Exam TFY4235: Investigating the Ising Model using Computational Physics

Candidate ID: 10071

ABSTRACT

In this project, the two-dimensional Ising model was studied using Monte Carlo simulations based on the Metropolis algorithm. Key thermodynamic quantities such as energy, magnetization, specific heat, and magnetic susceptibility were calculated as functions of temperature to investigate critical behavior near the phase transition. The critical temperature was estimated by analyzing the peak of the specific heat, with results in good agreement with the known analytical value. Additional simulations explored the effects of frozen impurity sites, which were shown to suppress magnetization and lower the apparent critical temperature due to disrupted spin alignment. Simulated annealing was used to identify low-energy configurations, revealing spin "islands" constrained by impurities. The impact of an external magnetic field was also investigated, showing that it stabilizes magnetization and shifts the critical point to higher temperatures. A scaling analysis of the magnetic susceptibility confirmed the expected data collapse, supporting the theoretical scaling behavior. The study also highlights the importance of sufficient sampling in Monte Carlo methods, as statistical fluctuations and random seed effects can significantly affect the accuracy of critical temperature estimates. Overall, the project demonstrates the effectiveness of computational approaches in exploring physical behavior and provides a foundation for future studies on disordered or more advanced spin models.

I. INTRODUCTION

The Ising model is a well-known system in statistical physics that helps us understand how certain materials behave when they go through phase transitions. It was first introduced by Wilhelm Lenz and later studied in one dimension by his student Ernst Ising in 1925[1]. In this model, each point on a grid (called a lattice) can have a spin pointing either up (+1) or down (-1), and each spin interacts with its nearest-neighbors. Even though the model is simple, it can still show complex behavior, like a sudden change in magnetization at a critical temperature. This change was first shown in the two-dimensional version of the model, solved exactly by Lars Onsager in 1944[2].

Due to it's simplicity, the Ising model is a good example to test and understand numerical methods. It shows important features of magnetic systems, like how spontaneous magnetization can appear when the temperature drops below a certain point. Even though it does not include all the small details of real materials, it still helps us understand general behavior in statistical physics. The model has also been used in other areas, such as neuroscience, social behavior, and image processing[3].

In this project, we investigate the two-dimensional ferromagnetic Ising model on a square lattice using Monte Carlo methods in Python. In particular, we apply the Metropolis algorithm[4] to simulate the system in thermal equilibrium and to compute thermodynamic observables such as energy, magnetization, specific heat, and magnetic susceptibility as functions of temperature. We pay special attention to the emergence of critical behavior near the phase transition point and examine how thermodynamic quantities fluctuate as the system approaches T_c. The Metropolis method, developed in the early days of computing, remains one of the most widely used sampling techniques for equilibrium statistical physics[5,6].

The report is organized as follows: In Section II, we present the theoretical background of the Ising model and describe the Metropolis Monte Carlo algorithm. In Section III we show and discuss the results of the simulations. Finally Section IV provides the conclusions and perspectives for further study.

II. MODELS AND METHODS

A. Models and Theoretical Background

All equations that are not cited directly in this section are obtained from the exam paper.

1. 2D Ising Model

The two-dimensional Ising model describes electrons in a material by placing discrete spin variables on a lattice, where each spin can point either up $(+1)$ or down (-1). In this model, each spin interacts with its four nearest neighbors and with an external magnetic field H. The total energy of the system is given by:

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

where the first term sums over all pairs of nearest-neighbor spins s_i and s_j, and the second term accounts for the interaction with the external magnetic field. Here, J represents the interaction strength between neighboring spins, and μ is the Bohr magneton. By looking at the equation for the total energy we see that the energy is minimized when spins have the same direction, as the first term becomes more negative. We can also infer that spins will align with the magnetic field.

We can now find the equation for the change in energy when we flip one of the spins in the lattice:

$$\Delta E = 2s_i \left(J \sum_{\langle i,j \rangle} s_j + \mu H \right). \tag{2}$$

For much of this project we investigate the situation where $H = 0$, thus eliminating the last part from Equation (2).

2. Metropolis algorithm

As we now have the equation for the energy change, we can apply the Metropolis algorithm to the Ising model to determine how the system evolves. The algorithm proceeds as follows:

1. Initialize the lattice by assigning random spin values $s_i = \pm 1$.
2. For each Monte Carlo cycle:
   (a) Select a random lattice site and propose to flip its spin s_i.
   (b) Compute the change in energy ΔE resulting from this proposed flip.
   (c) Get the acceptance probability $P^* = \exp(-\frac{\Delta E}{k_b T})$.
   (d) Draw a random number r from the interval $[0, 1)$.
   (e) If $r < P^*$, accept the move by flipping the spin; otherwise, leave the configuration unchanged.

By following this algorithm, the system gradually evolves toward more probable configurations without the need to evaluate the energy of every possible state—an approach that would be computationally infeasible even for relatively small systems due to the exponential number of configurations. Once the system has reached equilibrium, physical observables such as the magnetization can be estimated using simple, unweighted averages over the sampled configurations:

$$M = \frac{1}{N_{sites}} \sum_i s_i. \tag{3}$$

The magnetization ranges from -1 to 1, reaching its maximum absolute value when all spins are aligned, and approaching zero when the spins are completely disordered.

3. Specific Heat

The specific heat of the system can be expressed in terms of energy fluctuations using the relation:

$$C = \frac{\mathrm{Var}(E)}{k_B T^2}, \tag{4}$$

where $\mathrm{Var}(E) = \langle E^2 \rangle - \langle E \rangle^2$ is the variance of the system's energy, k_B is the Boltzmann constant, and T is the temperature. This formula is a consequence of the fluctuation-dissipation theorem, which connects response functions, such as specific heat, to equilibrium fluctuations in the corresponding observable[7].

The specific heat can also be shown to follow a power law of the form

$$C \sim \frac{1}{|T - T_c|^\alpha}, \tag{5}$$

where α is a critical exponent, and T_c is the critical temperature where a phase transition occurs. By analyzing the graph obtained from Equation (4) one can thus find the critical temperature as the temperature where the specific heat diverges. It is also possible to find the exponent α by investigating the sharpness of the divergence.

4. Magnetic Susceptibility

If we activate an external magnetic field we can measure the magnetic susceptibility of the system by a similar relation as for the specific heat:

$$\chi = \frac{\mathrm{Var}(M)}{k_B T}, \tag{6}$$

where we instead use the variance of the magnetization M.

For the two-dimensional Ising model, the scaling behavior of the magnetic susceptibility can be written as

$$\chi(t, h) = |t|^{-\gamma} g_\pm \left(\frac{h}{|t|^{\beta\delta}} \right), \tag{7}$$

where $t \approx (T - T_c)/T_c$ and $h \equiv \mu H / J$ is the dimensionless magnetic field. The function $g_\pm$ describes the scaling behavior above and below T_c, while γ, β, and δ are critical exponents with known values for the 2D Ising model: $\gamma = \frac{7}{4}$, $\beta = \frac{1}{8}$, and $\delta = 15$.

B. Implementation in Python

1. Without Magnetic Field

Firstly a lattice with values (-1) and (+1) placed randomly was initialized using the function `random.choice()` from the NumPy library[8]. I then created a `sweep_lattice` function that performs one iteration of the Metropolis algorithm as described in Section II A 2. To avoid edge-effects when sweeping the lattice, periodic boundary conditions were used when calculating the energy change. Practically this was implemented by using the modulo operator, which automatically wraps around when reaching the edge.

Then I created the main function of the program: `run_sweeps`, which runs a specified number of sweeps over the lattice, at a given temperature. The function also calculates and keeps track of the necessary quantities: magnetization M, energy E and the variance $\mathrm{Var}(E)$ of the energy throughout the sweep. This function could then be used in for-loops to get the temperature evolution of the system. Snapshots of the system were also obtained inside the for-loops.

For simplicity, and to express temperature as a dimensionless quantity, both the interaction strength J and the Boltzmann constant k_B were set to 1 in all simulations. Monte Carlo simulations were performed for lattice sizes $L = 10, 20, 30$, and 40, where $L \times L$ corresponds to the total number of spins. Each simulation was run for 10,000 lattice sweeps. The temperature was varied from $T = 1$ to $T = 4$ using 300 evenly spaced values to capture the behavior near the critical region.

Due to the large number of operations required for each simulation, especially at higher lattice sizes and temperature resolutions, the code initially had long run times. To address this, the `@jit` decorator from the *Numba* library was used in *nopython* mode. Numba is a Just-In-Time (JIT) compiler that translates Python functions into highly optimized machine code[9]. By applying `@jit(nopython=True)` to the performance-critical functions, the execution speed was significantly improved, making large-scale simulations feasible.

However, translating the functions into machine code introduced an unexpected issue related to the handling of random numbers. By default, NumPy uses the Mersenne Twister algorithm, which generates high-quality pseudo-random numbers with an extremely long period of $2^{19937} - 1$[10]. When using `@jit(nopython=True)`, however, Numba replaces this with its own random number generator: XORSHIFT128+. This generator is extremely fast and efficient, but has a shorter period of $2^{128} - 1$[11]. While this is still more than sufficient for our simulations, it complicates the process of setting and controlling random seeds, and this was therefore not done in the present project. The implications of this choice are discussed further in the results section.

2. *Impurities*

Additional simulations were performed with frozen impurity sites—lattice sites where the spin is fixed and cannot flip. This was implemented by creating an impurity mask during lattice initialization. The rest of the simulation procedure remained unchanged, except for an added condition that skips the spin-flip step if the selected site is marked as an impurity. This was tested using three different values of the impurity fraction, $p = 0.01$, 0.1, and 0.3, which determine the proportion of lattice sites set as impurities. For each case, the energy, magnetization, and visual snapshots of the system were recorded and analyzed.

In addition to the standard simulations, a technique known as *Simulated Annealing (SA)* was also implemented. In SA, the system is gradually cooled by performing a limited number of sweeps at each temperature, allowing it to settle into low-energy configurations over time[12]. The method is inspired by the physical process of annealing in metallurgy, where a material is slowly cooled to reduce defects and minimize its internal energy.

3. *With External Magnetic Field*

Lastly, simulations were carried out with an external magnetic field applied to the system. The field strength was set to $H = 0.01$, 0.03, and 0.05, with the Bohr magneton μ set to 1 for simplicity.

III. RESULTS

A. Without External Magnetic Field

Figure 1 shows the time evolution of the magnetization for three different temperatures at lattice size $L = 40$. At $T = 2.1$, the system quickly settles into a state where nearly all spins are aligned in the -1 direction, resulting in a stable and strongly negative magnetization. In contrast, the simulations at $T = 2.3$ and $T = 2.5$ display more irregular behavior with significant fluctuations. While both fluctuate, the magnetization at $T = 2.3$ tends to remain in the (+1) direction, whereas at $T = 2.5$, it consistently averages close to zero, indicating a disordered, high-temperature phase.

An important observation from the figure is that the system takes some time to reach equilibrium. This is particularly evident at $T = 2.3$, where large fluctuations occur during the initial sweeps, while gradually moving toward more stable behavior after around 5000 sweeps. Because of this, further simulations were consistently run with 10,000 sweeps per temperature, and averages of the magnetization and energy were computed only after equilibrium was met.

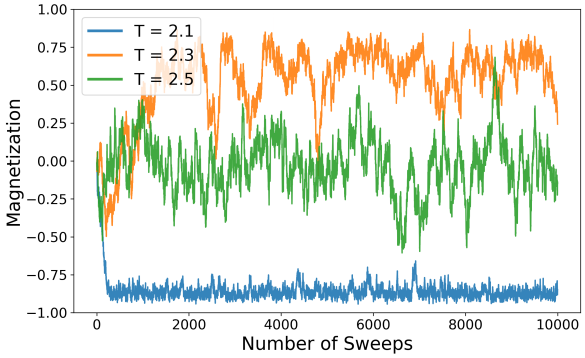

FIG. 1. Time evolution of magnetization at three different temperatures: $T = 2.1$, 2.3, and 2.5.

Figure 2 shows the absolute value of the magnetization and the energy per spin, obtained from a temperature sweep ranging from $T = 1$ to $T = 4$ in 300 steps for a system with $L = 40$. Figure 3 presents snapshots of the spin configurations from the same temperature sweep, where red indicates spins with values $(+1)$ and blue indicates (-1). The magnetization exhibits a sharp drop around $T = 2.3$, which closely matches the analytical value of the critical temperature for the two-dimensional Ising model, $T_c \approx 2.27$[2]. At this temperature, the system undergoes a phase transition from an ordered (magnetized) state to a disordered one. This is also evident from the snapshot at $T = 2.3$ where we see more red spins appearing in the "sea" of (-1) spins. As spins become increasingly unaligned due to thermal fluctuations, the magnetization reduces and the energy increases and we reach a completely unordered phase as visualized in the snapshot at $T = 2.5$.

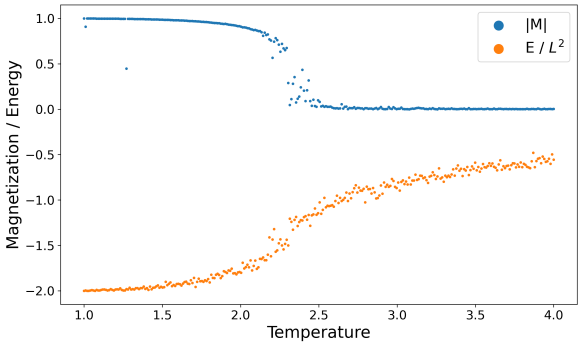

FIG. 2. Temperature evolution of magnetization and energy for a system with $L = 40$. The magnetization is displayed as its absolute value $|M|$ and the energy is shown as the energy per spin E/L^2.

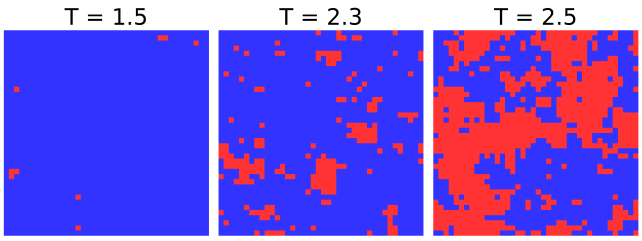

FIG. 3. Snapshots of the system at three different temperatures $T = 1.5$, 2.3, and 2.5. Blue spots indicate spins with value (-1) while red indicates $(+1)$.

Figure 4 shows the specific heat, calculated using Equation (4), as a function of temperature for all simulated lattice sizes. The plots also indicate the estimated critical temperatures, identified as the points where the specific heat reaches its maximum. Several observations can be made from this figure. First, for some lattice sizes—particularly $L = 40$—there are isolated points with unusually high specific heat values. These were identified as outliers likely caused by statistical fluctuations and were excluded from the estimation of T_c, as they did not follow the general trend of the curve.

Second, we observe that the estimated critical temperatures for smaller lattices deviate significantly from the analytical value $T_c \approx 2.27$. This discrepancy is expected due to finite-size effects, as the analytical result is derived in the thermodynamic limit, where $L \to \infty$[2].

Due to limited resolution and statistical noise, the specific heat curves were not smooth or precise enough to reliably extract the critical exponent α from Equation (5). A possible approach to improve this would be to increase the number of temperature points around the critical region to better resolve the shape of the peak.

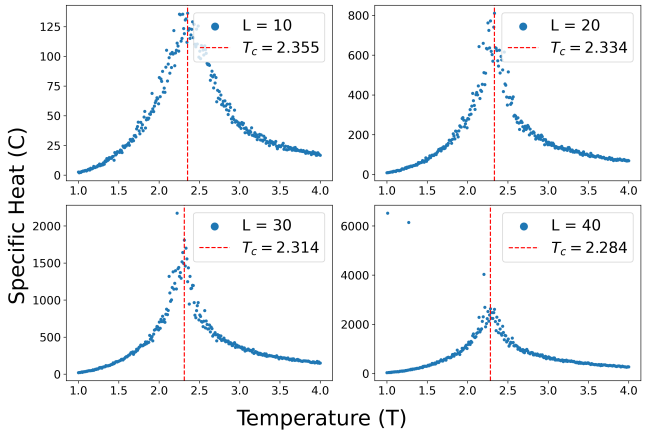

FIG. 4. Specific heat of all the lattice sizes as a function of temperature. Critical temperatures are displayed as the striped red lines.

While running these simulations, I encountered one of the key limitations of Monte Carlo methods: their strong dependence on sample size and statistical fluctuations. As discussed in Section II B, the use of Numba's `nopython` mode made it difficult to control or fix the random seed. As a result, each run of the simulation—though identical in setup—produced slightly different results for the estimated critical temperature. In some cases, T_c was close to the analytical value; in others, it deviated noticeably. This variability highlights a fundamental property of Monte Carlo simulations: meaningful and reliable results require large sample sizes and averaging over many independent runs.

To improve accuracy, one could perform a convergence analysis by increasing the number of sweeps per temperature step, running multiple independent simulations, and averaging the results. Repeating temperature sweeps and comparing outcomes would help identify whether key observables, like the peak of the specific heat, stabilize with increasing sample size. These approaches are essential for reducing statistical noise and making robust conclusions in large-scale Monte Carlo studies—something that was beyond the scope of this project but remains an important consideration for future work.

B. With Impurities

Figure 5a) shows the magnetization and energy per spin, while Figure 5b) displays the specific heat, all plotted for dif-

ferent impurity concentrations at $L = 40$. As the fraction of impurity sites increases, the magnetization decreases, since frozen spins disrupt the system's ability to align. This also leads to a reduction in the critical temperature, as seen from the shift of the specific heat peak in panel b).

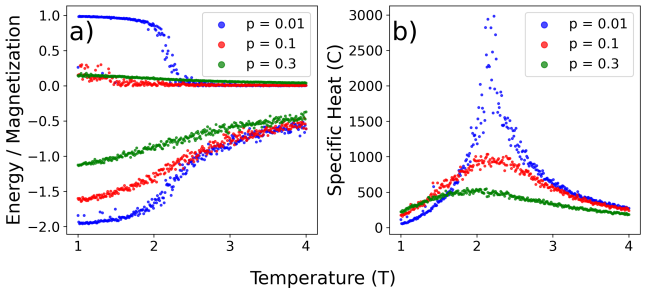

FIG. 5. Results from a temperature sweep of lattices with different fractions of impurity sites at $L = 40$. The upper points in panel a) shows the absolute value of the magnetization while the points at the bottom correspond to the energy per spin. The specific heat is displayed in panel b). In both panels the quantities are plotted as functions of temperature.

An interesting observation is that while the magnetization for $p = 0.1$ and $p = 0.3$ is often similar across many temperatures, the energy for $p = 0.3$ is consistently lower. This can be explained by the formation of spin "islands"—clusters of aligned spins that form around the frozen impurity sites. These islands contribute little to the overall magnetization, since their orientations may cancel out, but they still minimize energy internally. The energy cost arises mainly from the spins at the boundaries of these islands, where unfavorable interactions occur. At lower impurity concentrations (e.g., $p = 0.1$), these islands tend to be larger and more isolated due to less impurities. This results in fewer boundary sites per island, effectively reducing the system's total "surface energy" and leading to a lower overall energy compared to systems with smaller, more fragmented islands as in the $p = 0.3$ case.

The results of the simulated annealing runs for $L = 40$, performed from $T = 5$ down to $T = 0.1$ in 10,000 steps with 5 sweeps per temperature, are summarized in Table I. The data follow the same general trend observed in Figure 5: as the impurity concentration increases, the energy becomes higher and the magnetization moves closer to zero, reflecting the system's increasing disorder.

Impurity p	Magnetization M			Final energy E/L^2		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
0.01	-0.99625	0.99000	-0.99000	-1.9850	-1.9650	-1.9600
0.10	-0.23125	-0.13625	-0.19500	-1.6300	-1.6075	-1.6325
0.30	-0.14000	0.10625	0.18625	-1.1700	-1.1550	-1.2250

TABLE I. Final magnetization and energy per spin from three simulated annealing runs for each impurity probability. Simulations were performed on an $L = 40$ lattice, annealed from $T = 5$ to $T = 0.1$ in 10,000 steps with 5 sweeps per temperature step.

Snapshots of the lowest-energy configurations from the simulated annealing runs are shown in Figure 6, where the darker sites represent frozen spin impurities. The snapshots clearly illustrate the formation of spin "islands" around the impurities, as discussed earlier.

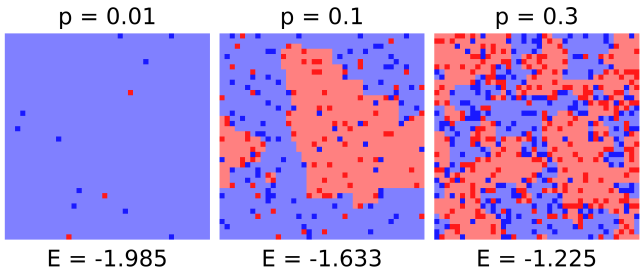

FIG. 6. Snapshots of the lowest energy configurations during simulated annealing.

C. With External Magnetic Field

A visualization of the magnetization over time, along with a snapshot of the final configuration from a simulation at $T = 2.3$ and $H = 0.01$, is shown in Figure 7. Compared to the corresponding case without an external field (Figure 1), the system exhibits a much more stable and higher magnetization at the same temperature. This behavior is a direct consequence of the magnetic field, which tends to align spins in its direction in order to minimize the system's energy. Since $H > 0$, the majority of spins point in the $+1$ direction. If the field were negative, the system would instead stabilize at a negative magnetization.

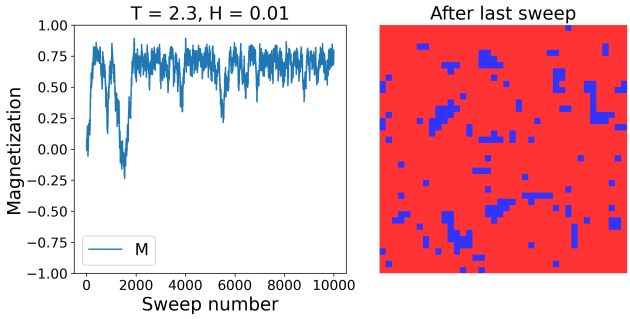

FIG. 7. The left plot shows the time evolution of magnetization with an external magnetic field $H = 0.01$ at $T = 2.3$. The right panel displays the system after the last sweep.

Figure 8 shows the absolute magnetization and magnetic susceptibility from a temperature sweep at different magnetic field strengths. As the field increases, the drop in magnetization shifts to higher temperatures. This is because the magnetic field promotes spin alignment, requiring more thermal energy to push the system into a disordered state. Consequently, the transition appears to occur at a higher temperature.

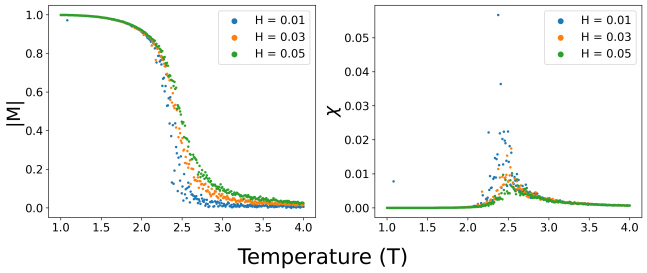

FIG. 8. The left panel displays the absolute magnetization as a function of temperature at different strengths of an external magnetic field H. The right panel shows the magnetic susceptibility at the same field strengths.

To study data collapse, Equation (7) was used together with the known critical exponents listed. The rescaled susceptibility $\chi|t|^{\gamma}$ was plotted against the scaled magnetic field $h/|t|^{\beta\delta}$ for different values of H, as shown in Figure 9. The resulting plot shows a clear collapse of the data into two distinct branches—one corresponding to temperatures above T_c, and one below. This successful collapse provides strong evidence that the system follows the expected scaling behavior near the critical point.

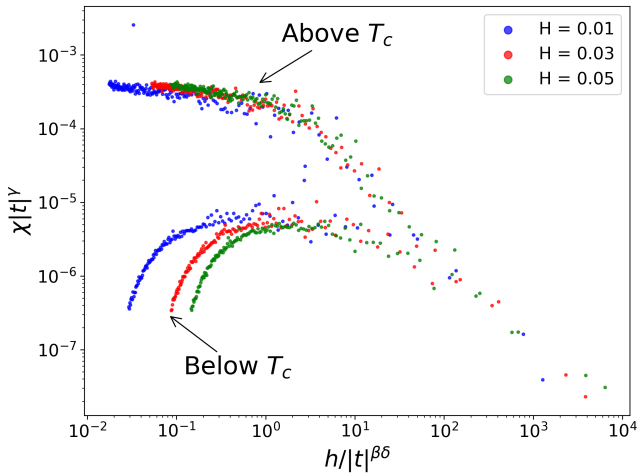

FIG. 9. Visualization of the data collapse of the magnetic susceptibility χ at different field strengths.

IV. CONCLUSIONS

In this project, the two-dimensional Ising model was investigated using Monte Carlo simulations based on the Metropolis algorithm. Key thermodynamic quantities such as energy, magnetization, specific heat, and magnetic susceptibility were computed as functions of temperature, allowing for identification of the critical temperature T_c through both direct observation and analysis of fluctuations. The results showed good agreement with the known analytical value $T_c \approx 2.27$, especially for larger system sizes.

The introduction of frozen impurity sites demonstrated how disorder influences the system's behavior. Impurities disrupt the alignment of spins, preventing large-scale ordering and thereby reducing the overall magnetization. As a result, the system transitions to a disordered state at a lower temperature, effectively lowering the observed critical temperature. Simulated annealing runs supported this interpretation by revealing low-energy configurations characterized by spin "islands" constrained by the impurities. Additionally, simulations with an external magnetic field illustrated how the field promotes spin alignment and shifts the transition to higher temperatures. A successful data collapse of the rescaled magnetic susceptibility further confirmed the expected scaling behavior near the critical point.

Throughout the project, it became clear that the statistical nature of Monte Carlo methods places strong demands on sample size and averaging. Limitations related to random number generation and finite sampling were acknowledged, and possible improvements—such as convergence analysis and multiple independent simulations—were discussed.

Overall, the simulations captured the key features of the 2D Ising model and provided insight into critical phenomena and the effects of disorder and external fields. The study highlights the power of computational methods in exploring physical systems and lays the groundwork for future investigations with improved accuracy, larger system sizes, or more complex models.

V. REFERENCES

[1] E. Ising, "Beitrag zur theorie des ferromagnetismus," Zeitschrift für Physik **31**, 253–258 (1925).
[2] L. Onsager, "Crystal statistics. i. a two-dimensional model with an order-disorder transition," Physical Review **65**, 117–149 (1944).
[3] S. Galam, "Sociophysics: a review of galam models," International Journal of Modern Physics C **19**, 409–440 (2008).
[4] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," The Journal of Chemical Physics **21**, 1087–1092 (1953).
[5] M. E. J. Newman and G. T. Barkema, *Monte Carlo Methods in Statistical Physics* (Oxford University Press, 1999).
[6] D. P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, 2nd ed. (Cambridge University Press, 2005).
[7] L. E. Reichl, *A Modern Course in Statistical Physics*, 4th ed. (Wiley-VCH, 2016).
[8] T. E. Oliphant *et al.*, "Numpy v1.21.0 documentation," (2021), accessed: 2023-03-17.
[9] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC , 1–6 (2015).
[10] P. S. Foundation, "The python standard library: random module," `https://docs.python.org/3/library/random.html` (2025), accessed: February 17, 2025.
[11] S. Vigna, "Further scramblings of marsaglia's xorshift generators," arXiv preprint (2014), arXiv:1402.6246 [cs.DS].
[12] Wikipedia contributors, "Simulated annealing — wikipedia, the free encyclopedia," (2024), accessed: 2025-04-30.

VI. ACKNOWLEDGMENTS

I would like to thank candidates 10075, 10039 and 10037 for sharing and discussing results, which helped shape a better final report.

VII. AI DECLARATION

AI (ChatGPT-4o) was used to assist with clearer language and sentence construction. All contributions from the AI have been carefully reviewed by me, the author, before being included in the final report.

VIII. PROBLEM 2: SHORT QUESTIONS

All tasks were done in Python.

A. Task a)

$x = 10^8$ and $y = 1.0$ were cast into the floating precisions float32, float64 and longdouble. When calculating the value $z = (x + y) - x$, we expect mathematically to end up with $z = y = 1.0$. However the operation for the variables defined as float32 returned $z = 0.0$. This is because float32 only has a memory usage of 32 bits and thus a precision of only 7 digits. This means that $10^8 + 1.0 = 10^8$ because the 1.0 is below the 7-digit threshold. For the variables defined as float64 and longdouble, we get the correct result because their higher precisions—approximately 15 digits for float64 and 19 or more for longdouble—are sufficient to represent the added value accurately.

B. Task b)

When defining $a = 0.1 + 0.2$ and $b = 0.3$, the operation $a == b$ will return False. This is because of floating point precision. The decimal numbers in a and b cannot be represented exactly in binary, thus when trying to compare them we are actually comparing different values. By checking we find that the actual values stored are: $a = 3.0000000000000004, b = 0.29999999999999999$. A workaround to compare the numbers is to use a function like np.isclose(a, b), which will compare with a certain tolerance that you can set yourself.

C. Task c)

When doing the recurrence formula that adds 10^{-16} to 1.0 a total of 10^6 times, we still end up with $x = 1.0$. This is because of the machine epsilon value. The machine epsilon is the smallest number that, when added to 1.0, gives a result distinguishable from 1.0 in a given floating-point format. The machine epsilon for float64 is $\approx 2.2 \cdot 10^{-16}$, and this can also be checked by running `np.finfo(float).eps`. This means that any values lower than this machine epsilon—such as 10^{-16} in our case—will not contribute when added to the existing value, thus leaving x at the same value 1.0.

D. Task d)

Calculating the factorial of 170 and converting to float (double precision arithmetic, 64 bits) will work, however doing the same for 171 will give an overflow error message. This is because the maximum number that can be represented with double precision floating point (IEEE-754) is $\approx 1.8 \cdot 10^{308}$. The factorial of 170 is $\approx 7.26 \cdot 10^{306}$, which is below the maximum limit, however $171! \approx 1.24 \cdot 10^{309}$, which exceeds the limit, thus leading to overflow.

Due to the overflow, calculating the difference led to an error message in python, however by doing some mathematics we can use that

$$171! = 171 \times 170!,$$

thus:

$$\text{difference} = 171! - 170! = 170! \times (171 - 1) = 170! \times 170$$

which we can put into python without getting an error message. This will however give us output of `inf`, as what we are basically calculating is

$$\text{infinite} - \text{finite} = \text{infinite}.$$

E. Task e)

The problem we get is that we subtract two numbers that are both extremely close to 1. A small rounding error in either one of them leads worse floating point precision in the final answer. To address this one can use the identity

$$\sqrt{a} - \sqrt{b} = \frac{a - b}{\sqrt{a} + \sqrt{b}}.$$

Adding the square roots is not as vulnerable to rounding errors, meaning we get a much more exact result in the denominator. However this will still not be perfect, as we still have two numbers very close to 1 that are subtracted in the numerator, however they are larger than their square roots, and will thus produce a better result.