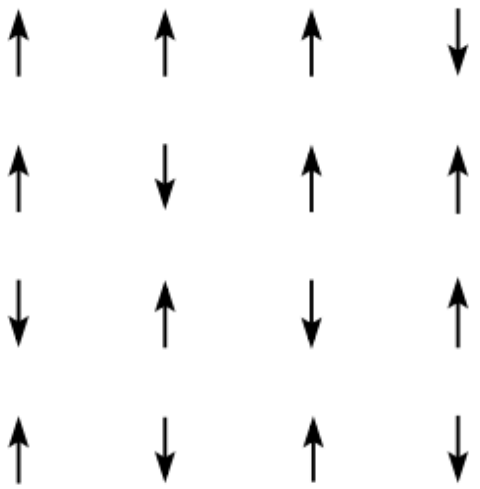


Figure 1: Two Dimensional lattice illustration of an Ising Model. The up and down arrows represent a postive and a negative spin respectively

One key theoretical consequence of site dilution is the shift in the critical temperature. As the fraction of diluted sites increases, long-range order becomes harder to sustain, and the system's ability to undergo spontaneous magnetization diminishes. Analytical and numerical studies have shown that beyond a certain dilution threshold, known as the percolation threshold, the system may no longer support a global magnetic phase [4, 6].
Moreover, the presence of impurities affects thermodynamic quantities such as the specific heat and magnetic susceptibility. Since fewer interacting neighbors exist, fluctuations in energy and magnetization are suppressed, often resulting in broader or weaker singularities at the transition point [5, 7].
Therefore, diluted site impurities in the Ising model serve as a theoretical framework for understanding how structural disorder modifies phase transitions and critical behavior in magnetic systems [4].

2.3 The Monte Carlo Method

The Monte Carlo method estimates the expectation value of an observable $\langle Q \rangle$, which is defined as the sum over all possible micro states σ weighted by the Boltzmann factor $e^{-\beta H(\sigma)}$ [5]:

$$\langle Q \rangle = \sum_{\sigma} Q(\sigma) \frac{e^{-\beta \mathcal{H}(\sigma)}}{\mathcal{Z}} \tag{7}$$

where $\beta = \frac{1}{k_B T}$ as usual. The expectation value is normalized by the canonical (NVT) ensemble partition function $\mathcal{Z}$, which can be rewritten as a sum over the system energy E, with $\Omega(E)$ denoting the density of states[1].

$$\mathcal{Z} = \sum_{\sigma} e^{-\beta \mathcal{H}(\sigma)} = \sum_{E} \Omega(E) e^{-\beta E} \tag{8}$$

[1] number of available microstates at that energy

Finally, we combine the Boltzmann factor with the partition function to define the equilibrium probability density function.

$$P^{\text{eq}}(\sigma) = \frac{e^{-\beta\mathcal{H}(\sigma)}}{\mathcal{Z}} \tag{9}$$

It allows for estimating the expectation of thermodynamic observables, whose probability distribution cannot be efficiently sampled via random sampling due to the enormous number of possible micro states. Even for a relatively simple 2D Ising model, the number of possible micro states is 2^{L^2}, corresponding to 65536 microstates for a small lattice of size 4×4!

Sampling inefficiency can be solved by *importance sampling*, which a variance reduction technique used for estimating expectation values [8]. The core of this method involves treating the whole system as a Markov chain, which samples future system configurations purely based on the current state of the system and the transition probabilities from one configuration to another. Its defining feature is the *Markov property*;

$$P\{X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \ldots, X_1 = i_1, X_0 = i_0\} = P\{X_{n+1} = j \mid X_n = i\} = W_{ij}, \tag{10}$$

which can be intuitively described as the 'future' (X_{n+1}) purely depending on the 'present' (X_n), with the 'past' ($X_{n-1}, ..X_0$) having no effect on the probability of transitioning to a future state σ_j from the current state σ_i.Further probabilistic framework defining Markov chains (*irreducubility*, *aperiodicity*) is beyond the scope of this paper, we refer the interested reader to a textbook by Sheldon Ross [9].

Transition probabilities W_{ij} must satisfy the following conditions:

$$\text{i) } W_{ij} \geq 0 \quad \forall\, i, j \tag{11}$$

$$\text{ii) } \sum_j W_{ij} = 1 \quad \forall\, i \tag{12}$$

$$\text{iii) } \sum_i W_{ij} P_i^{\text{eq}} = P_j^{\text{eq}} \quad \forall\, j \tag{13}$$

which enforce that W_{ij} is a normalized probability distribution (i)-ii)). Equation 13 is the so called *balance condition*, which requires that the equilibrium distribution P^{eq} is a stationary[2] distribution of the transition probability matrix W_{ij}[5]. An even stricter condition is the so called *detailed balance*;

$$W_{ij} P_j^{\text{eq}} = W_{ji} P_i^{\text{eq}} \tag{14}$$

which solves the *master equation* of the system [10] and is the basis of a local update Monte Carlo algorithm; Metropolis.

2.4 Metropolis Monte Carlo

The Metropolis Monte Carlo method consists of generating a Markov chain of configurations following the Boltzmann distribution [10]. Each *Metropolis sweep* flips L^2 randomly chosen

[2] time index independent

spins, based on *trial probability*. The trial probability is equal to

$$f_{ij} = \frac{1}{L^2} \tag{15}$$

for states σ_i and σ_j, which differ by one spin and is 0 otherwise.

The probability of flipping an individual spin is related to the resulting energy change ΔE. Flips resulting in negative ΔE are always accepted, whereas flips with positive ΔE are some times accepted, sometimes not based on the value of *acceptance probability*;

$$q_{ij} = e^{-\beta \Delta E(\sigma_i \to \sigma_j)}. \tag{16}$$

Analogously, the proposed spin flip is rejected with a probability $1 - q_{ij}$. The above described procedure fulfills the *detailed balance condition* [10] and thus is a valid method for propagating the Ising model in time. It is important to note that one Metropolis Monte Carlo time step is equivalent to L^2 trial moves. Periodic boundary conditions are imposed to avoid discrepancies at lattice boundary.

Algorithm 1 Metropolis Monte Carlo Algorithm for 2D Ising model

Require: Initial lattice state x_0, trial probability f, number of Metropolis sweeps N

$x \leftarrow x_0$

for $n = 1$ to N **do**

Select spin site to flip with trial probability f

Lattice configuration with flipped spin $\leftarrow x'$

Compute change in energy ΔE:

$$\Delta E = E(x) - E(x')$$

Compute acceptance probability:

$$q = \min\left(1, e^{-\beta \Delta E}\right)$$

Generate $u \sim \mathcal{U}(0, 1)$

if $u < q$ **then**

Accept spin flip: $x \leftarrow x'$

else

Reject spin flip: keep x

end if

end for

2.5 Wolff algorithm

The Ising model suffers from *critical slowing down*, a phenomena when the correlation time diverges near a critical point [10]. Inspired by the Swendsen and Wang (SW) algorithm [11], Uli Wolff developed an scheme which flips clusters of spins rather than individual spins [12].

Both the SW and Wolff algorithms reduce the critical slowing phenomena comparative to the Metropolis algorithm, with the Wolff algorithm being even more efficient [3] and straightforward to implement. The Wolff algorithm consists of picking a lattice site at random, growing a spin cluster from said lattice site and flipping all cluster spins simultaneously. If this cluster turns out to be large, correlations are destroyed as effectively as by means of the large clusters in the SW algorithm, without the effort of creating the smaller clusters that fill up the remainder of the system. If the Wolff cluster turns out to be small, then not much is gained, but also not much computational effort is required [14].

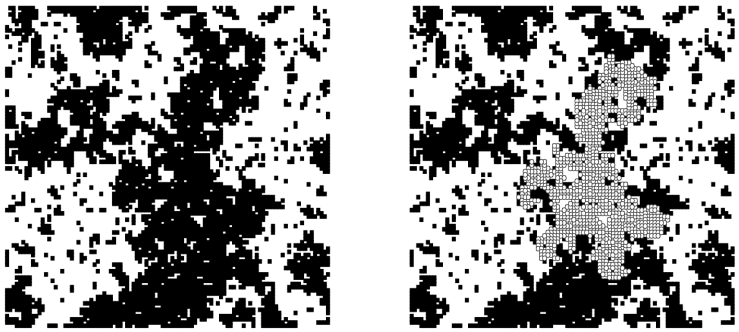


Figure 2: Illustration of a single Wolff cluster for a 100×100 Ising lattice. *Left)*: Initial lattice configuration. *Right)* Highlighted Wolff cluster (consisting of only black spins). Figure adapted from[5].

The Wolff algorithm works as follows. Similarly to Metropolis a random lattice site is chosen with probability dependent on lattice size according to equation 15. Subsequently, we consider its nearest neighbors, particularly those with parallel spin. These bonds are frozen with a probability;

$$P_W = 1 - e^{-2\beta J}. \tag{17}$$

That is to say, a nearest neighbor of identical spin is added to the cluster with probability defined by equation 17. 'Successful' nearest neighbors have their spin flipped and continue to grow the cluster via their neighborhoods. The cluster is considered fully grown, when no additional spins are added to it. This point defines the end of a single Wolff step and its result is an updated Ising lattice; see figure 2.

[3] dynamic critical exponent $z = 0.27$ [13]

Algorithm 2 Single cluster Wolff Algorithm for 2D Ising model

Require: Initial lattice state x_0, trial probability f, freezing probability p, number of Wolff steps N, cluster queue q

```
for n = 1 to N do
    Select cluster spin from x_0 with trial probability f
    Add cluster spin to the queue q
    Flip cluster spin
    while q isn't empty do
        Retrieve spin from the queue
        Consider its nearest neighbors (not yet added to the cluster) ← NN_list
        for neighbor in NN_list do
            if neighbor's spin is parallel to cluster spin then
                Generate u ~ U(0, 1)
                if u < p then
                    Add neighbor to the queue q
                    Flip neighbor's spin
                end if
            end if
        end for
    end while
end for
```

3 Methods

In this section, the methods used during the implementation of the simulation are discussed.. This includes the non-dimensionalization of the system to provide intuitive scaling to the observables,the equilibration process of the system, error analysis of the observables, and finally the simulation parameters used in this report.

3.1 Nondimensionalization

Nondimensionalization is a technique used to simplify numerical problems by removing its physical dimensions [15]. This is particularly useful in the context of simulations as it allows for the scaling of observables such that they are viewed in relation to the system and are thus standardized.

It can be seen in the Hamiltonian (equation 1) that temperature, T, plays a significant role in the simulation. Using dimensional analysis, it is found that T can be nondimensionalized as

$$T^* = \frac{k_B T}{J}, \tag{18}$$

where k_B is the Boltzmann constant and J is the nearest neighbor coupling constant.

To achieve a nondimensional system, the calculated lattice energy per spin, E, should also be nondimensionalized. This is done by using

$$E^* = \frac{E}{J}. \tag{19}$$

Now consider other physical values found in the simulation, magnetization, M, susceptibility, χ, and specific heat, C. M is already dimensionless hence $M^* = M$. The dimensions in χ and C arise from their dependencies on T, E, and M. Thus, when the nondimensional values T^*, E^*, and M^* are used instead, χ and C will also be nondimensional.

3.2 Equilibration

The simulation is started from an initialized lattice of random spins. Then, a series of Monte Carlo steps are performed, where a random spin is flipped. The spin is then accepted or rejected based on a Boltzmann probability as described in equation 9. The system thus evolves with each step and eventually reaches a state of uniform energy, also known as the thermalized state [6]. This state is mathematically seen as the state when the values of the Hamiltonian stabilize.

In the algorithm, the thermalization process is implemented by using a rolling average window of 100 time steps where the average energy is calculated for each window and compared to the next window. If the difference between the values is less than the set tolerance, the system is said to be thermalized. This process is implemented for each temperature where the simulation is performed thus providing a dynamic thermalization process.

Once thermalization has been achieved, the steps used in the thermalization process are discarded and further Monte Carlo steps are performed to obtain the final observables.

3.3 Error analysis

After implementation of the Metropolis and Wolff algorithms, the errors associated with each observable were calculated. The data between each consecutive simulation step is correlated by design for both Metropolis and Wolff. This is further seen when an autocorrelation function was used on the computed data of an observable and plotted. The plot obtained was then used to fit an exponential of the form

$$f(t) = ae^{bt}, \tag{20}$$

where a, and b are constants and t is the variable of the function. An example of the plot can be found in the appendix, A.1. The autocorrelation time, τ, was then determined as $\tau = -\frac{1}{b}$.

In order to circumvents the need to derive an analytical formula for error calculation, the block bootstrapping method was used. Block bootstrapping accounts for the correlation of the data by sampling blocks of consecutive data points instead of individual points [16]. Thus, each block mimics the time-dependencies present in the data; preserving the correlation between data points [17].

The size of the blocks used on the data set is of significant importance when using the block bootstrapping method. The block size should be at least multiple factors of τ in order to preserve the correlation of the data within the blocks. However, the total number of blocks should also be large, approximately 30 blocks, to ensure reliability of the calculated error [18]. In the simulation, a static block size of 100 steps was used.

Then, for each block the mean and standard deviation are calculated where the standard deviation, σ, is then used to calculate the error with

$$error = \frac{\sigma}{\sqrt{n_{bins}}}, \tag{21}$$

where n_{bins} is the number of blocks used on the data.

3.4 Simulation Parameters

The simulation of the Ising model is defined to be dependent on a set of simulation parameters. Throughout this report, a few main parameters were consistently set as the following:

- Number of Monte Carlo steps (after discarding thermalization), $n\_steps = 20000$ or 50000 for Metropolis and Wolff algorithms respectively
- Window size (for equilibration), $window\_size = 100$
- Tolerance (for equilibration), $tol = 0.001$
- Block size (for block bootstrapping), $bin\_size = 100$
- Nearest neighbor coupling constant, $J = 1$
- External magnetic field strength, $H = 0$
- Max thermalization (for dynamic thermalization), $max\_therm = 20000$

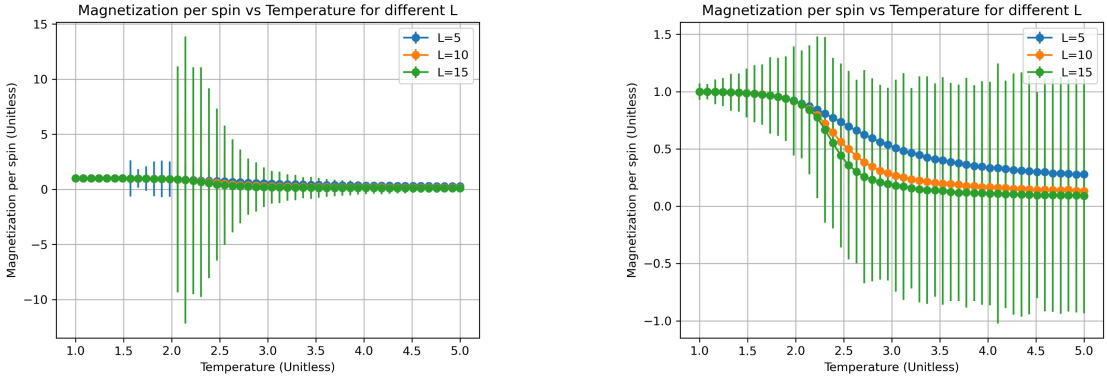
The Boltzmann constant, k_B, was also set to 1 throughout the experiment to simplify calculations. Additionally, other parameters were varied in order to investigate their effects in different use cases. These parameters include temperature, T, and lattice size, L.

4 Results and Discussion

This section illustrates simulated Ising model observables. Temperature dependence of magnetization, energy, susceptibility and specific heat are discussed by directly comparing Metropolis Monte Carlo and the Wolff algorithm. Furthermore, additional magnetization plots were validated against literature for smaller lattice sizes; see Appendix: Validation A.2 for further sanity check details. Lastly, the main research question is investigated by simulating magnetization and internal energy as a function of temperature at different lattice impurity concentrations.

4.1 2D Ising model validation: Metropolis vs Wolff

This section serves as a validation of the 2D Ising model. Both Metropolis and Wolff algorithm were applied to simulate magnetization (per spin), energy (per spin), susceptibility and specific heat (per spin).

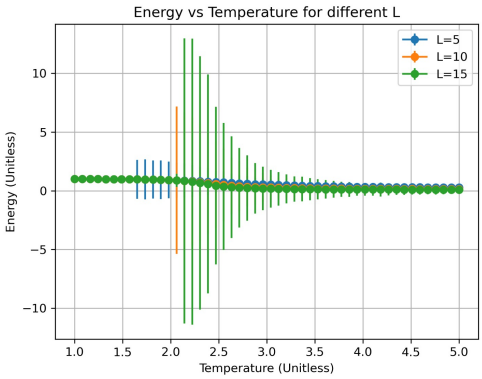


(a) Metropolis Monte Carlo ($n\_steps = 50,000$)

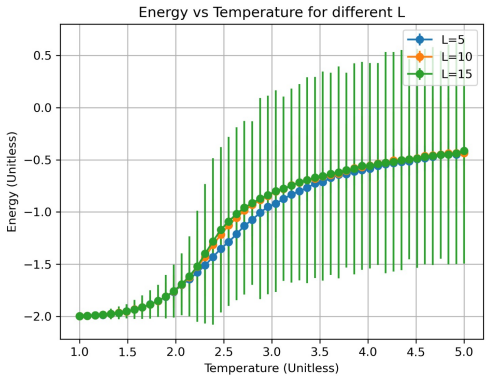
(b) Wolff algorithm ($n\_steps = 20,000$)

Figure 3: Magnetization per spin as a function of dimensionless temperature T for lattice size $L = 5, 10, 15$. Error bars were calculate via block bootstrapping. Simulation parameters: $window\_size = 100, bin\_size = 100, tol = 0.001, J = 1$, and $H = 0$

Figure 3 shows how absolute average magnetization per spin decreases from 1 to 0 as temperature increases. Larger lattice sizes are characterized with a steeper slope near the critical temperature ($2.269\frac{K}{J}$), which agrees with both exact analytical solution by Onsager [19] and random path sampling by Amaral et al [7]. Metropolis results in notably larger error bars, which effectively conceal expected characteristics magnetization curve. This is due to larger correlation time near the phase transition ; critical slowing down. Metropolis resulted in errors order of magnitude larger than Wolff; 15 and 1.5 respectively.
Each data point was averaged over thousands of simulation timestep after system thermalization. Thermalization was conducted dynamically, to account for possible temperature dependence. While current results agree with previous analytical and simulation work both qualitatively and quantitatively, averaging over more simulation time steps would result in lower errors.



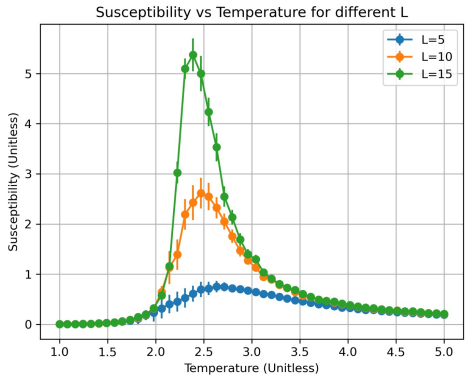
(a) Metropolis Monte Carlo ($n\_steps = 50,000$)



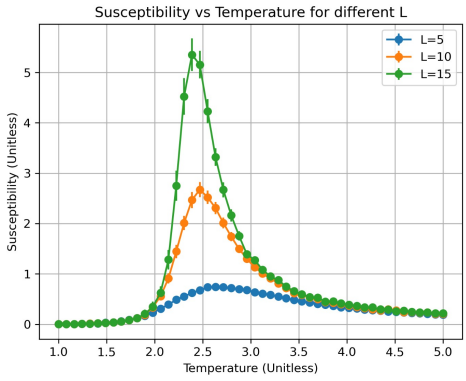
(b) Wolff algorithm ($n\_steps = 20,000$)

Figure 4: Energy per spin as a function of dimensionless temperature T for lattice size $L = 5, 10, 15$. Error bars were calculate via block bootstrapping. Simulation parameters: $window\_size = 100, bin\_size = 100, tol = 0.001, J = 1$, and $H = 0$

Figure 4 shows that in absence of an external magnetic field ($H = 0$), time average of internal energy increases from -2 to -0.5. While at low temperatures the spins are mostly aligned, increase in temperature results in random spin configurations, which cancel out leading to 0 internal energy per spin. Simulated non-dimensional energy range agrees with results by Amaral et al [7], however the lower limit disagrees with Kramer's analytical solution: $-\frac{NJ}{2}$[20]. This discrepancy arises from a factor 1/2 difference in the definition of Kramer's interaction Hamiltonian, proving full agreement between theory and simulation. Yet again, Metropolis results are noisier near the critical temperature even after time averaging over 30, 000 more simulation steps compared to the Wolff algorithm. Large Metropolis error estimates near the critical temperature (especially for $L = 15$) may appear abnormal, however the internal energy does increase in identical manner to Wolff results.



(a) Metropolis Monte Carlo ($n\_steps = 50,000$)

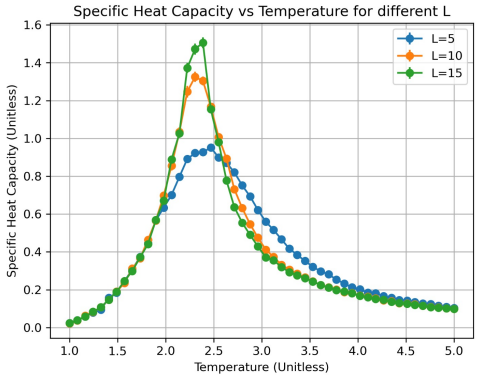


(b) Wolff algorithm ($n\_steps = 20,000$)

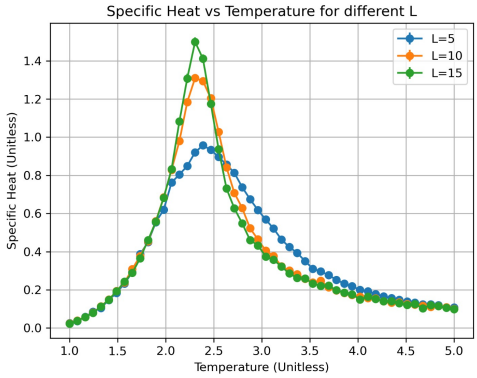
Figure 5: Susceptibility per spin as a function of dimensionless temperature T for lattice size $L = 5, 10, 15$. Error bars were calculate via block bootstrapping. Simulation parameters: $window\_size = 100, bin\_size = 100, tol = 0.001, J = 1$, and $H = 0$

Figure 5 shows how increasing lattice size L narrows the susceptibility curve, agreeing with

previous work [21]. In the infinite limit ($L \to \infty$) susceptibility diverges according to the scaling law $\chi \sim \left|1 - \frac{T}{T_c}\right|^{-\gamma}$ [5], with γ denoting the critical exponent. Note that the peaks occur at the critical temperature for all lattice sizes. Similarly to previous observables, Metropolis Monte Carlo results in larger error bars compared to the Wolff algorithm.



(a) Metropolis Monte Carlo ($n\_steps = 50,000$)



(b) Wolff algorithm ($n\_steps = 20,000$)

Figure 6: Specific heat per spin as a function of dimensionless temperature T for lattice size $L = 5, 10, 15$. Error bars were calculate via block bootstrapping. Simulation parameters: $window\_size = 100, bin\_size = 100, tol = 0.001, J = 1$, and $H = 0$

The last validation figure 6, shows specific heat per spin as a function of non-dimensional temperature and lattice size. Visible peaks occur at critical temperature as expected due to larger energy fluctuations leading to larger internal energy variance (see equation 5). Larger lattice size results in sharper peaks, with all peaks defined by the critical temperature. Interestingly, specific heat was shown to vary the least between Metropolis and Wolff. Having said that, Metropolis was averaged over 30 000 more time steps, hence the figures are not fully identical.

This section showed that both Metropolis and Wolff implementation give observables comparable to both analytical and simulation studies, with large Metropolis errors often concealing the underlying phase transition curves. The next section builds on the simple 2D Ising model by simulating lattice impurities.

4.2 Impurities results

The simulations on the Wolff algorithm were then repeated with impurities in the lattice as described in section 2.2. The observables obtained were plotted similarly to those without impurities. In this section, the energy and magnetization of the lattice with impurities is investigated. First, the energy of the lattice per spin was tested with different impurity concentrations. The plot obtained is shown below.

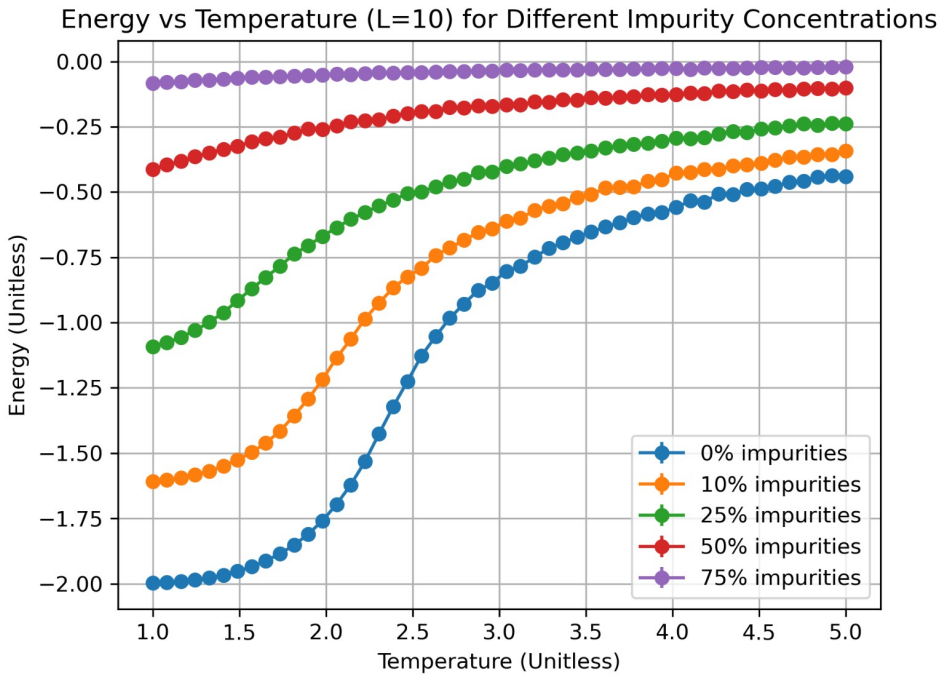


Figure 7: Plot of energy per spin against temperature using the Wolff algorithm for five different impurity concentrations, 0%, 10%, 25%, 50%, and 75%. Error bars were calculated via block bootstrapping. Simulation parameters: $L = 10, n\_steps = 20000, window\_size = 100, bin\_size = 100, tol = 0.001, J = 1$, and $H = 0$.

From the graph, it is observed that the general trend of the data remains unchanged even with the addition of the impurities. The curves plotted exhibit a shift as the system transitions from a disordered system to an ordered system. This is to be expected, as the addition of impurities does not change the fundamental workings of the system. However, the range of the data decreases with increasing impurity concentration and the transition point is less visible. The overall affect of temperature is much smaller and the energy is higher at higher impurity concentrations; seen more obviously at lower temperatures .

The energy of the system is dependent on the interactions between neighboring spins. In the presence of impurities, these interaction bonds are 'broken' and the spins are uncoupled. At low temperatures, the system should be in an ordered state which minimizes energy. However, with increasing impurity concentration, the system is less ordered as all spins in the impurity are set to 0. Hence causing the overall energy at such low temperatures to be higher. This explains the correlation between high energy and high impurity concentration at low temperatures.

Additionally, due to these 'broken' bonds, there are less spin-spin interactions and long-range order is difficult to achieve. Thus, there is also less change in energy as the system transitions into the ordered state, explaining the decrease in change of energy with temperature.

Next, the magnetization of the system was investigated at different temperatures and impurity concentrations as shown in the figure below.

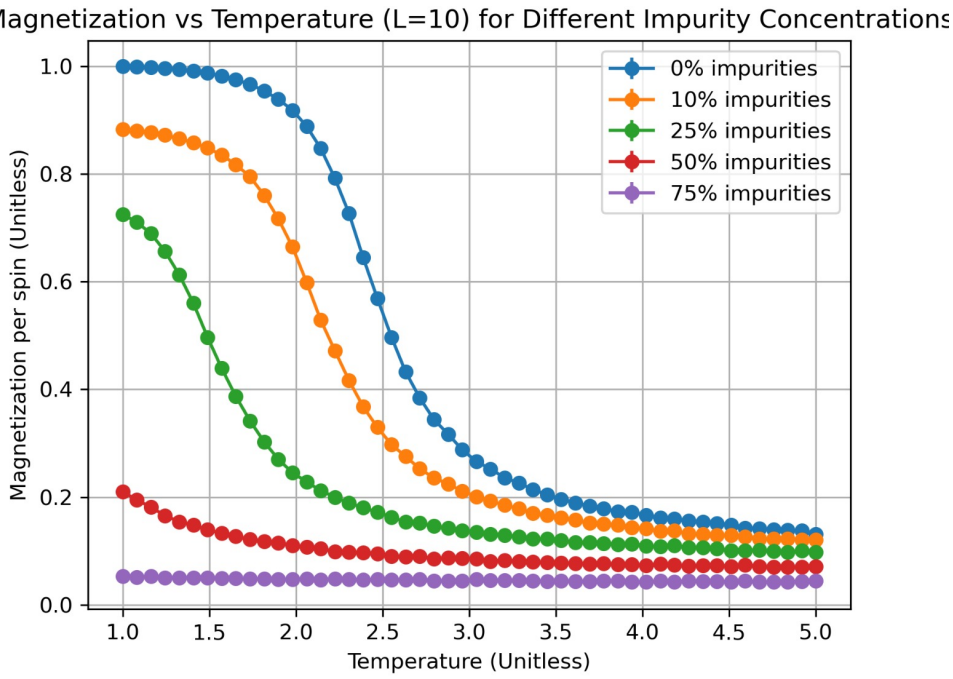


Figure 8: Plot of magnetization per spin against temperature using the Wolff algorithm for five different impurity concentrations, 0%, 10%, 25%, 50%, and 75%. Error bars were calculated via block bootstrapping. Simulation parameters: $L = 10, n\_steps = 20000, window\_size = 100, bin\_size = 100, tol = 0.001, J = 1$, and $H = 0$.

The plot shows that the critical temperature, T_C, is shifted lower with increasing impurity concentrations. The impurities in the lattice prevent the system from forming a large magnetized cluster over the whole lattice as long-range order becomes harder to sustain. Instead, it is expected that smaller isolated clusters are formed. The disorder in the system thus causes the transition point (critical temperature) to shift lower and for the transition to be slower.

5 Conclusion

The 2D Ising model is a widely adapted analytically solved model describing how individual spins give rise to continuous magmatic phase transitions. This work implemented both Metropolis Monte Carlo and single cluster Wolff algorithm to simulate absolute average magnetization per spin, internal energy per spin, susceptibility and specific heat per spin. Both algorithms were validated against literature, with Wolff showing drastically lower error due to shorter correlation time near the critical temperature of the system ($2.269\frac{K}{J}$). The main research question investigated the effect of lattice impurities (site dilution) on internal energy per spin and magnetization per spin as a function of temperature. High impurity concentration reduces interaction between spins and thus also reduces energy. There is a decrease in the value of critical temperature as a result of less spin interactions and isolated cluster formations. Future perspectives include expanding the Ising model into 3D [22] or analyzing local impurity effects [4].

6 Bibliography

[1] T. Nan and N. Sun, "12 - progress toward magnetoelectric spintronics," in *Composite Magnetoelectrics* (G. Srinivasan, S. Priya, and N. X. Sun, eds.), Woodhead Publishing Series in Electronic and Optical Materials, pp. 329–356, Woodhead Publishing, 2015.

[2] V. Marghussian, "4 - magnetic properties of nano-glass ceramics," in *Nano-Glass Ceramics* (V. Marghussian, ed.), pp. 181–223, Oxford: William Andrew Publishing, 2015.

[3] I. B. Djordjevic, "Chapter 12 - quantum machine learning," in *Quantum Communication, Quantum Networks, and Quantum Sensing* (I. B. Djordjevic, ed.), pp. 491–561, Academic Press, 2023.

[4] R. Schäfer, H. Beck, and H. Thomas, "The two-dimensional ising model with isolated impurity," *Zeitschrift für Physik B Condensed Matter*, vol. 41, pp. 259–268, 1981.

[5] W. Janke, *Monte Carlo Methods in Classical Statistical Physics*, pp. 79–140. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[6] R. Neupane, H. R. K. Gauli, K. B. Rai, and K. Giri, "Monte-carlo simulation of phase transition in 2d and 3d ising model," *Scientific World*, vol. 16, p. 12–20, Aug. 2023.

[7] J. S. Amaral, J. N. Gonçalves, and V. S. Amaral, "Thermodynamics of the 2-d ising model from a random path sampling method," *IEEE Transactions on Magnetics*, vol. 50, no. 11, pp. 1–4, 2014.

[8] S. Theodoridis, "Chapter 14 - monte carlo methods," in *Machine Learning (Second Edition)* (S. Theodoridis, ed.), pp. 731–769, Academic Press, second edition ed., 2020.

[9] S. Ross, "-markov chains," in *Introduction to Probability Models (Eleventh Edition)* (S. Ross, ed.), pp. 183–276, Boston: Academic Press, eleventh edition ed., 2014.

[10] J. Thijssen, *The Monte Carlo Method*, ch. 10. Cambridge: Cambridge University Press, 2 ed., 2007.

[11] R. H. Swendsen and J. S. Wang, "Nonuniversal critical dynamics in monte carlo simulations," *Physical Review Letters*, vol. 58, no. 2, pp. 86–88, 1987.

[12] U. Wolff, "Collective monte carlo updating for spin systems," *Phys. Rev. Lett.*, vol. 62, pp. 361–364, Jan 1989.

[13] J. Thijssen, *Computational methods for lattice field theories*, ch. 15. Cambridge: Cambridge University Press, 2 ed., 2007.

[14] E. Luijten, "Introduction to cluster monte carlo algorithms," in *Computer Simulations in Condensed Matter Systems: From Materials to Chemical Biology Volume 1* (M. Ferrario, G. Ciccotti, and K. Binder, eds.), vol. 703 of *Lecture Notes in Physics*, pp. 13–38, Berlin, Heidelberg: Springer, 2006.

[15] Sept. 2024. Page Version ID: 1244787812.

[16] Apr. 2025. Page Version ID: 1285771474.

[17] JT, "Block bootstrapping with time series and spatial data," Feb. 2023.

[18] *Quantum simulations of complex many-body systems: from theory to algorithms: winter school, 25 February - 1 March 2002, Rolduc Conference Centre, Kerkrade, the Netherlands; lecture notes*. NIC series, Jülich: NIC-Secretariat, 2002.

[19] L. Onsager, "Crystal statistics. i. a two-dimensional model with an order-disorder transition," *Phys. Rev.*, vol. 65, pp. 117–149, Feb 1944.

[20] H. A. Kramers and G. H. Wannier, "Statistics of the two-dimensional ferromagnet. part i," *Phys. Rev.*, vol. 60, pp. 252–262, Aug 1941.

[21] P. Secular, "Monte-carlo simulation of small 2d ising lattice with metropolis dynamics." http://secular.me.uk/physics/ising-model.pdf, 2015. Dated: 6th February 2015.

[22] A. M. Ferrenberg, J. Xu, and D. P. Landau, "Pushing the limits of monte carlo simulations for the three-dimensional ising model," *Physical Review E*, vol. 97, Apr. 2018.

[23] J. Amaral, J. Goncalves, and V. Amaral, "Thermodynamics of the 2-d ising model from a random path sampling method," *IEEE Transactions on Magnetics*, vol. 50, p. 1002204, 11 2014.

A Appendix

A.1 Autocorrelation

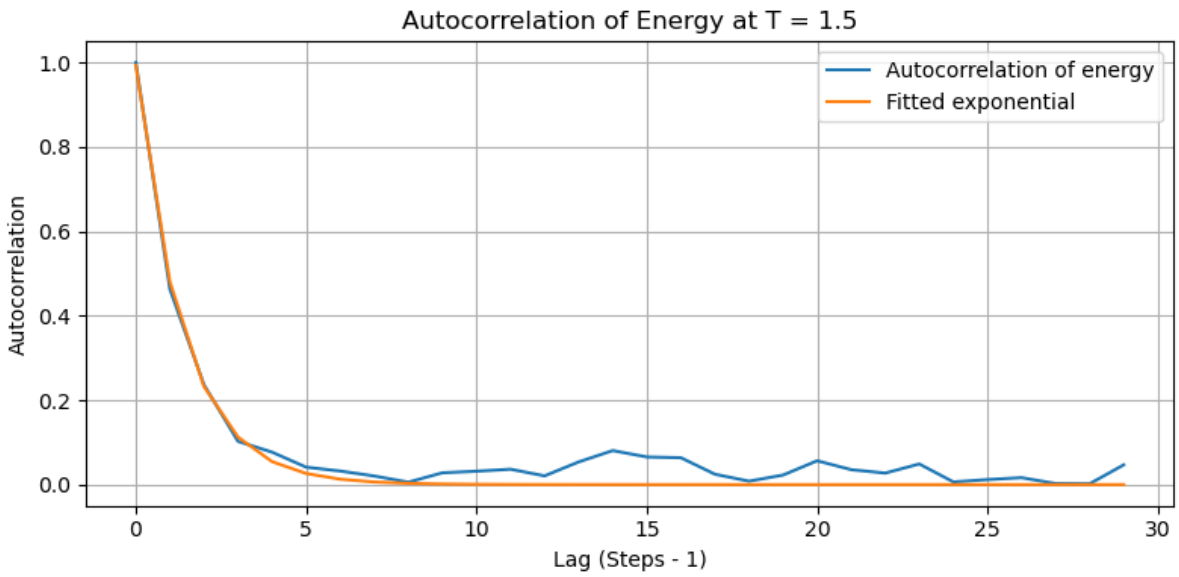
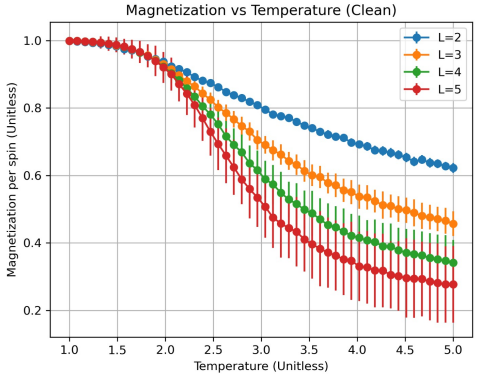


Figure 9: Plot of the autocorrelation function on the energy of the system (in blue) and a fitted exponential (in orange) vs lag. Energies of the lattice were obtained using the Metropolis algorithm. Simulation parameters: $L = 15, T = 1.5, n\_steps = 1000, static\_thermalization = 200, J = 1$, and $H = 0$.

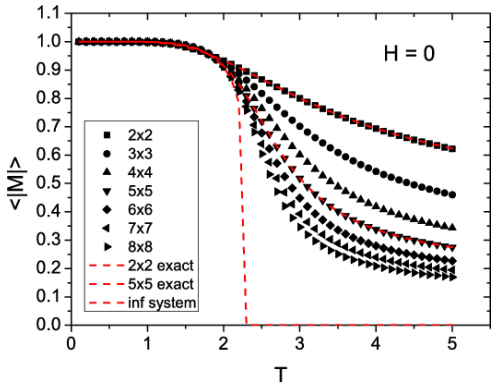
Figure 9 is an example of the plotted autocorrelation function and its corresponding fitted exponential. The plot was used to ensure that the exponential function was a good fit which provides feedback on the accuracy of the obtained autocorrelation time.

A.2 Validation

In order to answer the main research question, validation of both Metropolis and Wolff algorithm was required. Validation was conducted by simulating at settings (temperature, lattice size, boundary conditions) identical to literature. Main sanity checks consisted of identifying the critical temperature (the shift/peak in an observable graph), noting the observable axis limits and qualitative behavior for differing lattice sizes. We report magnetization graph as illustration, but all observables were compared to literature prior to results generation



(a) Wolff simulated absolute average magnetization per spin



(b) Absolute average magnetization per spin from Amaral et al. [23]

Figure 10: Comparison of Wolff simulated absolute average magnetization per spin with literature. Magnetization is plotted as a function of dimensionless temperature T. Both plots include no external magnetic field ($H = 0$) and identical coupling constant ($J = 1$). Curves correspond to lattice size $L = 2, 3, 4, 5$.

A.3 Code Profiling

The profiling displayed below was performed utilizing the *%lprun* command in a Jupyter notebook environment. The run_temperature_sweep function was profiled for both Metropolis and Wolff algorithm implementation. The analysis was conducted to pin-point optimization candidates, however the current implementation was already designed with efficiency in mind. For example multiple approaches for determining the nearest neighbors more efficiently were compared prior to finalizing the MC algorithms. Finally, all results were simulated locally on multiple cores simultaneously, with each core calculating specific temperature and lattice size setting. This approach required longer for Metropolis compared to Wolff, proving that Wolff algorithm is a better candidate for simulating phase transitions within the 2D Ising model.

```
329
330                                                 #runs over n_steps+thermalisation MC time steps, so that time averages can be taken over n_steps (excluding thermalisation, see below)
331    240200   42995982.0    179.0      0.1         for step in range(n_steps + thermalisation):
332    240000   46546570.0    193.9      0.2             if alg=='W':
333                                                         wolff_step(lattice, T, J=1,)
334                                                     else:
335    240000        2e+10  86987.0     69.1                 metropolis_step(lattice, T, J, H)[0]
336
337    240000 8009822703.0  33374.3     26.5             E = calculate_energy(lattice, J, H)
338    240000 1110116742.0   4625.5      3.7             M = np.sum(lattice)
339
340    240000   45855073.0    191.1      0.2             energies.append(E)
341    240000   44155531.0    184.0      0.1             magnetizations.append(M)
342
343                                                 #averages per spin, discards thermalisation
344       200      45969.0    229.8      0.0         N = n * n
345       200    9926530.0  49632.7      0.0         E_arr = np.array(energies[thermalisation:])
346       200    9784061.0  48920.3      0.0         M_arr = np.array(magnetizations[thermalisation:])
347
348                                                 #calculates observables: energy, magnetisation, specific heat and susceptibility
349       200    3969207.0  19846.0      0.0         e_mean = np.mean(E_arr) / N
350       200    2620650.0  13103.2      0.0         m_mean = np.mean(np.abs(M_arr)) / N
351       200   10336642.0  51683.2      0.0         c_heat = (np.var(E_arr) / (T**2)) / N
352       200    6175111.0  30875.6      0.0         susceptibility = (np.var(np.abs(M_arr)) / T) / N
353
354                                                 #record observable for current temp
355       200      52180.0    260.9      0.0         E_avg.append(e_mean)
356       200      37445.0    187.2      0.0         M_avg.append(m_mean)
357       200      35093.0    175.5      0.0         C.append(c_heat)
358       200      47874.0    239.4      0.0         chi.append(susceptibility)
```

Figure 11: Computation time distribution Metropolis temperature sweep

We see that majority of the compute time is spent in Metropolis step (70%), as we expected.

The other notable function is calculate_energy() with approximately 30%.

317					#data storage
318	1	582.0	582.0	0.0	E_avg = []
319	1	159.0	159.0	0.0	M_avg = []
320	1	128.0	128.0	0.0	C = []
321	1	119.0	119.0	0.0	chi = []
322					
323	201	87813.0	436.9	0.0	for T in T_range:
324					# different random lattice for each temp
325	200	5414704.0	27073.5	0.0	lattice = initialize_lattice(n)
326					
327	200	1373645.0	6868.2	0.0	energies = []
328	200	1274843.0	6374.2	0.0	magnetizations = []
329					
330					#runs over n_steps+thermalisation MC time steps, so that time averages can be taken over n_steps (excluding thermalisation, see below)
331	240200	54430853.0	226.6	0.1	for step in range(n_steps + thermalisation):
332	240000	49681381.0	207.0	0.1	if alg=='W':
333	240000	8e+10	328463.5	88.2	wolff_step(lattice, T, J=1,)
334					else:
335					metropolis_step(lattice, T, J, H)[0]
336					
337	240000	9132183647.0	38050.8	10.2	E = calculate_energy(lattice, J, H)
338	240000	1114288883.0	4642.9	1.2	M = np.sum(lattice)
339					
340	240000	51646317.0	215.2	0.1	energies.append(E)
341	240000	43656694.0	181.9	0.0	magnetizations.append(M)
342					
343					#averages per spin, discards thermalisation
344	200	50126.0	250.6	0.0	N = n * n
345	200	9948995.0	49745.0	0.0	E_arr = np.array(energies[thermalisation:])
346	200	9719563.0	48597.8	0.0	M_arr = np.array(magnetizations[thermalisation:])
347					
348					#calculates observables: energy, magnetisation, specific heat and susceptibility
349	200	4547112.0	22735.6	0.0	e_mean = np.mean(E_arr) / N
350	200	3177383.0	15886.9	0.0	m_mean = np.mean(np.abs(M_arr)) / N
351	200	11973035.0	59865.2	0.0	c_heat = (np.var(E_arr) / (T**2)) / N
352	200	6285830.0	31429.2	0.0	susceptibility = (np.var(np.abs(M_arr)) / T) / N
353					
354					#record observable for current temp
355	200	53955.0	269.8	0.0	E_avg.append(e_mean)
356	200	37762.0	188.8	0.0	M_avg.append(m_mean)
357	200	34038.0	170.2	0.0	C.append(c_heat)
358	200	56095.0	280.5	0.0	chi.append(susceptibility)
359					
360	1	8405.0	8405.0	0.0	return {
361	1	284.0	284.0	0.0	'T': T_range,
362	1	225.0	225.0	0.0	'E': E_avg,
363	1	204.0	204.0	0.0	'M': M_avg,
364	1	202.0	202.0	0.0	'C': C,
365	1	211.0	211.0	0.0	'Chi': chi
366					}

Figure 12: Computation time distribution Wolff temperature sweep

Interestingly, the Wolff algorithm takes 3x longer than Metropolis under identical simulation settings via *lprun* (90s vs 30s). However, we saw that during result generation Wolff took considerable less time to execute. Approximately 90% of time is spent in the Wolff step and the remaining 10% is in calculate_energy().

```
102
103                                             #lattice size
104         1       2261.0   2261.0      0.5    n = lattice.shape[0]
105         1        184.0    184.0      0.0    accepted = 0
106
107                                             #sampling outside the loop to increase code efficiency-> one big sampling call for all random spin flips
108         1      50624.0  50624.0     10.9    sample=np.random.randint(low=0, high=n, size=2*n*n, dtype=int)
109
110       101      15621.0    154.7      3.4    for k in range(n*n):
111                                                 # Choose a random spin
112       100      22844.0    228.4      4.9        i = sample[2*k]
113       100      21715.0    217.2      4.7        j = sample[2*k+1]
114       100      27034.0    270.3      5.8        s = lattice[i,j]
115
116                                                 #NN
117       300      86652.0    288.8     18.7        neighbors = lattice[(i+1)%n, j] + lattice[(i-1)%n, j] + \
118       200      61952.0    309.8     13.4                    lattice[i, (j+1)%n] + lattice[i, (j-1)%n]
119
120                                                 # ΔE = E_after-E_before, where E_before=E_local(s) and E_after=E_local(-s)
121                                                 #Energy change if we flip this spin
122       100      36394.0    363.9      7.9        delta_E = 2*s * (J * neighbors +H)
123
124                                                 # Accept flip with Metropolis probability
125       100      99202.0    992.0     21.4        if delta_E <= 0 or np.random.rand() < acceptance(delta_E, T):
126        65      26406.0    406.2      5.7            lattice[i, j] *= -1  # Flip the spin
127        65      10319.0    158.8      2.2            accepted += 1
128
129         1        418.0    418.0      0.1    acceptance_ratio = accepted / (n*n)
130
131         1       1392.0   1392.0      0.3    return lattice, acceptance_ratio
```

Figure 13: Metropolis algorithm single step analysis

Overall the main contributions for Metropolis come from generating the random numbers (combined both for random spin selection and flip probability) and getting the nearest neighbors. We previously tried to generate most random numbers at the same time with a larger size to optimize the RNG calls. Further optimization would include applying the same approach for simulating spin probability. We previously tried to optimize the neighbors via the convolution kernel, but that was slower than the current implementation. If we were to try to optimize the Metropolis algorithm now, we would focus on the neighbors or the numpyfied approach.

```
                                            """
  1       7110.0   7110.0      0.5    n = lattice.shape[0]
  1      19230.0  19230.0      1.4    #keeps track of spins already added to the cluster
  1      34727.0  34727.0      2.5    visited=np.zeros((n,n))
                                      #creates queue to keep track of spins to be considered
                                      q = Queue(maxsize = n*n)

                                      #Cluster initialisation

                                      #select random spin to start with
  1     152063.0 152063.0     10.9    cluster_start = list(np.random.randint(0, n, size=2))
                                      #define cluster spin
  1       5504.0   5504.0      0.4    cluster_spin=lattice[cluster_start[0],cluster_start[1]]
                                      #put into the queue so it can be used to grow the cluster
  1      37301.0  37301.0      2.7    q.put(cluster_start)
                                      #mark starting point as visited
  1       3922.0   3922.0      0.3    visited[cluster_start[0],cluster_start[1]]=1
                                      #flip starting point spin with 100% probability
  1       1976.0   1976.0      0.1    lattice[cluster_start[0],cluster_start[1]]=-lattice[cluster_start[0],cluster_start[1]]

 44      78847.0   1792.0      5.7    while q.qsize()>0:
                                          # get current spin (q.get() deletes from the queue, so you can only use it once)
 43     268901.0   6253.5     19.3        i,j=q.get()

                                          #get nearest neighbours
 43      60100.0   1397.7      4.3        neighbors_idx = [[(i+1)%n, j], [(i-1)%n, j], [i, (j+1)%n], [i, (j-1)%n]]
                                          #get not yet visited nearest neighbours
215     148896.0    692.5     10.7        to_add_idx=[ idx for idx in neighbors_idx if visited[idx[0],idx[1]]!=1]

                                          #only look at neighbours not yet in the cluster
146      52384.0    358.8      3.8        for element in to_add_idx:
                                              #add to cluster with p=1-e^-2J if the current spin=cluster spin
103      55318.0    537.1      4.0            if lattice[element[0], element[1]]==cluster_spin:
 45     145212.0   3226.9     10.4                if np.random.rand()<(1-np.exp(-2*J)):
                                                     #only mark as visited after adding it to the cluster, (we must allow all 4 nn to try to add a this spin to the cluster in order not to bias the distribution)
 42      20069.0    477.8      1.4                    visited[element[0],element[1]]=1
                                                     #flip spin=add to cluster
 42      30111.0    716.9      2.2                    lattice[element[0], element[1]]=-lattice[element[0], element[1]]
                                                     #add to the queue so it can be used to grow the cluster further (via its nearest neighbours)
 42     266154.0   6337.0     19.1                    q.put(element)
  1       2670.0   2670.0      0.2    return lattice
```

Figure 14: Wolff algorithm single step analysis

We see that queue manipulation (to keep track of added spins) takes up approximately 45% of compute time, RNG calls(for random start and add to cluster probability) make up 20% of the compute time and the last notable component is a list comprehension to check if nearest neighbors have been added yet or not (10%). A clear optimization target would be the queuing

structure. Having said that, this is one of the recommended approaches for increasing cluster size while keeping track of already visited spin positions. Therefore we will not focus on optimizing the algorithm further.