Monte Carlo Simulation of the 2-D Ising Model

Hantao Xiao
MACSS

Kaiwen Fu
MCAM

Alexander Cram
MCAM

March 31, 2024

Abstract

This research endeavors to gain a comprehensive understanding of the order-disorder transition at the critical temperature within the Ising model, a representative model in statistical mechanics for magnetic materials. Using Monte Carlo simulations in conjunction with the Metropolis algorithm and Wolff algorithm, we simulate the Ising model on a 2-D lattice and observe variations in magnetization per site and specific heat across different temperatures. In doing so, this study aims to shed light on the magnetic properties transformation from ordered to disordered states at the critical temperature, thereby enriching our comprehension of magnetic materials' behavior.

1 Introduction

In this project, we employ the Markov Chain Monte Carlo (MCMC) method to simulate the Ising model of ferromagnetism. Notably, the Metropolis-Hastings algorithm, a widely recognized and powerful technique for generating the Markov Chain, will be discussed in the following section [Binder, 1997]. We also explore the Wolff clustering algorithm in less granular detail. Subsequently, we utilize the Metropolis-Hastings and Wolff algorithms to simulate the 2-D Ising model and extract its physical properties. The other most common clustering algorithm is the Swendsen-Wang algorithm, but Wolff seems to be preferred.

MCMC is an important method, since the Ising model is NP-hard to simulate analytically [Kalinin and Berloff, 2022], so for even moderate sized lattices, it is infeasible. Using stochastic methods provides a dramatic simplification in the complexity, while retaining useful properties. We will provide an overview of the two stochastic methods used here.

The overview of the Metropolis-Hastings algorithm is as follows: Construct a lattice of points, each with a random spin state -1 or 1. Nearby points having the same spin is a lower energy state, so points on the lattice tend towards their neighbors values. However, the temperature of the metal adds in some randomness, and allows points to flip to a higher energy state. Thus, to iterate forward in time, you pick a point at random and determine whether or not flipping it would lower the energy. If it does, flip it. If not, then choose to flip it or not, with a probability affected by the interaction strength and temperature. One can also model an additional external magnetic field acting on the lattice, which affects the transition probability. Such a process is one step in the algorithm, with a simulation proceeding for many steps.

The overview of the Wolff clustering algorithm is as follows [deLyra, 2006] [Luijten, 2006a]: As before, construct a lattice of points, each with random spins, and select a point at random. Then, look at that point's neighbors and if they have the same spin orientation, pass a random selection test, and are not already in the cluster, they are added to the cluster. Each point that was added to the cluster then has all of its neighbors checked as before, and a cluster is thus iteratively constructed. Once there are no more points with neighbors to add, the entire cluster is flipped. This is one step in the simulation, with the model proceeding for many steps.

Like solving an ODE, one useful application is to run a model forward in time for many steps to try to find an equilibrium state, or to determine the probability distribution of the model. It is also interesting that except for the external field application, this model is strictly local. Far away points do not interact with each other directly. However, one gets large scale behavior as a consequence of chains of behavior (like the game of life). One also discovers that at a high enough temperature, there is a phase transition between ordered and disordered behavior. This effectively means that a magnet breaks at high enough temperatures and loses its magnetism.

2 Numerical Methods

In our research, we delve into the 2D Ising model, a statistically significant system in the realm of physics. By employing Markov Chain Monte Carlo methods, we aim to calculate the approximate averages of various physical quantities associated with this model. The numerical programming techniques involved are fairly straightforward. We use the `random`, `numpy`, `matplotlib`, and `imageio` packages. Most of the work involved is in understanding the physics and math underlying the simulation, and transforming that into functioning code. The rest if this section is devoted to an in-depth explanation of the underlying science.[Fitzpatrick, 2006a] [Jindal, 2007] [Singh, 2014] [Mancini, 2021]

2.1 Ising model

We consider a 2-dimensional nearest-neighbour Ising Model with:

- $N \times N$ = total number of sites on the lattice,
- $\sigma_k \in \{+1, -1\}$ = individual spin site on the lattice.
- H $(h_{i,j})$ denotes the external influence and the default is 0
- J denotes the the exchange energy, with default 1.

The Hamiltonian is given by:

$$E(\vec{\sigma}) = -\sum_{\langle ij \rangle} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i \tag{1}$$

where J_{ij} represents the interaction strength and h_i is the external local magnetic field.
In this system, a particular configuration $\vec{\sigma}$ has the probability density:

$$P(\vec{\sigma}) = \frac{e^{-\beta E(\vec{\sigma})}}{Z} \tag{2}$$

where

$$Z = \sum_{\{\vec{\sigma}\}} e^{-\beta E(\vec{\sigma})} \tag{3}$$

is the partition function with $\beta = \frac{1}{k_B T}$, and the sum is over all spin configurations of the system.

2.2 Energy change in Ising Model using Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is a Monte Carlo method employed for generating a Markov Chain with a desired probability distribution.

In the Ising model, the energy of a system is determined by the interactions between neighboring spins and the influence of an external magnetic field. The energy of a configuration is typically given by the Hamiltonian:

$$-J\sum_{\langle ij \rangle} \sigma_i\sigma_j \tag{4}$$

Here, J is the interaction strength between neighboring spins, σ_i and σ_j are the spins at sites i and j, and the sum is over all pairs of neighboring sites.

Energy Change ΔE

When you flip a single spin, the change in energy, ΔE, is the difference between the energies before and after the flip. If the spin at site i changes from σ_i to $-\sigma_i$, the change in energy is:

$$\Delta E = E_{\text{after}} - E_{\text{before}} \tag{5}$$

Before the flip, the energy contribution from the interactions of spin i with its neighbors is:

$$-J\sigma_i(s_{\text{top}} + s_{\text{bottom}} + s_{\text{left}} + s_{\text{right}}) \tag{6}$$

After the flip, it becomes:

$$-J(-\sigma_i)(s_{\text{top}} + s_{\text{bottom}} + s_{\text{left}} + s_{\text{right}}) \tag{7}$$

The change in energy is thus:

$$\Delta E = 2J\sigma_i(s_{\text{top}} + s_{\text{bottom}} + s_{\text{left}} + s_{\text{right}}) \tag{8}$$

Including the External Magnetic Field H

If there's an external magnetic field H, it contributes an additional term to the Hamiltonian:

$$-H\sum_i \sigma_i \tag{9}$$

The change in energy due to the magnetic field when flipping the spin at site i is:

$$\Delta E_{\text{field}} = 2H\sigma_i \tag{10}$$

2.3 Energy change in Ising Model using Wolff Algorithm

At this point the primary bottleneck of our code is that the Markov chain has a large autocrrelation time near the critical point. This phenomena is called critical slowing down.

To help resolve this problem, essentially we move to more sophisticated Monte Carlo techiques. For this particular model, the most famous of these choices is the Wolff algorithm. The Wolff algorithm is a cluster-move which flips an entire cluster instead of a single spin at a time.Luijten [2006b]

Final Equation

Combining the interaction and magnetic field contributions, the total change in energy when flipping a spin is:

$$\Delta E = 2\sigma_i(s_{\text{top}} + s_{\text{bottom}} + s_{\text{left}} + s_{\text{right}}) + 2H\sigma_i \tag{11}$$

If we absorb the interaction strength J into the definition of the energy scale (setting $J = 1$ for simplicity), the equation becomes:

$$\Delta E = 2\sigma_i(s_{\text{top}} + s_{\text{bottom}} + s_{\text{left}} + s_{\text{right}} + H) \tag{12}$$

This equation is a simplified representation assuming a unit interaction strength and appropriately scaled magnetic field, which is common in theoretical treatments of the Ising model.

Then, if $\Delta E \leq 0$, this move is acceptable, otherwise, we accept the move with probability $A = \exp(-\Delta E/T)$(Metropolis-Hasting algorithm) [Clarki, 2022] [Fitzpatrick, 2006b].

3 Project Structure

Our `src` file is composed of a few main functions. We initially have a function `calculate-total-energy` to compute the total energy of the system, which is used in both simulation methods. Then, we begin with the code for the Metropolis-Hastings method. We first have a function `calculate_energy_change` to determine the change in energy of the system if a lattice value were to be flipped. Then, the function `MCMC_step` carries out one Metropolis-Hastings step, and finally the function `simulate_ising_model` uses `MCMC_step` to simulate the whole process by completing many Metropolis-Hastings steps in a row, and outputs some additional data. Then, we have code for the Wolff method. This method begins by defining `wolff_step`, which computes one entire Wolff clustering algorithm step. `simulate_wolff_model` then uses `wolff_step` to compute many steps in the model, outputting all the same data as `simulate_ising_model`. `Src` also contains some code to generate gifs of the process and save images, as well as some code that executes automatically to produce results if the file is run in a terminal.

4 Project Test

In this section, we are going to have several tests to validate our code. First, we will have a code about the Hamiltonian(about E specifically). Second, we will have a test about the energy change in the Ising model(ΔE specifically). If these two tests are passes, then we can say that our code are appropriate.

The provided code snippet below illustrates the testing segment of our project. Specifically, it aims to verify if the output from the function `calculate-total-energy` matches the result obtained from `calculate_energy_change`.

```
l1 = np.array([[1, 1, 1], [1, 1, 1], [1, 1, 1]])
l2 = np.array([[1, 1, 1], [1, -1, 1], [1, -1, 1]])
l3 = np.array([[-1, 1, -1], [-1, 1, -1], [-1, 1, -1]])
l4 = np.array([[1, 1, 1], [1, -1, 1], [1, 1, 1]])
l5 = np.array([[1, 1, 1], [1, -1, 1], [1, 1, 1]])
l6 = np.array([[-1, -1, -1], [-1, 1, -1], [-1, 1, -1]])


class TestEnergy(unittest.TestCase):

    def test_energy(self):
        self.assertEqual(tools.calculate_total_energy(l1,
        external_field=0), -18)
        self.assertEqual(tools.calculate_total_energy(l2,
        external_field=0), -6)
        self.assertEqual(tools.calculate_total_energy(l3,
        external_field=0), -6)

    def test_energy_change(self):
        self.assertEqual(tools.calculate_energy_change(l1, 1, 1,
        external_field=0), tools.calculate_total_energy(l4,
        external_field=0) - tools.calculate_total_energy(l1,
        external_field=0))

        self.assertEqual(tools.calculate_energy_change(l2, 2, 1,
        external_field=0), tools.calculate_total_energy(l5,
        external_field=0) - tools.calculate_total_energy(l2,
        external_field=0))

        self.assertEqual(tools.calculate_energy_change(l3, 0, 1,
        external_field=0), tools.calculate_total_energy(l6,
        external_field=0) - tools.calculate_total_energy(l3,
        external_field=0))

if __name__ == '__main__':
    unittest.main()
```

We also perform the Monte Carlo simulation multiple times (10 times, as mentioned) with the same number of steps in each run. This is to ensure that each simulation is statistically comparable.

After running the simulations, we calculate the variance in the results of each quantity of interest (Energy, Magnetization, Specific Heat, Susceptibility). Variance is a measure of how much the values differ from each other, which in this context, serves as an estimator for the error of the Monte Carlo simulation.

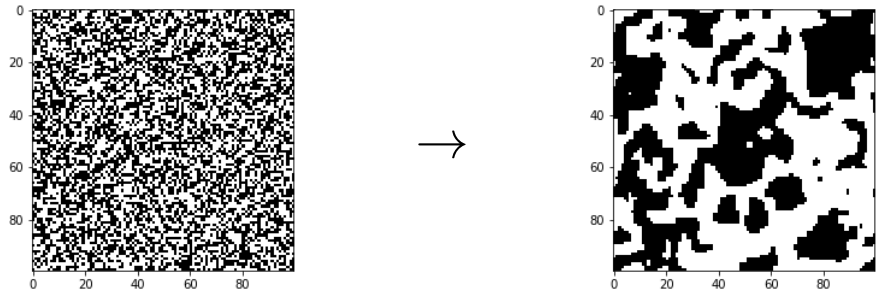
- Energy Variance: 0.0023
- Magnetization Variance: 0.0155
- Specific Heat Variance: 30.683
- Susceptibility Variance: 1.003

Additionally, we conducted an independent assessment of our `MCMC_step` function, yielding variance results for Energy and Magnetization at 3.5673 and 4.2312, respectively.

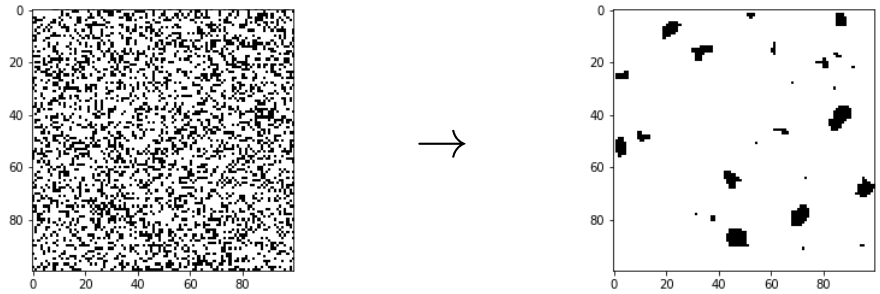
5 Investigations

The following pictures are the beginning and ending states for a variety of initial parameters, that elucidate the relevant behavior we are looking for. [Hammel, 2017] All were run on a 50x50 grid, with 100000 steps for Metropolis-Hastings, and 5000 steps for Wolff.

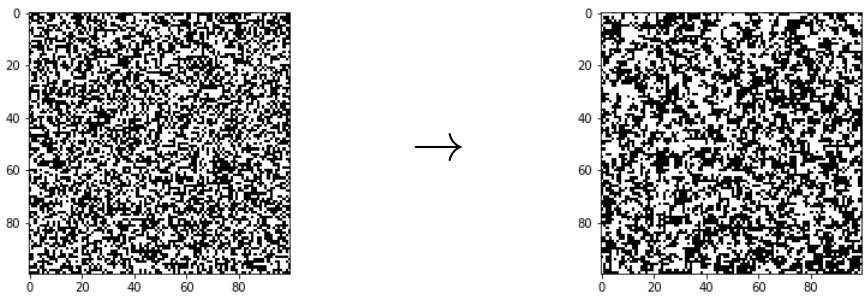
For an initial picture of what happens, we see the Metropolis-Hastings method on a lattice at 0.5 fill and temperature of 0.1. The lattice organizes itself into clusters of spin states, as desired.



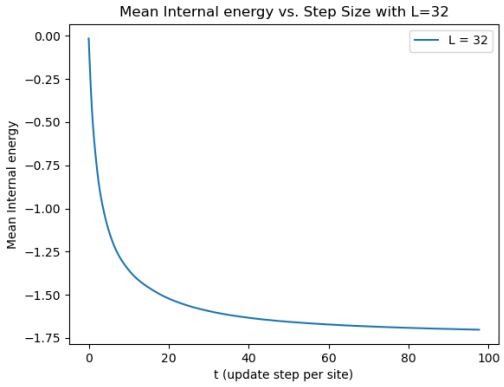
We can also see that if there is a dominant initial condition, the lattice trends towards that state, as we expect. The example here has an initial fill of 0.1.



For a fill of 0.5 and temperature of 5, we see the following behavior that demonstrates the phase transition. Even after 100000 time steps, the lattice is barely more organized than at the beginning. The metal is too hot to magnetize. Higher temperatures will produce similar behavior.



These results are all as expected, which initially indicates that our algorithm is correctly simulating the behavior. We can also graph the mean internal energy for an initial state and show that it decays to a minimum value, as desired.



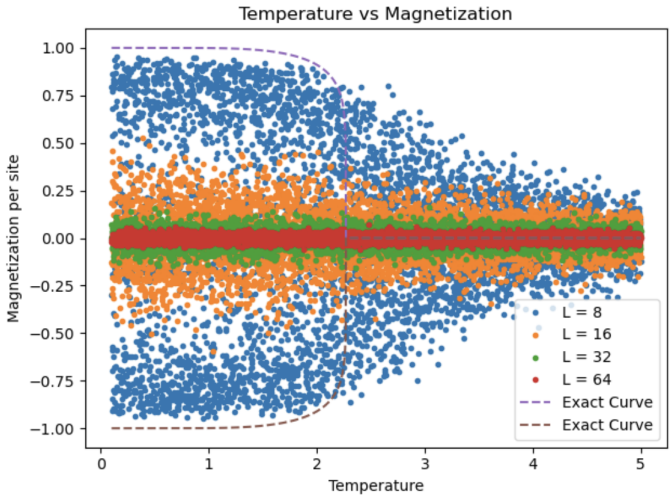
Theoretically, we can find the critical temperature by simple calculation [Wikipedia contributors, 2023b]

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

By this critical temperature, we can find the relations between magnetization and the critical temperature in this case, [Wikipedia contributors, 2023a]

$$M = \begin{cases} 0 & \text{for } T \geq T_c \\ (1 - \sinh^{-4}(2/T))^{1/8} & \text{for } T < T_c \end{cases}$$

We implemented different lattice sizes, $L = 8, 16, 32, 64$ with 1000 steps. Comparing our output with the theoretical output in the figure below, we can see that our models correspond very nicely to the expected result, with a decay towards zero net magnetization at approxmately T = 2.27. Note that lattices with more points are clustered more towards an average of zero, due to the law of large numbers.



Then, we compare the lattice with different temperature, $T = 1.8, 2.3, 4.0$ to show the spin configurations when $L = 32$

Then, we compare the lattice with different temperature, $T = 1.8, 2.3, 4.0$ to show the spin configurations when $L = 64$

From the image above with different lattice size, then we can observe that the spin configurations exhibit distinct patterns at varying temperatures.

1. Low temperatures ($T < Tc$): In this range, well-defined domains characterized by aligned spins are present, separated by domain walls. The system displays robust spin correlations and elevated magnetization, signifying a ferromagnetic nature.

(a) $T = 1.8K$ (b) $T = 2.3K$ (c) $T = 4.0K$

Figure 1: L = 32 with different temperature

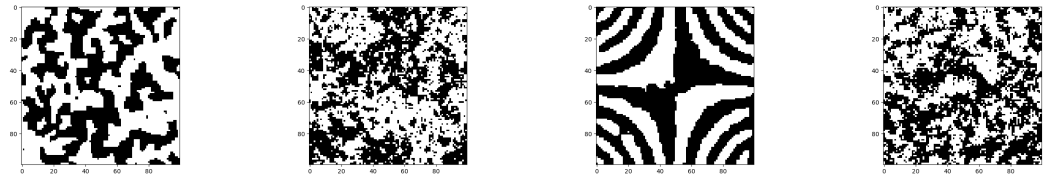
(a) $T = 1.8K$ (b) $T = 2.3K$ (c) $T = 4.0K$

Figure 2: L = 64 with different temperature

2. Near the critical temperature ($T \approx Tc$): As temperature nears the critical point, domain regions become smaller and somewhat more disorganized. The structural features become less conspicuous and more short-ranged, indicating a transition towards a state with reduced magnetization.

3. High temperatures ($T > Tc$): Beyond the critical temperature, the system adopts a completely random and disordered configuration. No observable spin configurations with well-defined domains exist. Spin orientations become random, and the system behaves as a disordered assembly of spins in a paramagnetic state.

Exploring the order-disorder transition in the Ising model will not only improve our comprehension of this pivotal model but also advance our understanding of magnetic materials' behavior, especially at critical junctures.

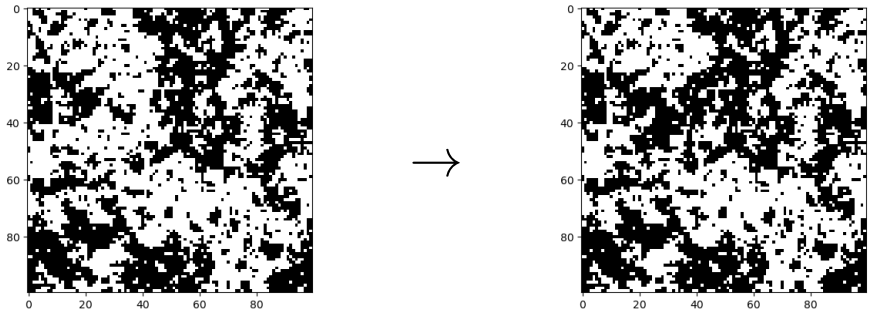


(a) No Field M-H (b) No Field Wolff (c) varying Field M-H (d) Varying Field Wolff

Figure 3: Varying the different Magnetic fire

Both of our models seem to be able to model the quenching process correctly, as seen in the final results below. They can capture a wide range of behavior, including the effects of a varying magnetic field, in this case $sin(x * y)$

Moreover, we can see that the Wolff algorithm does efficiently flip large groups of points, which allows it to more efficiently explore the state space. Note the large cluster near the center that flips from white to black.



6 Discussion

We were also successfully able to find the transition behavior for the Metropolis-Hastings algorithm, which is the desired behavior for the system. However, due to some difficulties with the algorithm, and the length of time the unpolished method takes to run, we could not find the critical temperature for the Wolff method.

The two algorithms have trade-offs in terms of speed. One Wolff algorithm step takes much longer to compute, since constructing the cluster to flip can take very long, as checking all the points to add is expensive. One Metropolis-Hastings step, in comparison, requires just to check if one point swap would raise the energy. However, the Wolff algorithm requires much fewer steps to quench, and is much more efficient about exploring the state space. Metropolis-Hastings has difficulty flipping large islands, while Wolff has no such difficulty, so it is less likely to get stuck in one region of the state space. In the initial example above, with no field, the Metropolis-Hastings algorithm ran for 100000 steps, while the Wolff algorithm ran for a mere 5000. They took approximately equivalent time to run, however, and it seems that Metropolis-Hastings has a more complete quench, but for so few iterations, Wolff is quite powerful.

There is lots of room for improvement in the efficiency of the Wolff algorithm. In particular, checking whether or not a point is already in the cluster is very expensive, since the cluster becomes so large. There must be some optimizations around searching lists, perhaps an ordering on the lattice points of some sort, that would allow the code to execute more quickly.

As in any project, there is always more work to be done. We could implement the Swendsen-Wang algorithm to see how fast it quenches and how fast it explores the state space. A 3-D implementation of any of the models also remains to be done.

7 Conclusion

In this project, we used the Monte Carlo Markov Chain technique to simulate the Ising model of magnetism from statistical physics. Our two algorithms of choice were the Metropolis-Hastings and Wolff clustering algorithms, with more focus on the Metropolis-Hastings algorithm. These algorithms stochastically model our problem, which is otherwise NP-hard to model analytically. We successfully created models that produced the desired behavior for a variety of conditions. Importantly, using the Metropolis-Hastings algorithm, we were able to demonstrate the key behavior of this model: above some critical temperature, magnetization is not possible, since the energy introduced by the temperature overwhelms any alignment behavior we might otherwise get. Thus, the lattice stays disordered on long time scales. While there remain further areas of exploration, this project serves as a good primer for transforming a physical system into a stochastic model and extracting key behavior.

References

Kurt Binder. Applications of monte carlo methods to statistical physics. *Reports on Progress in Physics*, 60(5):487, 1997.

Bryan Clarki. Ising model project, 2022. URL https://courses.physics.illinois.edu/phys498cmp/sp2022/Overview.html#make-up-week. [UIUC Computing in Physics(498 CMP)].

Jorge deLyra. The wolff algorithm, Jul 2006. URL http://latt.if.usp.br/technical-pages/twawesab/Text.html/node1.html.

Richard Fitzpatrick. The ising model, 2006a. URL https://farside.ph.utexas.edu/teaching/329/lectures/node110.html#eising. [Online; accessed 14-November-2023].

Richard Fitzpatrick. Ising model project, 2006b. URL https://farside.ph.utexas.edu/teaching/329/lectures/node110.html. [UT Austin: Computational Physics].

B. D. Hammel. The ising model, 2017. URL https://github.com/bdhammel/ising-model.

Saryu Jindal. Monte carlo simulation of the ising model. *Department of Chemical Engineering and Material Sciences*, page 11, 2007.

Kirill P. Kalinin and Natalia G. Berloff. Computational complexity continuum within ising formulation of np problems. *Communications Physics*, 5(1):20, Jan 2022. ISSN 2399-3650. doi: 10.1038/s42005-021-00792-0. URL https://doi.org/10.1038/s42005-021-00792-0.

Erik Luijten. Introduction to cluster monte carlo algorithms, 2006a. URL https://csml.northwestern.edu/resources/Reprints/lnp_color.pdf.

Erik Luijten. Introduction to cluster monte carlo algorithms. In *Computer Simulations in Condensed Matter Systems: From Materials to Chemical Biology Volume 1*, pages 13–38. Springer, 2006b.

Lorenzo Mancini. Ising model, 2021. URL https://github.com/lorenzomancini1/IsingModel2D_MonteCarlo/blob/master/Ising2D.ipynb. [Online; accessed 14-November-2023].

Rajesh Singh. Ising model, 2014. URL https://rajeshrinet.github.io/blog/2014/ising-model/. [Online; accessed 14-November-2023].

Wikipedia contributors. Square lattice ising model — Wikipedia, the free encyclopedia, 2023a. URL https://en.wikipedia.org/w/index.php?title=Square_lattice_Ising_model&oldid=1152934917. [Online; accessed 9-December-2023].

Wikipedia contributors. Ising model — Wikipedia, the free encyclopedia, 2023b. URL https://en.wikipedia.org/w/index.php?title=Ising_model&oldid=1188214835. [Online; accessed 9-December-2023].