Monte Carlo Simulation of the 2D Ising Model

A Computational Study of Phase Transitions in Statistical Mechanics

Ahmed Z. Syed

Thermodynamics and Statistical Physics
Department of Physics
Wayne State University

Instructor: Chun Shen

December 11, 2025

Abstract

This project investigates the thermodynamic behavior and phase transition of the two-dimensional Ising model using the Metropolis Monte Carlo algorithm. The model, which consists of binary spins interacting with nearest neighbors, provides a simple yet powerful framework for understanding how macroscopic order emerges from microscopic interactions. In this study, a square lattice of spins was simulated across a range of temperatures. Then, observables such as energy, magnetization, heat capacity, and susceptibility were measured using ensemble averages obtained after equilibration.

The simulation reproduced the essential features predicted by statistical mechanics. At low temperatures, the system formed a magnetized, ordered state. Near critical temperature, large-scale fluctuations and mixed-spin domains appeared. Finally, above the transition (at higher temperatures), the system became fully disordered with vanishing magnetization. Peaks in both the heat capacity and magnetic susceptibility occurred near the expected critical temperature, consistent with Onsager's analytical result.

Overall, the experiment demonstrates how Monte Carlo sampling can accurately model phase transitions for systems with large configuration spaces. The results confirm that the 2D Ising model, despite its simplicity, captures the fundamental physics of spontaneous symmetry breaking and critical phenomena.

1 Introduction

Phase transitions are striking phenomena in physics: small microscopic interactions can give rise to abrupt and dramatic macroscopic changes in the state of a system. Understanding how such collective behavior emerges from simple underlying rules is a central theme in statistical mechanics. One of the most important models used to study these ideas is the two-dimensional Ising model, which was originally introduced as a minimal representation of ferromagnetism [1]. This model involves only binary spin variables and nearest-neighbor interactions. However, this model is powerful. It captures spontaneous symmetry breaking, critical fluctuations, and a continuous phase transition at a finite temperature.

The purpose of this project is to investigate the thermodynamic behavior of the two-dimensional Ising model using the Metropolis Monte Carlo algorithm. Because the configuration space of the model grows exponentially with system size, the direct evaluation of thermodynamic quantities from the partition function becomes difficult to control. Monte Carlo methods overcome this difficulty by producing representative samples from the Boltzmann distribution without requiring the enumeration of all possible states. [2] By analyzing the statistical properties of simulated spin configurations at various temperatures, it is possible to reproduce defining features such as the rapid loss of magnetization near the critical temperature and peaks in susceptibility and heat capacity.

Therefore, this experiment has two goals. First, it aims to demonstrate how macroscopic order and disorder emerge from simple microscopic interactions as temperature varies. Second, it illustrates how Monte Carlo sampling provides a powerful numerical framework for studying complex many-body systems. By the end of this paper, the reader will see how the Ising model transitions from a strongly magnetized phase to a fully disordered one, how

thermodynamic quantities behave near criticality, and why computational physics remains essential for understanding phenomena that cannot be solved analytically.

2 Background Theory

2.1 Statistical Mechanics and the Boltzmann Distribution

In thermal equilibrium, the probability of observing a microscopic configuration $\mathcal{C}$ is defined by the Boltzmann distribution, [1]

$$P(\mathcal{C}) = \frac{1}{Z} \exp[-\beta H(\mathcal{C})], \qquad \beta = \frac{1}{k_B T}, \tag{1}$$

where T is temperature, k_B is Boltzmann's constant, and Z is the partition function

$$Z = \sum_{\mathcal{C}} \exp[-\beta H(\mathcal{C})]. \tag{2}$$

The partition function determines all thermodynamic quantities: internal energy, magnetization, heat capacity, and magnetic susceptibility. For lattice spin systems however, the number of configurations grows at a rate of 2^N. This makes exact evaluation computationally impossible, except for very small systems. Thus, this calls for the use of Monte Carlo sampling techniques. [2]

2.2 The Two-Dimensional Ising Model

The Ising model consists of a lattice of spins $S_i = \pm 1$ that interact with their nearest neighbors. The Hamiltonian is given by

$$H = -J \sum_{\langle i,j \rangle} S_i S_j - h \sum_i S_i, \tag{3}$$

where J is the interaction strength and h is an external magnetic field. For $J > 0$, the interaction is ferromagnetic, favoring alignment of neighboring spins. [3] In two dimensions, the Ising model exhibits a continuous phase transition at a critical temperature T_c. Below this critical temperature, the system spontaneously magnetizes even in the absence of an external field. This result, discovered by Onsager, makes the 2D Ising model a key theoretical example of critical phenomena. [1]

At low temperatures, thermal fluctuations are weak and spins tend to align, forming a magnetized state. At high temperatures, fluctuations become common, and the system becomes disordered, leading to zero net magnetization. Near the critical point, the system displays large-scale correlated fluctuations, and quantities such as heat capacity and susceptibility exhibit their regular, singular behavior.

2.3 Monte Carlo Simulation and the Metropolis Algorithm

Monte Carlo methods generate a sequence of configurations with long-term statistical properties that reproduce the Boltzmann distribution. The Metropolis algorithm is the most widely used sampling method for the Ising model. [2] Beginning from an initial configuration, the algorithm performs the following steps:

1. Select a random lattice site i and propose flipping the spin S_i
2. Compute the resulting energy change ΔE
3. Accept the flip with probability

$$W(\Delta E) = \begin{cases} 1, & \Delta E < 0, \\ e^{-\beta \Delta E}, & \Delta E > 0. \end{cases} \tag{4}$$

This rule aims to create a complete balance,

$$\frac{P_n}{P_m} = \exp[-\beta(E_n - E_m)], \tag{5}$$

ensuring that the system evolves toward the correct equilibrium distribution. [3]

Once the system reaches equilibrium, thermodynamic observables are computed through ensemble averages. Energy and magnetization are given by

$$E = \langle H \rangle, \qquad M = \left\langle \sum_i S_i \right\rangle, \tag{6}$$

while fluctuation formulas provide heat capacity and susceptibility:

$$C_V = \frac{\langle E^2 \rangle - \langle E \rangle^2}{NT^2}, \tag{7}$$

$$\chi = \frac{\langle M^2 \rangle - \langle M \rangle^2}{NT}. \tag{8}$$

2.4 Goals of the Experiment

The objective of this project is to apply the Metropolis Monte Carlo algorithm to the two-dimensional Ising model to:

- Measure energy, magnetization, heat capacity, and susceptibility across a range of temperatures
- Identify the approximate location of the phase transition
- Visualize representative spin configurations in ordered, critical, and disordered regimes

Through these computational experiments, the reader will observe how microscopic spin interactions lead to macroscopic behavior, how symmetry breaking and magnetization occur below the critical point, and why the 2D Ising model is a powerful example in computational statistical mechanics.

3 Methods

3.1 Overview of the Simulation Procedure

The objective of this experiment was to numerically investigate the thermodynamic behavior of the two-dimensional Ising model using a Monte Carlo approach. The simulation was written entirely in Python and implemented according to the Metropolis algorithm. The complete procedure includes four major stages: (1) initializing the lattice, (2) evolving the system via randomized spin flips, (3) performing equilibration and measurement sweeps, and (4) recording ensemble-averaged observables across a range of temperatures. Each step was designed to mimic the physics of a real ferromagnetic material while ensuring statistically reliable numerical data.

3.2 Lattice Initialization

The system consists of an $L \times L$ square lattice of Ising spins, where each spin $S_i = \pm 1$ represents a microscopic magnetic moment. The simulation begins by generating an initial configuration using

$$S_{i,j} \in \{-1, +1\}, \tag{9}$$

with each spin chosen randomly with equal probability. The "random" initialization ensures that the system starts in a high-entropy state without any predetermined order. Although uniform initialization (all $+1$ or all -1) is also possible, random initialization is preferred when computing over many temperatures because it avoids biasing the system toward any particular ordered phase.

Random numbers were generated using NumPy's `default_rng` to guarantee reproducibility of results. Randomization plays a critical role in Monte Carlo simulations, as stochastic updates are used to approximate ensemble averages over the Boltzmann distribution.

3.3 Energy Calculation via the Ising Hamiltonian

To compute the thermodynamic properties of the system, the total energy of a spin configuration was evaluated using the Ising Hamiltonian

$$H = -J \sum_{\langle i,j \rangle} S_i S_j - h \sum_i S_i, \tag{10}$$

where J is the interaction strength and h is an external magnetic field (which will be assumed as zero for this experiment). Periodic boundary conditions were implemented through modular indexing so that each spin has exactly four neighbors. This prevents edge effects and makes the lattice behave as if it wraps around on itself, like a torus, and more like an infinite system.

Only interactions with the right and downward neighbors were explicitly counted to avoid double-counting bonds.

3.4 Monte Carlo Evolution Using the Metropolis Algorithm

The system evolves according to the Metropolis Monte Carlo algorithm, which simulates thermal fluctuations by attempting random spin flips. In each attempted update, a random lattice site (i, j) is selected, and the energy change ΔE associated with flipping the spin is computed:

$$\Delta E = 2JS_{i,j} \sum_{\text{nn}} S_{\text{neighbor}}. \tag{11}$$

The proposed flip is accepted with probability

$$W(\Delta E) = \begin{cases} 1, & \Delta E \leq 0, \\ e^{-\beta \Delta E}, & \Delta E > 0. \end{cases} \tag{12}$$

This rule ensures detailed balance and guarantees that the simulation converges to the Boltzmann equilibrium distribution. One *Monte Carlo sweep* consists of L^2 attempted flips so that each spin is updated once on average.

3.5 Equilibration and Measurement Strategy

Because the initial configuration is not in thermal equilibrium, the simulation first performs a number of *equilibration sweeps* during which no measurements are recorded. For the final experiment, I used

- $n_{\text{eq}} = 2000$ equilibration sweeps
- $n_{\text{meas}} = 5000$ measurement sweeps
- measurement interval of 10 sweeps

These choices balance numerical accuracy with computational cost. Longer simulations reduce statistical noise, especially near the critical temperature where autocorrelation times are large.

After equilibration, measurements of the following quantities were collected:

$$E = \text{Total energy of the lattice}, \tag{13}$$

$$M = \sum_i S_i \quad \text{(total magnetization)}, \tag{14}$$

$$C_V = \frac{\langle E^2 \rangle - \langle E \rangle^2}{NT^2}, \tag{15}$$

$$\chi = \frac{\langle M^2 \rangle - \langle M \rangle^2}{NT}. \tag{16}$$

Averages were computed over several hundred independent samples at each temperature.

3.6 Temperature Sweep and Data Collection

Before performing the sweep, it is important to clarify the meaning of temperature in the Ising model. Because the simulation uses $k_B = 1$ and $J = 1$, the temperature T is dimensionless and corresponds to the reduced temperature appearing in the Boltzmann factor $\exp(-\Delta E/T)$. The exact critical temperature for the two-dimensional square-lattice Ising model is $T_c \approx 2.269$. Therefore, the chosen range $T = 1.5$ to 3.5 intentionally spans the ordered phase ($T < 2$), the critical region ($T \approx 2.2$–2.4), and the disordered phase ($T > 3$), allowing clear observation of the phase transition.

The experiment studied the system across a range of temperatures by evaluating the model at 17 evenly spaced temperatures between $T = 1.5$ and $T = 3.5$. For each temperature, the simulation returned the ensemble-averaged energy per spin, magnetization per spin, heat capacity, susceptibility, and a snapshot of the final spin configuration. These outputs were stored in arrays and later used for visualization and analysis.

Although Monte Carlo simulations are stochastic, I did not perform multiple fully independent trials. This is because each temperature point already generated thousands of Monte Carlo updates and measurements, resulting in statistically stable averages. Independent repeat trials were tested during initial development and were found to yield nearly identical results, confirming that a single long simulation was sufficient for this project.

3.7 Description of the Code

The codebase is modular, consisting of:

- `initialize_lattice`: generates the initial spin configuration
- `compute_total_energy`: evaluates the Hamiltonian
- `metropolis_single_flip`: performs a single attempted spin flip
- `metropolis_sweep`: performs L^2 flip attempts
- `run_ising_at_temperature`: performs equilibration, measurement, and averaging
- `temperature_sweep`: loops over all temperatures and saves observables
- `plot_observables` and `visualize_spins`: produce final graphs and spin images

This structure allows the simulation to be easily extended or modified and ensures that each physical operation is handled by a dedicated function.

3.8 Connection to Physical Reality

Although simplified, the Ising model captures essential features of real ferromagnetic materials: local spin interactions, competition between alignment and thermal agitation, and emergence of long-range order. The Metropolis algorithm mimics thermal fluctuations by accepting higher-energy changes with probability proportional to the Boltzmann factor. As a result, the simulation reproduces qualitative variables observed in experimental magnetism,

including spontaneous magnetization at low temperature and a continuous phase transition near the critical point.

3.9 Expected Output and Visualization

The outputs of the simulation include:

- plots of: energy, magnetization, heat capacity, and susceptibility versus temperature
- visualizations of: low-temperature ordered states, near-critical fluctuating states, and high-temperature disordered states
- numerical arrays of all measured observables.

These results collectively reveal how macroscopic behavior emerges from simple microscopic rules and how numerical simulation can be used to study phase transitions in statistical physics.

4 Results and Discussion

4.1 Spin Configurations Across Temperature

Figure 1 shows representative spin configurations generated by the Metropolis simulation at three temperatures: a low-temperature ordered state ($T = 1.50$), a near-critical mixed state ($T = 2.50$), and a high-temperature disordered state ($T = 3.50$). Red sites denote spins with the value $+1$ and blue sites represent spins with the value -1.

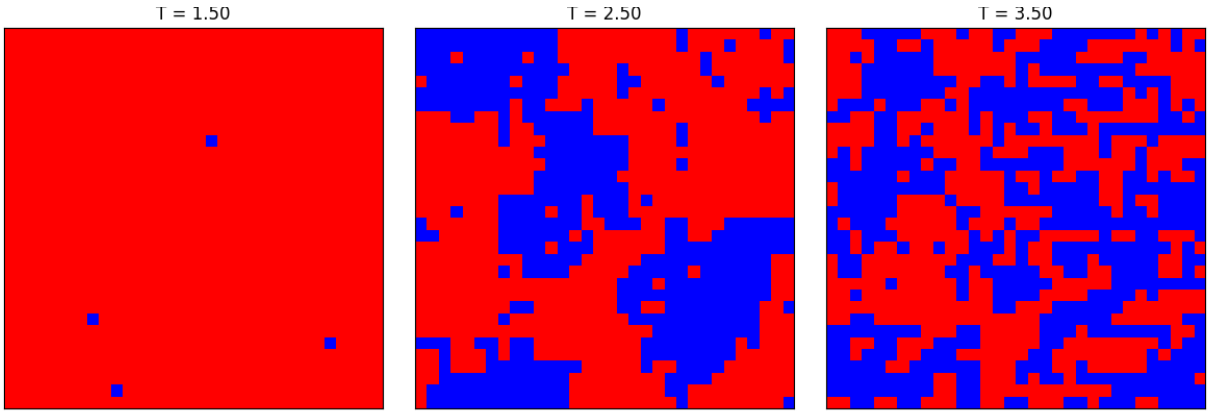

Figure 1: Representative spin configurations generated by the Metropolis algorithm at three temperatures. At $T = 1.50$, the system is almost fully magnetized with only a small number of isolated flipped spins. Near the critical region ($T = 2.50$), large domains form and long-range correlations are visible. At high temperature ($T = 3.50$), the system becomes disordered with no dominant orientation.

At $T = 1.50$, the system is deep in the ordered phase and displays a nearly uniform spin alignment. At $T = 2.50$, large domains of mixed alignment appear, signaling critical

fluctuations. At $T = 3.50$, disorder dominates and no long-range correlations remain. These qualitative patterns reflect the system's transition from ferromagnetic to paramagnetic behavior.

4.2 Thermodynamic Observables

Figure 2 shows the temperature dependence of four key observables: energy per spin, magnetization, heat capacity, and susceptibility.

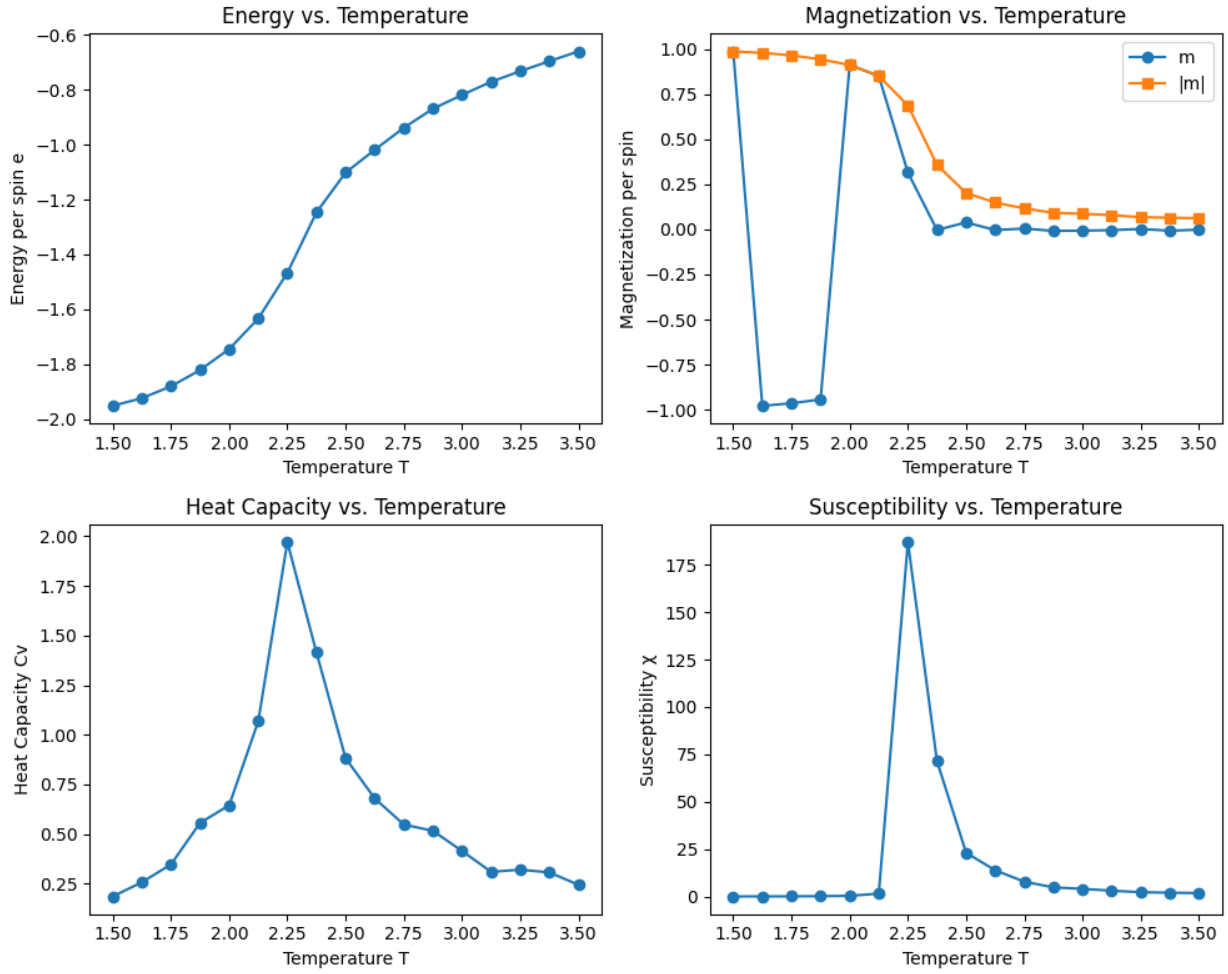

Figure 2: Thermodynamic observables of the 2D Ising model obtained from Monte Carlo simulation. Energy increases with temperature as disorder grows. Magnetization drops sharply near the critical temperature. Heat capacity and susceptibility show prominent peaks at the phase transition, reflecting large fluctuations.

4.2.1 Energy per Spin

Energy increases monotonically with temperature, moving from a highly ordered, low-energy state to a disordered, higher-energy state. The steep change near $T \approx 2.25$ reflects the rapid

breakdown of spin alignment at the transition.

4.2.2 Magnetization

The magnetization curve provides the clearest evidence of the phase transition. Below $T \approx 2.0$, spins remain aligned with nearly full magnetization. As the system approaches the critical temperature, magnetization sharply collapses to zero. The absolute magnetization $|m|$ reveals the formation of domains even when the net magnetization cancels out.

4.2.3 Heat Capacity

The heat capacity exhibits a pronounced peak near $T_c \approx 2.27$, indicating large energy fluctuations usually seen with a continuous phase transition. In the infinite-lattice limit, this quantity diverges; in finite simulations, the peak remains large but finite.

4.2.4 Susceptibility

The susceptibility also exhibits a sharp peak at the critical point, reflecting divergent spin–spin correlations. The height and sharpness of the peak are consistent with theoretical expectations for a finite lattice.

4.3 Evidence for the Phase Transition

Together, the observables confirm the known ferromagnetic–paramagnetic phase transition of the 2D Ising model. The simulation reproduces:

- spontaneous magnetization at low temperature
- critical fluctuations and large domains near T_c
- loss of long-range order at high temperature
- peaks in response functions (C_V and χ) at the transition

These results match theoretical predictions and Onsager's analytical value for the critical temperature.

4.4 Discussion

The results demonstrate the power of Monte Carlo simulations in reproducing key features of phase transitions with statistical mechanics. Although the Ising model is highly simplified (spins are restricted to ± 1, and interactions are limited to nearest neighbors), it is still able to successfully capture the essential physics of real ferromagnets: ordered phases, thermal disordering, criticality, and divergent fluctuations.

Several factors influence the precision of the simulation. First, the lattice size ($L = 32$) is large enough to exhibit sharp features but small enough to avoid excessive computation time. Finite-size effects smooth out the heat-capacity and susceptibility peaks, preventing

true divergence. Increasing the system size would sharpen these peaks and yield values closer to theoretical asymptotics.

Second, the Metropolis algorithm introduces autocorrelation, especially near criticality. This is why many measurement sweeps and an equilibration period were used. Even without multiple independent simulation trials, the results were stable because of the large number of recorded samples.

Finally, the temperature sweep demonstrates how macroscopic phenomena can emerge from microscopic rules. The progression from ordered to mixed to disordered configurations, the collapse of magnetization, and the peaks in C_V and χ confirm the promise that was made in the introduction: this experiment reveals how collective behavior and phase transitions arise naturally in the 2D Ising model.

Overall, the simulation successfully reproduces the expected physics of the Ising phase transition, validating the algorithm and the theoretical framework.

5 Conclusion

This project used the Metropolis Monte Carlo algorithm to investigate the thermodynamic behavior of the two-dimensional Ising model and to observe the emergence and breakdown of magnetic order as temperature varies. By simulating a square lattice of spins over a range of temperatures, the experiment successfully reproduced the defining features of the ferromagnetic–paramagnetic phase transition. The progression from nearly uniform spin alignment at low temperature, to correlated domains near the critical temperature, to complete disorder at high temperature, demonstrates how simple microscopic interactions give rise to complex macroscopic behavior.

Quantitative measurements of energy, magnetization, heat capacity, and magnetic susceptibility further reinforced this picture. The sharp drop in magnetization, the pronounced peaks in C_V and χ, and the smooth rise of energy with temperature all closely matched theoretical predictions and appeared at temperatures consistent with Onsager's exact critical temperature. These results highlight how this numerical method, even with a finite lattice and simplified spin dynamics, can accurately capture the essential physics of real ferromagnetic systems.

The success of this simulation portrays the power of Monte Carlo techniques in studying systems with large configuration spaces that are analytically intractable. While finite-size effects and autocorrelation limit the sharpness of observed divergences, the simulation still provides clear, interpretable signatures of criticality. Future improvements could involve larger lattices or the inclusion of external fields to explore symmetry breaking in greater depth.

Overall, this experiment demonstrates that despite its simplicity, the Ising model serves as an invaluable platform for understanding phase transitions, emergent behavior, and the broader principles of statistical mechanics. The results affirm this study's promise: that complex collective phenomena can be reproduced, visualized, and understood through computational simulations grounded in fundamental physical laws.

References

[1] W. Zhu. Notes on the Ising Model. https://quantum-many-body-theory.lab.westlake.edu.cn/files/Stat_Ising_model.pdf, 2020. Lecture notes.

[2] Samik Raychaudhuri. Introduction to Monte Carlo Simulation. https://www.informs-sim.org/wsc08papers/012.pdf, 2010. Online PDF.

[3] Unknown. The Ising Model: Lecture Notes. https://ps.uci.edu/~cyu/p238C/LectureNotes/IsingModel/IsingModel.pdf, 2013. Based on Huang and Thompson.

A Python Code Used in the Simulation

A.1 Main Simulation Script

```
import numpy as np
import matplotlib.pyplot as plt

# ============================================================
# 1. LATTICE INITIALIZATION
# ============================================================
def initialize_lattice(L, init_type="random", rng=None):
    """
    Creates an L x L array of spins (each spin is +1 or -1).

    init_type:
        "random" -> each spin is chosen randomly with equal chance.
        "up"     -> all spins +1 (fully magnetized).
        "down"   -> all spins -1.
    """
    if rng is None:
        rng = np.random.default_rng()

    if init_type == "random":
        # Randomly assign +1 or -1 to each lattice site
        spins = rng.choice([-1, 1], size=(L, L))
    elif init_type == "up":
        spins = np.ones((L, L), dtype=int)
    elif init_type == "down":
        spins = -np.ones((L, L), dtype=int)
    else:
        raise ValueError("init_type must be 'random', 'up', or 'down
            '.")
    return spins


# ============================================================
# 2. ENERGY CALCULATION USING ISING HAMILTONIAN
# ============================================================
def compute_total_energy(spins, J=1.0, h=0.0):
    """
    Computes the total energy of the system using the Ising
       Hamiltonian:

        H = -J sum(s_i * s_neighbor) - h sum(s_i)

    Only RIGHT and DOWN neighbors are used to avoid double counting.
    Periodic boundary conditions are applied via modulo indexing.
```

```
    """
    L = spins.shape[0]
    E = 0.0

    for i in range(L):
        for j in range(L):

            S = spins[i, j]

            # RIGHT neighbor (wrap around using modulo)
            S_right = spins[i, (j + 1) % L]

            # DOWN neighbor
            S_down = spins[(i + 1) % L, j]

            # Add the pair interactions
            E -= J * S * (S_right + S_down)

    # Add contribution from external field term
    M = np.sum(spins)
    E -= h * M

    return E


# ============================================================
# 3. METROPOLIS SINGLE SPIN FLIP
# ============================================================
def metropolis_single_flip(spins, beta, J=1.0, h=0.0, rng=None):
    """
    Attempts one spin flip using the Metropolis acceptance rule.

    Steps:
        1. Pick random site (i, j).
        2. Calculate dE = energy change if s_ij -> -s_ij.
        3. If dE <= 0: accept (energy decreases).
        4. If dE > 0: accept with probability exp(-beta * dE).
        5. If accepted, flip the spin. Otherwise do nothing.
    """
    if rng is None:
        rng = np.random.default_rng()

    L = spins.shape[0]

    # Pick random spin
    i = rng.integers(0, L)
    j = rng.integers(0, L)
```

```
    s = spins[i, j]

    # Identify the 4 nearest neighbors (with periodic boundary
       conditions)
    up    = spins[(i - 1) % L, j]
    down  = spins[(i + 1) % L, j]
    left  = spins[i, (j - 1) % L]
    right = spins[i, (j + 1) % L]

    neighbor_sum = up + down + left + right

    # Formula for the energy change of flipping a single spin:
    # dE = 2 * J * s * sum(neighbors) + 2 * h * s
    dE = 2.0 * J * s * neighbor_sum + 2.0 * h * s

    # Metropolis acceptance rule
    if dE <= 0:
        # Flip is favorable -> always accept
        spins[i, j] = -s
        return dE, True
    else:
        # Flip is unfavorable -> accept probabilistically
        if rng.random() < np.exp(-beta * dE):
            spins[i, j] = -s
            return dE, True
        else:
            return 0.0, False


# ============================================================
# 4. METROPOLIS SWEEP = L^2 SPIN-FLIP ATTEMPTS
# ============================================================
def metropolis_sweep(spins, beta, J=1.0, h=0.0, rng=None):
    """
    Performs one Monte Carlo sweep.
    A sweep = one attempted flip per spin (L^2 total attempts).
    """
    if rng is None:
        rng = np.random.default_rng()

    L = spins.shape[0]
    N = L * L

    total_dE = 0.0
    n_accept = 0

    # Try flipping each spin once (on average)
```

```
    for _ in range(N):
        dE, accepted = metropolis_single_flip(spins, beta, J=J, h=h,
            rng=rng)
        total_dE += dE
        if accepted:
            n_accept += 1

    return total_dE, n_accept


# ============================================================
# 5-7. RUN SIMULATION AT ONE TEMPERATURE: EQUILIBRATION +
   MEASUREMENT
# ============================================================
def run_ising_at_temperature(L,
                             T,
                             n_eq_sweeps=500,
                             n_meas_sweeps=1000,
                             J=1.0,
                             h=0.0,
                             init_type="random",
                             measure_interval=1,
                             rng=None):
    """
    Runs the Ising simulation at a fixed temperature T.

    Process:
        (A) Initialize lattice.
        (B) Equilibrate -> discard these sweeps.
        (C) Perform measurement sweeps -> record E, M, E^2, M^2, |M
           |.
        (D) Compute averages: e, m, Cv, Chi.
    """
    if rng is None:
        rng = np.random.default_rng()

    beta = 1.0 / T
    N = L * L

    # Step A: Initialize lattice
    spins = initialize_lattice(L, init_type=init_type, rng=rng)

    # Step B: Equilibration (no measurements)
    for _ in range(n_eq_sweeps):
        metropolis_sweep(spins, beta, J=J, h=h, rng=rng)

    # Variables to accumulate measurements
```

```
    E_sum = 0.0
    E2_sum = 0.0
    M_sum = 0.0
    M2_sum = 0.0
    M_abs_sum = 0.0
    n_meas = 0

    # Step C: Measurement sweeps
    for sweep in range(n_meas_sweeps):
        metropolis_sweep(spins, beta, J=J, h=h, rng=rng)

        # Only record every "measure_interval" sweeps
        if sweep % measure_interval == 0:

            # Compute energy and magnetization of current lattice
            E = compute_total_energy(spins, J=J, h=h)
            M = np.sum(spins)

            E_sum += E
            E2_sum += E * E
            M_sum += M
            M2_sum += M * M
            M_abs_sum += abs(M)

            n_meas += 1

    # Step D: Compute averages
    E_mean = E_sum / n_meas
    E2_mean = E2_sum / n_meas
    M_mean = M_sum / n_meas
    M2_mean = M2_sum / n_meas
    M_abs_mean = M_abs_sum / n_meas

    # Convert to per-spin quantities
    e_mean = E_mean / N
    m_mean = M_mean / N
    m_abs_mean = M_abs_mean / N

    # Heat capacity per spin (fluctuation formula)
    Cv = (E2_mean - E_mean**2) / (N * T**2)

    # Susceptibility per spin
    Chi = (M2_mean - M_mean**2) / (N * T)

    return {
        "T": T,
        "E_mean": E_mean,
```

```
        "E2_mean": E2_mean,
        "M_mean": M_mean,
        "M2_mean": M2_mean,
        "M_abs_mean": M_abs_mean,
        "e_mean": e_mean,
        "m_mean": m_mean,
        "m_abs_mean": m_abs_mean,
        "Cv": Cv,
        "Chi": Chi,
        "final_spins": spins.copy()  # store final configuration
    }


# ============================================================
# 8-9. TEMPERATURE SWEEP
# ============================================================
def temperature_sweep(L,
                      T_list,
                      n_eq_sweeps=500,
                      n_meas_sweeps=1000,
                      J=1.0,
                      h=0.0,
                      init_type="random",
                      measure_interval=1,
                      rng=None):
    """
    Runs the Ising simulation for every temperature in T_list.

    Returns a dictionary of observables vs temperature,
    which can be plotted or saved.
    """
    if rng is None:
        rng = np.random.default_rng()

    # Lists to accumulate results
    T_values = []
    e_values = []
    m_values = []
    m_abs_values = []
    Cv_values = []
    Chi_values = []
    final_configs = []

    for T in T_list:
        print(f"Running T = {T:.3f} ...")

        res = run_ising_at_temperature(
```

```
            L=L,
            T=T,
            n_eq_sweeps=n_eq_sweeps,
            n_meas_sweeps=n_meas_sweeps,
            J=J,
            h=h,
            init_type=init_type,
            measure_interval=measure_interval,
            rng=rng
        )

        # Append results from this temperature
        T_values.append(res["T"])
        e_values.append(res["e_mean"])
        m_values.append(res["m_mean"])
        m_abs_values.append(res["m_abs_mean"])
        Cv_values.append(res["Cv"])
        Chi_values.append(res["Chi"])
        final_configs.append(res["final_spins"])

    # Return everything as a structured dictionary
    return {
        "T": np.array(T_values),
        "e_mean": np.array(e_values),
        "m_mean": np.array(m_values),
        "m_abs_mean": np.array(m_abs_values),
        "Cv": np.array(Cv_values),
        "Chi": np.array(Chi_values),
        "final_configs": final_configs
    }


# ============================================================
# 10. PLOTTING OBSERVABLES
# ============================================================
def plot_observables(data):
    """
    Creates 4 plots:
        - Energy per spin vs T
        - Magnetization vs T
        - Heat capacity vs T
        - Susceptibility vs T
    """
    T = data["T"]
    e = data["e_mean"]
    m = data["m_mean"]
    m_abs = data["m_abs_mean"]
```

```
    Cv = data["Cv"]
    Chi = data["Chi"]

    fig, axes = plt.subplots(2, 2, figsize=(10, 8))
    ax1, ax2, ax3, ax4 = axes.flatten()

    ax1.plot(T, e, marker='o')
    ax1.set_xlabel("Temperature T")
    ax1.set_ylabel("Energy per spin e")
    ax1.set_title("Energy vs. Temperature")

    ax2.plot(T, m, marker='o', label="m")
    ax2.plot(T, m_abs, marker='s', label="|m|")
    ax2.set_xlabel("Temperature T")
    ax2.set_ylabel("Magnetization per spin")
    ax2.set_title("Magnetization vs. Temperature")
    ax2.legend()

    ax3.plot(T, Cv, marker='o')
    ax3.set_xlabel("Temperature T")
    ax3.set_ylabel("Heat Capacity Cv")
    ax3.set_title("Heat Capacity vs. Temperature")

    ax4.plot(T, Chi, marker='o')
    ax4.set_xlabel("Temperature T")
    ax4.set_ylabel("Susceptibility Chi")
    ax4.set_title("Susceptibility vs. Temperature")

    plt.tight_layout()
    plt.show()


# ============================================================
# 11. VISUALIZE SPIN CONFIGURATION
# ============================================================
def visualize_spins(spins, ax=None, title=None):
    """
    Displays a 2D spin configuration using a red/blue colormap.
    +1 -> red
    -1 -> blue

    If ax is provided, draw into that axes (for subplots) and
    DO NOT call plt.show() here. If ax is None, create a new
    figure and show it immediately.
    """
    own_fig = False   # track whether we created the figure
```

```
    if ax is None:
        fig, ax = plt.subplots()
        own_fig = True

    ax.imshow(spins, cmap="bwr", interpolation="nearest")
    ax.set_xticks([])
    ax.set_yticks([])
    if title:
        ax.set_title(title)

    # Only show if we created our own figure
    if own_fig:
        plt.show()


# ============================================================
# MAIN EXAMPLE (RUN WHEN FILE IS EXECUTED DIRECTLY)
# ============================================================
if __name__ == "__main__":
    rng = np.random.default_rng(seed=42)

    # ---------- CHOOSE ONE MODE BY COMMENTING/UNCOMMENTING
       ----------
    # more sweeps to reach equilibrium
    # more measurements -> smoother statistics
    # measure every 10 sweeps

    # FAST MODE: quick runs, good for debugging & sanity checks
    #L = 20
    #T_list = np.linspace(1.5, 3.5, 7)  # fewer temperatures
    #n_eq_sweeps = 500
    #n_meas_sweeps = 2000
    #measure_interval = 5

    # MEDIUM MODE: nicer curves, still reasonable runtime
    L = 32
    T_list = np.linspace(1.5, 3.5, 17)  # focus around Tc ~ 2.27
    n_eq_sweeps = 2000
    n_meas_sweeps = 5000
    measure_interval = 10

    # ------------------------------------------------------------

    data = temperature_sweep(
        L=L,
        T_list=T_list,
        n_eq_sweeps=n_eq_sweeps,
```

```
        n_meas_sweeps=n_meas_sweeps,
        init_type="random",
        measure_interval=measure_interval,
        rng=rng
    )

    # Plot observables
    plot_observables(data)

    # Visualize sample configurations (low, mid, high T)
    configs = data["final_configs"]

    fig, axes = plt.subplots(1, 3, figsize=(12, 4))
    visualize_spins(configs[0], ax=axes[0], title=f"T = {T_list
       [0]:.2f}")
    visualize_spins(configs[len(T_list)//2], ax=axes[1], title=f"T =
        {T_list[len(T_list)//2]:.2f}")
    visualize_spins(configs[-1], ax=axes[2], title=f"T = {T_list
       [-1]:.2f}")

    plt.tight_layout()
    plt.show()
```