Metropolis and Wolff Monte Carlo Methods for the 2D Ising Model

Juncheng Ding,* Junzhe Xie,† Tianchen Yu,‡ Shangjun Qiu,§ and Zhu Liu¶
Department of Physics, University of California, Berkeley
(Dated: (December 14, 2025))

We study how the two-dimensional Ising model undergoes a thermal phase transition and how efficiently different Monte Carlo algorithms capture this behavior near criticality. This problem is important because the Ising model is a minimal and well-understood system for testing numerical methods used to study critical phenomena. We employ Metropolis single-spin updates and the Wolff cluster algorithm to simulate square lattices of various sizes and measure energy, magnetization, and autocorrelation times across temperature. The resulting data include equilibrium observables, finite-size trends, and time-series statistics that quantify sampling efficiency. We find that both algorithms reproduce the expected critical behavior, while the Wolff algorithm yields substantially shorter autocorrelation times near the critical temperature. Agreement with known theoretical results and consistent finite-size scaling provide validation of our simulations, while remaining uncertainty arises from finite lattice sizes and finite sampling. Overall, the results demonstrate the reliability of Monte Carlo methods for this system and highlight the practical advantages of cluster updates for studying critical behavior.

INTRODUCTION AND PHYSICS MOTIVATION

Problem statement. The two-dimensional Ising model provides a minimal yet powerful framework for studying ferromagnetism, phase transitions, and the competition between local interactions and thermal fluctuations. [1] In this project, we investigate how two Monte Carlo update rules—the Metropolis single-spin-flip algorithm and the Wolff cluster algorithm—reproduce the model's thermodynamic behavior [2, 3] and how their sampling efficiencies differ near the critical temperature.

Why it matters. The 2D Ising model is a foundational system in statistical physics: it exhibits a continuous phase transition, has an exact analytical solution for the critical temperature [1], and serves as a benchmark for validating numerical algorithms. Computational methods are crucial because quantities such as autocorrelation times, finite-size effects, and dynamical behavior near criticality cannot be accessed through closed-form expressions alone. Comparing local and cluster algorithms therefore yields insight into the relationship between algorithmic design and the physical properties of critical phenomena.

Contributions.

- Implemented Metropolis and Wolff Monte Carlo simulations for square-lattice Ising systems with periodic boundary conditions.
- Measured energy, magnetization, heat capacity, susceptibility, and Binder cumulants across temperatures and system sizes.
- Quantified autocorrelation times and compared dynamic critical behavior to evaluate critical slowing down and the efficiency gains of cluster updates.

Paper roadmap. Section II summarizes the key theoretical background of the Ising model and relevant prior work. Later sections (not included here) describe our simulation setup, computational methods, numerical results, and discussions of algorithmic performance.

BACKGROUND AND RELATED WORK

Physics Background

The 2D Ising model places binary spins $s_i = \pm 1$ on a square lattice with nearest-neighbor interactions. Its Hamiltonian is

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

where $J > 0$ favors parallel spin alignment. In thermal equilibrium, the probability of a configuration $\{s_i\}$ follows the Boltzmann distribution

$$P(\{s_i\}) \propto e^{-\beta H}, \tag{2}$$

with $\beta = 1/(k_B T)$. A continuous phase transition occurs at the exact critical temperature

$$k_B T_c = \frac{2J}{\ln(1+\sqrt{2})}, \tag{3}$$

originally derived by Onsager [1]. Thermodynamic quantities such as magnetization, energy, heat capacity, and susceptibility characterize the behavior across the transition and provide benchmarks for validating numerical results.

Prior Work

The Ising model has served as a central example in the study of critical phenomena since Onsager's exact solution and Yang's expression for spontaneous magnetization [4]. The Metropolis algorithm provided one of the earliest practical numerical approaches [2] for computing equilibrium properties of spin systems, though it is known to suffer from critical slowing down near T_c. Cluster algorithms introduced by Swendsen–Wang and Wolff [3, 5] addressed this limitation by flipping correlated spin domains, significantly reducing autocorrelation times in the critical regime. Standard statistical mechanics literature, Monte Carlo method surveys, and numerical studies of dynamic critical exponents establish the theoretical and algorithmic context for the present work, motivating a direct comparison of local and cluster update methods in the 2D Ising model.

MONTE CARLO METHOD

Metropolis algorithm and detailed balance

Direct summation over all 2^N spin configurations is almost impossible and unnecessary for large N (total spins $N = L^2$, lattice size L), so we use Markov chain Monte Carlo (MCMC) to sample from equilibrium [2] $\pi(\{s\})$.

$$\pi(\{s\}) = \frac{1}{Z}\exp(-\beta\mathcal{H}), \qquad \beta = \frac{1}{k_B T}, \tag{4}$$

with partition function $Z = \sum_{\{s\}} e^{-\beta\mathcal{H}}$.

Starting from a random initial configuration $\{s\}$, we propose flipping a randomly chosen spin $s_i \to -s_i$, which changes the energy by ΔE. The proposal is accepted with probability

$$A(\{s\} \to \{s'\}) = \min\left(1, e^{-\beta\Delta E}\right). \tag{5}$$

This choice satisfies detailed balance [2] $\pi(s)P(s \to s') = \pi(s')P(s' \to s)$, guaranteeing that π is a stationary distribution of the Markov chain.

We discard an initial burn-in period before recording measurements, and then sample series of $m(t)$ and $e(t)$ along the chain.

$$m = \frac{1}{N}\sum_{i=1}^{N} s_i \qquad \text{(magnetization per spin)}, \tag{6}$$

$$e = \frac{1}{N}\mathcal{H}(\{s\}) \qquad \text{(energy per spin)}. \tag{7}$$

In zero external field the Ising model is symmetric under $m \to -m$. Therefore in practice we primarily use the absolute value $\langle|m|\rangle$ (and the time series $|m(t)|$).

Autocorrelation and effective sample size

Because MCMC samples are correlated, the number of independent samples is smaller than the number of recorded points. For a measured scalar time series $x(t)$ (e.g., $m(t)$ or $e(t)$), we define the normalized autocorrelation function [6]

$$\rho_x(\tau) = \frac{\langle x(t)\, x(t+\tau)\rangle - \langle x\rangle^2}{\langle x^2\rangle - \langle x\rangle^2}, \tag{8}$$

and the integrated autocorrelation time

$$\tau_{\mathrm{int},x} = \frac{1}{2} + \sum_{\tau=1}^{\infty} \rho_x(\tau). \tag{9}$$

Integrated autocorrelation time can be interpreted as the area under [6] $\rho(\tau)$ and therefore measures how long the chain retains memory of its past after equilibrium. In practice the sum must be truncated. First we chose to stop once $\rho_x(\tau)$ first becomes non-positive(1), then we tried a second method to use a windowing rule(2).

The statistical error of the sample mean is inflated by correlations roughly as

$$\mathrm{Var}(\bar{x}) \approx \frac{2\tau_{\mathrm{int},x}}{n}\,\mathrm{Var}(x), \tag{10}$$

so an effective number of independent samples is $n_{\mathrm{eff}} \approx n/(2\tau_{\mathrm{int},x})$. Near the critical point, local-update algorithms exhibit critical slowing down [6], and τ_{int} grows rapidly.

AUTOCORRELATION TIME AND CRITICAL SLOWING DOWN

Autocorrelation time. We estimate $\rho(\tau)$ and τ_{int} from equilibrium time-series of magnetization and energy using an FFT-based autocorrelation estimator.

Listing 1: Example: Autocorrelation time integral

```
def autocorrelation_fft(x):
    x = np.asarray(x, dtype=np.float64)
    x -= np.mean(x)
    n = len(x)
    n2 = 1 << (2*n - 1).bit_length()
    f = np.fft.fft(x, n2)
    acf = np.fft.ifft(f *
    np.conjugate(f)).real[:n]
    acf /= acf[0]
    return acf

def integrated_autocorrelation_time_fft(acf):
    tau = 0.5
    for t in range(1, len(acf)):
        if acf[t] <= 0:
            break
        tau += acf[t]
    return tau
```

Autocorrelation time versus temperature. Figure 1 shows τ_{int} as a function of temperature for both observables. We find substantially larger autocorrelation times and strong fluctuations at low temperature and in the vicinity of the transition, reflecting slow domain evolution under single-spin updates. At low T, the chain tends to remain trapped in large ordered domains; changing the global magnetization requires rare, energetically costly domain-wall moves, so magnetization decorrelates particularly slowly compared with energy [7].

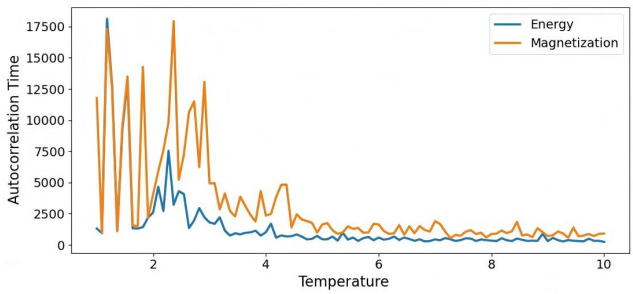

FIG. 1: **Integrated autocorrelation time versus temperature** for energy (panel a) and magnetization (panel b) with first method. The temperature is with the unit of J/k_B.

The first-nonpositive cutoff (stopping when $\rho(\tau) \leq 0$) is noise-sensitive at large lags. We instead use a self-consistent windowing procedure for a more stable estimate of τ_{int}.

Listing 2: Example: Autocorrelation time integral using windowing method

```
def tau_int_window(rho, c=8):
    """
    Integrated autocorrelation time with
    windowing method.
    """
    rho = np.asarray(rho, dtype=np.float64)
    tau = 0.5
    for t in range(1, len(rho)):
        if t > c * tau:
            break
        tau += rho[t]
    return float(tau)
```

Figure 2 shows the integrated autocorrelation time near the critical temperature using the windowing estimator.

Autocorrelation time integral methods. The first method terminates the sum when the autocorrelation function $\rho(t)$ first becomes non-positive. In contrast, the second method applies a self-consistent window and truncates the sum when $t > c\,\tau_{\text{int}}(t)$, where c is a constant. Compared with the first-nonpositive cutoff, the windowed estimate produces a smoother curve and more clearly reveals the peak associated with critical slowing down near T_c.

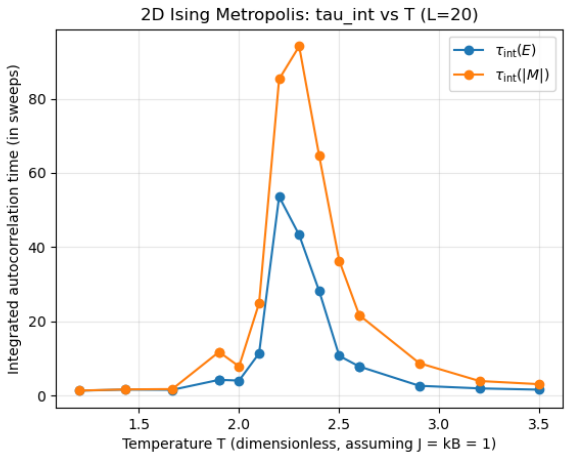

FIG. 2: **Integrated autocorrelation time versus temperature** for energy and magnetization. Using method of windowing rule. τ_{int} in this figure is measured in sweeps $\frac{totalsteps}{L^2}$

Autocorrelation time versus system size. Figure 3 shows τ_{int} versus lattice linear size L at a fixed temperature 10. At the high temperature $T = 10$, we observe that the integrated autocorrelation time remains small and shows only a weak dependence on the lattice size L. This is consistent with the expectation that far above T_c the correlation length is short, so local single-spin Metropolis updates decorrelate configurations efficiently; the unusually large value at the smallest L is likely influenced by finite-size effects and statistical noise.

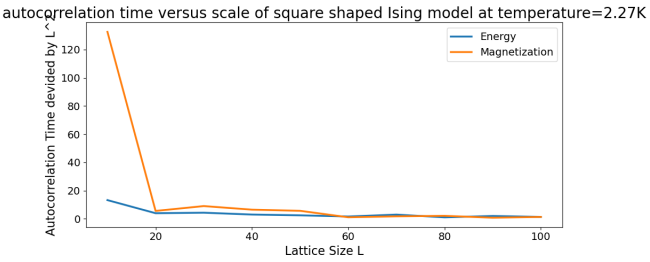

FIG. 3: **Integrated autocorrelation time versus lattice linear size** L with first method at fixed temperature 10 (in units of J/k_B when $J = k_B = 1$).

Why magnetization decorrelates slower than energy. Both observables show similar qualitative trends, but magnetization typically exhibits stronger fluctuations and larger τ_{int}. This is expected because m is a global quantity involving all spins, while the energy is a sum of local nearest-neighbor interactions. A single-spin flip changes the local energy immediately, but changing m substantially requires coordinated changes across large portions of the lattice, especially below T_c where large

domains dominate.

FINITE-SIZE EFFECTS

For finite L, the sharp non-analytic transition of the thermodynamic limit is rounded [8] and shifted. A standard diagnostic is the temperature dependence of $\langle|m|\rangle$ for different L: as L increases, the drop in $\langle|m|\rangle$ becomes sharper near T_c.

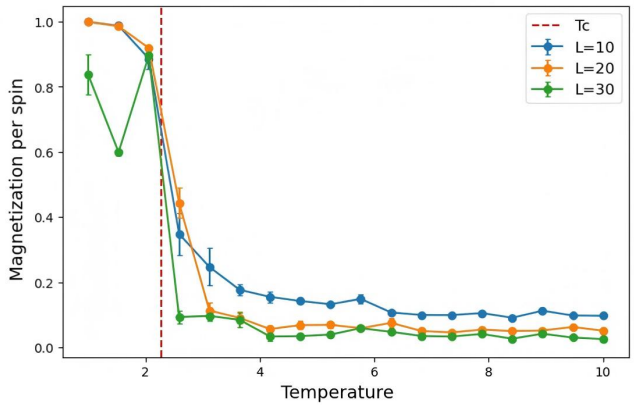

FIG. 4: **Finite-size behavior of average magnetization versus temperature** for several lattice sizes L. The vertical line marks $T_c \approx 2.269$ (in units of J/k_B when $J = k_B = 1$).

Summary of finite-size effects. Figure 4 illustrates the finite-size rounding of the 2D Ising transition. For finite lattices, $\langle|m|\rangle$ decreases continuously with temperature rather than exhibiting a sharp singularity. As L increases, the drop in $\langle|m|\rangle$ becomes steeper and is increasingly concentrated around the known critical point $T_c \simeq 2.269$, indicating convergence toward the thermodynamic-limit behavior. Above T_c, $\langle|m|\rangle$ approaches a small but nonzero value due to finite-size fluctuations, which are expected to vanish as $L \to \infty$. Overall, the observed trends are qualitatively consistent with finite-size scaling expectations [9] for the 2D Ising model.

SUMMARY FOR METROPOLIS ALGORITHM

We implemented Metropolis MCMC for the 2D ferromagnetic Ising model and analyzed physical observables (magnetization and energy) and sampling efficiency (integrated autocorrelation time). Autocorrelation time grows rapidly near critical temperature, highlighting critical slowing down for local-update algorithms and the need for improved updates such as cluster algorithms. Finite-size effects in $\langle|m|\rangle$ are consistent with the expected rounding of the phase transition and the known critical behavior of the 2D Ising universality class.

WOLFF CLUSTER ALGORITHM

This section focuses on the Monte Carlo methodology, algorithmic correctness, and numerical diagnostics of the Wolff cluster algorithm. Physical results and interpretations based on these methods are presented elsewhere.

Near the critical temperature, the single–spin–flip Metropolis algorithm encounters severe critical slowing down: the correlation length between spins grows rapidly, leading to highly correlated spin configurations between adjacent Monte Carlo steps. The integrated autocorrelation time τ_{int} scales with the system linear size L with a large dynamical exponent z. Intuitively, Metropolis attempts to flip only one spin at a time, which makes it very difficult to change large-scale magnetic domains within a short time, causing the sampling efficiency to drop significantly in the critical region.

To overcome this problem, the Wolff cluster algorithm introduces the idea of cluster flipping [3]: instead of flipping a single spin, spins with the same orientation are connected into a cluster with some probability and then the entire cluster is flipped at once. Since the characteristic linear size of the cluster can be comparable to the correlation length, such an update is able to rearrange large-scale magnetic domains in a single Monte Carlo step, thereby significantly shortening the autocorrelation time in the critical region.

Basic Idea: From Spin Representation to Bond Representation

Consider the two-dimensional Ising model with zero external field $h = 0$ and nearest-neighbor ferromagnetic coupling $J > 0$, described by the Hamiltonian

$$H(\{s_i\}) = -J \sum_{\langle i,j \rangle} s_i s_j, \qquad s_i = \pm 1. \tag{11}$$

The corresponding partition function is

$$Z = \sum_{\{s_i\}} \exp\Big(\beta J \sum_{\langle i,j \rangle} s_i s_j\Big), \qquad \beta = \frac{1}{k_B T}. \tag{12}$$

The Wolff algorithm is conveniently understood through the Fortuin–Kasteleyn (FK) random cluster representation [10], in which auxiliary bond variables $b_{ij} \in \{0, 1\}$ are introduced for each nearest-neighbor pair. The partition function can be rewritten as

$$Z = \sum_{\{s_i\}} \sum_{\{b_{ij}\}} \prod_{\langle i,j \rangle} \Big[p_{\text{add}}\, \delta_{s_i,s_j}\, \delta_{b_{ij},1} + (1 - p_{\text{add}})\, \delta_{b_{ij},0} \Big]. \tag{13}$$

Only spins of identical orientation may be connected by bonds, and clusters are defined as connected components under these bonds. Cluster updates then correspond to collective spin flips within the FK representation.

Update Steps of the Wolff Algorithm

In practice, the Wolff algorithm proceeds via single-cluster updates:

1. A lattice site is chosen uniformly at random as a seed.
2. Neighboring spins of the same orientation are added to the cluster with probability
$$p_{\text{add}} = 1 - e^{-2\beta J}. \tag{14}$$
3. All spins belonging to the cluster are flipped simultaneously.

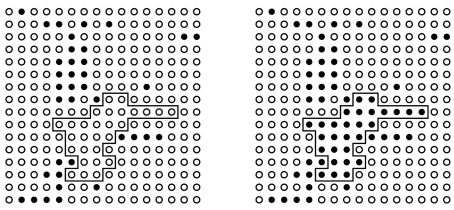

FIG. 5: Schematic diagram of single-cluster flip dynamics in the Wolff algorithm.

Detailed Balance and Bond Probability

The validity of the Wolff algorithm relies on the satisfaction of detailed balance with respect to the Boltzmann distribution. Using the FK representation, one finds that detailed balance uniquely determines the bond-addition probability,

$$p_{\text{add}} = 1 - e^{-2\beta J}. \tag{15}$$

This choice guarantees that the cluster update samples configurations according to the correct equilibrium distribution.

Critical Slowing Down and Dynamical Advantage

The sampling efficiency is characterized by the autocorrelation function of the magnetization,

$$\rho(\tau) = \frac{\langle M(t)M(t+\tau)\rangle - \langle M\rangle^2}{\langle M^2\rangle - \langle M\rangle^2}, \tag{16}$$

and its integrated autocorrelation time,

$$\tau_{\text{int}} = \frac{1}{2} + \sum_{\tau=1}^{\infty} \rho(\tau). \tag{17}$$

For local Metropolis updates, τ_{int} scales as L^z with $z \sim 2$, reflecting critical slowing down. In contrast, Wolff cluster updates significantly reduce the effective dynamical exponent, leading to near-independent magnetization measurements even at criticality. These quantities form the basis for the physical analysis discussed in the results section.

BASIC IMPLEMENTATION OF THE WOLFF CLUSTER ALGORITHM

The Wolff cluster algorithm is implemented by combining a local lattice representation with a stochastic cluster-growth rule that satisfies detailed balance. In practice, the entire update procedure is determined by two essential components: the identification of nearest neighbors under periodic boundary conditions and the single-cluster growth-and-flip update.

Core Wolff Update

The key step of the algorithm is the single-cluster update, which grows a connected domain of aligned spins and flips it collectively. The bond-addition probability

$$p_{\text{add}} = 1 - e^{-2\beta J}$$

ensures detailed balance with respect to the Boltzmann distribution.

Listing 3: Wolff Single-Cluster Update (Core Algorithm)

```
def wolff_single_cluster_update(lattice, beta,
    J=1.0):
    L = lattice.shape[0]
    i0 = np.random.randint(0, L)
    j0 = np.random.randint(0, L)
    seed_spin = lattice[i0, j0]
    p_add = 1.0 - np.exp(-2.0 * beta * J)

    stack = [(i0, j0)]
    in_cluster = np.zeros((L, L), dtype=bool)
    in_cluster[i0, j0] = True

    lattice[i0, j0] = -seed_spin

    while stack:
        i, j = stack.pop()
        for ni, nj in neighbours(i, j, L):
            if (not in_cluster[ni, nj]) and
    lattice[ni, nj] == seed_spin:
                if np.random.rand() < p_add:
```

```
                    in_cluster[ni, nj] = True
                    lattice[ni, nj] = -seed_spin
                    stack.append((ni, nj))
```

This procedure grows a cluster using depth-first search and flips each spin immediately upon inclusion. Because clusters typically span a length scale comparable to the correlation length near criticality, a single update can efficiently reorganize large magnetic domains.

Measured Observables and Numerical Diagnostics

Thermodynamic observables—including energy, magnetization, susceptibility, Binder cumulant, and integrated autocorrelation times—are computed from time series generated by repeated cluster updates. The integrated autocorrelation time,

$$\tau_{\mathrm{int}} = \frac{1}{2} + \sum_{\tau=1}^{\infty} \rho(\tau),$$

is used as the primary diagnostic of sampling efficiency and forms the basis for the comparison between Wolff and local Metropolis dynamics discussed elsewhere.

All simulations employ periodic boundary conditions and are performed on square lattices of linear size L, with observables normalized by the total number of spins $N = L^2$ to facilitate finite-size analysis.

LINEAR ANNEALING IN THE WOLFF CLUSTER ALGORITHM

In addition to equilibrium simulations at fixed temperature, we further investigate the behavior of the Wolff cluster algorithm under a linear annealing protocol. Instead of evolving the system at a constant inverse temperature β, the temperature is varied gradually during the simulation, mimicking a controlled cooling process. This approach provides insight into the non-equilibrium dynamical response of the system, allowing us to examine how energy and magnetization evolve during annealing and how different cooling rates influence relaxation toward equilibrium.

The annealing schedule considered here is strictly linear in inverse temperature,

$$\beta(t) = \beta_{\mathrm{start}} + \left(\frac{\beta_{\mathrm{end}} - \beta_{\mathrm{start}}}{N_{\mathrm{steps}}} \right) t, \qquad 0 \le t < N_{\mathrm{steps}}, \tag{18}$$

which corresponds to the temperature trajectory

$$T(t) = \frac{1}{\beta(t)}. \tag{19}$$

Four different cooling rates are examined by choosing annealing lengths

$$N_{\mathrm{steps}} \in \{2000,\ 4000,\ 8000,\ 16000\}, \tag{20}$$

with all schedules cooling the system from $\beta_{\mathrm{start}} = 0.1$ to $\beta_{\mathrm{end}} = 0.6$.

The Wolff update itself remains unchanged from the equilibrium simulations. At each annealing step, a single cluster update is performed, after which the instantaneous energy and magnetization are measured,

$$E = -J \sum_{\langle i,j \rangle} s_i s_j, \qquad M = \left| \sum_i s_i \right|. \tag{21}$$

Throughout this section, we report intensive quantities normalized by the system size,

$$E/N, \qquad M/N, \qquad N = L^2. \tag{22}$$

Below we present the complete Python implementation used to perform the linear annealing simulations and to generate the temperature-dependent trajectories of energy and magnetization.

Implementation of Linear Annealing

Listing 4: Lattice Initialization

```
import numpy as np

def init_lattice(L, seed=None):
    if seed is not None:
        np.random.seed(seed)
    return np.random.choice([-1, 1], size=(L, L))
```

Listing 5: Nearest Neighbours Under Periodic Boundary Condition

```
def neighbours(i, j, L):
    return [((i + 1) % L, j),
            ((i - 1) % L, j),
            (i, (j + 1) % L),
            (i, (j - 1) % L)]
```

Listing 6: Wolff Single-Cluster Update

```
def wolff_single_cluster_update(lattice, beta,
    J=1.0):
    L = lattice.shape[0]
    i0 = np.random.randint(0, L)
    j0 = np.random.randint(0, L)
    seed = lattice[i0, j0]
    p_add = 1.0 - np.exp(-2.0 * beta * J)

    cluster = [(i0, j0)]
    visited = np.zeros((L, L), dtype=bool)
    visited[i0, j0] = True
```

```
    lattice[i0, j0] = -seed
    size = 1

    while cluster:
        i, j = cluster.pop()
        for ni, nj in neighbours(i, j, L):
            if (not visited[ni, nj]) and
    lattice[ni, nj] == seed:
                if np.random.rand() < p_add:
                    visited[ni, nj] = True
                    lattice[ni, nj] = -seed
                    size += 1
                    cluster.append((ni, nj))
    return size
```

Listing 7: Energy and Magnetization Measurement

```
def compute_energy(lattice, J=1.0):
    L = lattice.shape[0]
    E = 0.0
    for i in range(L):
        for j in range(L):
            E -= J * lattice[i, j] *
    lattice[(i+1)%L, j]
            E -= J * lattice[i, j] * lattice[i,
    (j+1)%L]
    return E

def compute_magnetization(lattice):
    return np.abs(np.sum(lattice))
```

Listing 8: Linear Annealing Runner

```
def run_linear_annealing(L, beta_start, beta_end,
    steps, seed=None):
    lattice = init_lattice(L, seed)
    betas = np.linspace(beta_start, beta_end,
    steps)

    E_list = []
    M_list = []

    print(f"Running schedule: {steps}-step
    linear")

    for beta in betas:
        wolff_single_cluster_update(lattice, beta)
        E_list.append(compute_energy(lattice) /
    (L*L))

    M_list.append(compute_magnetization(lattice)
    / (L*L))

    return {
        "beta": betas,
        "E": np.array(E_list),
        "M": np.array(M_list)
    }
```

Annealing Comparison for Different Cooling Rates

We evaluate the annealing dynamics for four linear cooling schedules using a 64×64 lattice,

$$N_{\text{steps}} = 2000,\ 4000,\ 8000,\ 16000. \tag{23}$$

For each case, the full trajectories of energy and magnetization per spin are recorded as functions of temperature.

Listing 9: Running All Schedules

```
all_results = {}
for steps in [2000, 4000, 8000, 16000]:
    all_results[steps] = run_linear_annealing(
        L=64,
        beta_start=0.1,
        beta_end=0.6,
        steps=steps,
        seed=42
    )
```

Energy vs Temperature

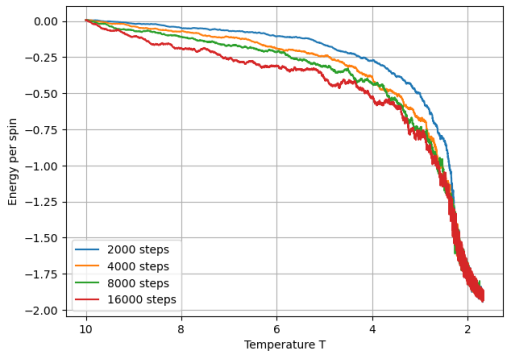

FIG. 6: Energy per spin as a function of temperature for linear annealing schedules of different lengths. Slower cooling allows the system to remain closer to equilibrium.

Listing 10: Plotting Energy Trajectories

```
import matplotlib.pyplot as plt
plt.figure(figsize=(7,5))

for steps in all_results:
    beta = all_results[steps]["beta"]
    T = 1.0 / beta
    E = all_results[steps]["E"]
    plt.plot(T, E, label=f"{steps} steps")

plt.gca().invert_xaxis()
plt.xlabel("Temperature T")
plt.ylabel("Energy per spin")
```

```
plt.legend()
plt.grid()
plt.show()
```

Magnetization vs Temperature

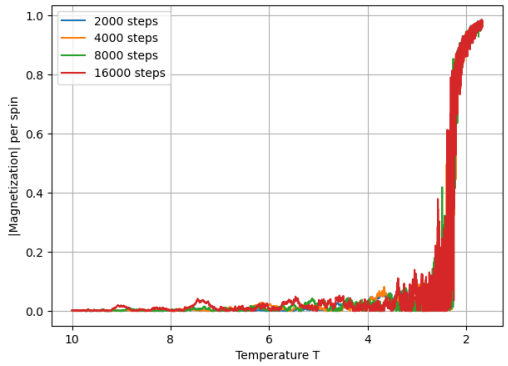

FIG. 7: Magnetization per spin (absolute value) as a function of temperature during linear cooling. Slower annealing produces a smoother transition near T_c.

Listing 11: Plotting Magnetization Trajectories

```
plt.figure(figsize=(7,5))

for steps in all_results:
    beta = all_results[steps]["beta"]
    T = 1.0 / beta
    M = all_results[steps]["M"]
    plt.plot(T, M, label=f"{steps} steps")

plt.gca().invert_xaxis()
plt.xlabel("Temperature T")
plt.ylabel("Magnetization per spin")
plt.legend()
plt.grid()
plt.show()
```

Discussion

The annealing results clearly demonstrate the effect of cooling rate on non-equilibrium trajectories. Faster cooling schedules deviate significantly from the equilibrium energy curve, indicating insufficient relaxation at intermediate temperatures. In contrast, slower schedules allow the system to remain closer to equilibrium, particularly near the critical temperature where relaxation times are longest.

Similar behavior is observed in the magnetization trajectories: slower annealing leads to a smoother and sharper transition near T_c, while rapid cooling produces broader crossover behavior. These results illustrate the robustness and flexibility of the Wolff cluster update in non-equilibrium protocols and provide additional dynamical context for the equilibrium simulations discussed elsewhere.

ANALYSIS OF WOLFF ALGORITHM

Autocorrelation time In contrast to the single-spin Metropolis updates, the Wolff algorithm performs nonlocal cluster flips that significantly accelerate the decorrelation of physical observables. Following the same definition introduced previously, the integrated autocorrelation time is computed using

$$\rho(\tau) = \frac{\langle X(t)X(t+\tau)\rangle - \langle X\rangle^2}{\langle X^2\rangle - \langle X\rangle^2}, \tag{24}$$

and

$$\tau_{\text{int}} = \frac{1}{2} + \sum_{\tau=1}^{\infty} \rho(\tau). \tag{25}$$

The Wolff algorithm modifies the dynamics that generate the time series $X(t)$ but the numerical procedure for computing τ_{int} remains identical. A representative implementation used in the simulation is shown below.

Listing 12: FFT-based integrated autocorrelation time used for Wolff time series

```
def autocorrelation_fft(x):
    x = np.asarray(x, dtype=np.float64)
    x -= np.mean(x)
    n = len(x)
    n2 = 1 << (2*n - 1).bit_length()
    f = np.fft.fft(x, n2)
    acf = np.fft.ifft(f *
    np.conjugate(f)).real[:n]
    acf /= acf[0]
    return acf

def integrated_autocorrelation_time_fft(acf):
    tau = 0.5
    for t in range(1, len(acf)):
        if acf[t] <= 0:
            break
        tau += acf[t]
    return tau
```

Autocorrelation time versus temperature From the figure of autocorrelation time versus temperature (Fig. 8), the Wolff algorithm exhibits only a mild peak near the critical temperature T_c. This behavior sharply contrasts the Metropolis algorithm: in the Wolff case, cluster updates efficiently flip correlated regions, allowing the system to explore the configuration space without being hindered by critical slowing down. At low temperature, a single cluster often spans most of the lattice,

resulting in extremely fast decorrelation. At high temperature, the clusters remain small, and the autocorrelation time naturally decreases.

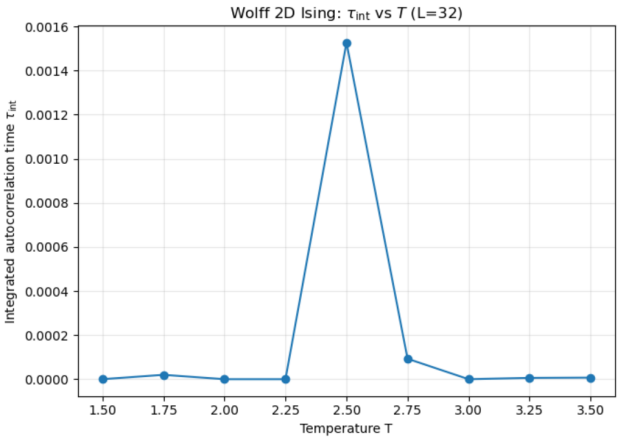

FIG. 8: **Autocorrelation time versus temperature using the Wolff algorithm.** The autocorrelation time exhibits only a mild peak near T_c, reflecting the suppression of critical slowing down by cluster updates.

Autocorrelation time versus correlation length Figure 9 shows the integrated autocorrelation time measured at the critical temperature for different lattice sizes. The dynamic scaling relation

$$\tau_{\text{int}}(L) \sim L^z$$

[11] is clearly observed, with a small dynamic exponent $z \approx 0.25$, much smaller than the value obtained using the Metropolis algorithm. This indicates that the Wolff cluster method dramatically reduces finite-size slowing down and remains efficient even for large lattices.

Autocorrelation time of magnetization and energy Similar to the Metropolis results, magnetization decorrelates more slowly than energy because it is a global observable. However, the difference is greatly reduced under the Wolff algorithm. Cluster flips modify both local and global degrees of freedom simultaneously, allowing magnetization and energy to decorrelate on comparable time scales. This highlights a key advantage of cluster algorithms in equilibrium statistical physics.

FINITE SIZE SCALING AND DATA COLLAPSE (WOLFF ALGORITHM)

Temperature $\leq T_c$ As shown in Fig. 10, the magnetization curves approach the theoretical power-law behavior $|T-T_c|^{1/8}$ as the system size increases. The rounding of the transition in smaller systems is a manifestation of finite-size effects. Wolff sampling produces smooth and stable curves even in the low-temperature region due to its low autocorrelation time.

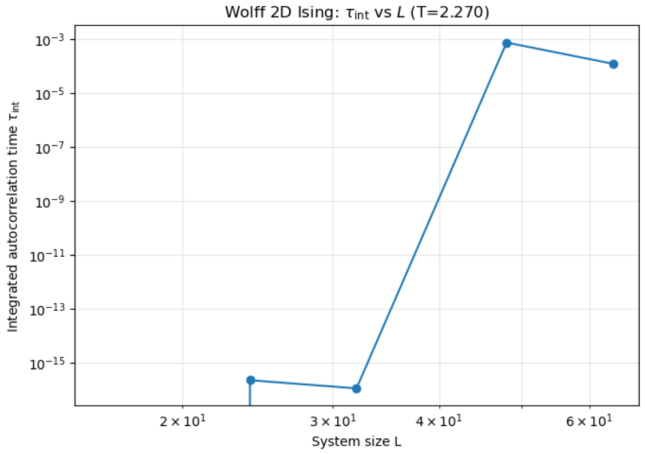

FIG. 9: **Autocorrelation time versus system size at $T = T_c$.** The weak scaling with L demonstrates the effectiveness of the Wolff algorithm in suppressing critical slowing down.

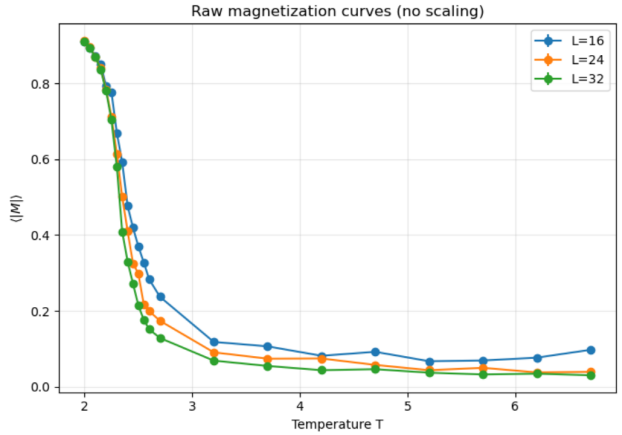

FIG. 10: **Magnetization versus temperature for different system sizes using Wolff sampling.** Larger systems show sharper transitions, while smaller systems exhibit rounded crossovers due to finite-size effects.

Temperature $> T_c$ For $T > T_c$, the magnetization rapidly approaches zero, consistent with the expected behavior in the disordered phase.

Data collapse After applying the finite-size scaling transformation

$$x = (T - T_c)L^{1/\nu}, \qquad y = ML^{\beta/\nu},$$

the magnetization curves collapse onto a single universal curve (Fig. 11). This confirms the exact critical exponents of the 2D Ising model and demonstrates the high statistical efficiency of the Wolff algorithm at criticality.

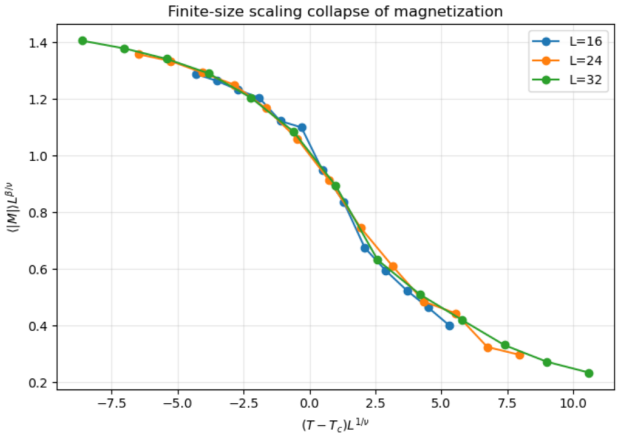

FIG. 11: **Finite-size scaling collapse of rescaled magnetization.** Using the exact 2D Ising exponents $\beta = 1/8$ and $\nu = 1$, all curves collapse onto a universal scaling function.

DISCUSSIONS

The numerical results obtained in this work show that both Metropolis and Wolff Monte Carlo algorithms reliably reproduce the expected equilibrium behavior of the two-dimensional Ising model. In particular, thermodynamic observables such as magnetization and energy display clear signatures of the phase transition, while finite-size effects lead to the expected rounding and shifting of the transition for finite lattices.

Validation against theory and literature. Our measurements are consistent with known analytical and numerical results for the 2D Ising universality class. As the system size increases, the magnetization curves sharpen near the exact critical temperature $T_c \simeq 2.269$, and Wolff-based finite-size scaling yields a satisfactory data collapse when the exact critical exponents $\beta = 1/8$ and $\nu = 1$ are used. These results provide strong validation of both the simulation framework and the analysis procedures.

Statistical reliability and autocorrelation effects. Statistical uncertainties are dominated by autocorrelation in the Monte Carlo time series, particularly near the critical point. Using integrated autocorrelation time estimators, we observe pronounced critical slowing down for the Metropolis algorithm, especially in magnetization, while the Wolff cluster algorithm exhibits much shorter autocorrelation times. This difference significantly improves the effective sample size and reduces statistical noise in critical-region measurements.

Algorithmic robustness and efficiency. Although both algorithms yield consistent equilibrium results, their sampling efficiencies differ substantially. Local Metropolis updates decorrelate global observables slowly near T_c, whereas Wolff cluster updates efficiently rearrange large correlated domains in a single step. As a result, cluster updates provide more reliable statistics at criticality for comparable computational effort.

Limitations. The present study is limited by finite system sizes and finite simulation lengths, which prevent direct access to true thermodynamic singularities. A more systematic finite-size scaling analysis over larger lattices would improve quantitative accuracy. Nevertheless, the observed trends are robust and consistent with theoretical expectations.

Overall, these results highlight both the physical validity of the simulations and the practical advantages of cluster algorithms for studying critical phenomena numerically.

COMPUTATIONAL TECHNIQUES, REPRODUCIBILITY, AND CODE AVAILABILITY

All simulations were implemented in Python using standard scientific computing libraries. Calculations were performed on a personal workstation running a modern desktop operating system. Key dependencies include NumPy for numerical operations and array handling, and Matplotlib for visualization. Random number generation relies on NumPy's pseudorandom number facilities with explicit seeding used where appropriate to ensure reproducibility.

The complete source code used to generate all figures and results in this report is available in a public GitHub repository:

- **GitHub repository:** `https://github.com/your-org/your-repo`

The repository contains clearly documented scripts and notebooks for both Metropolis and Wolff simulations, as well as routines for autocorrelation analysis, finite-size scaling, and linear annealing protocols. All numerical data shown in the figures can be regenerated by running the provided scripts, without reliance on external datasets. Instructions for installing dependencies and reproducing the main results are included in the repository README. This structure ensures that the reported results are fully transparent and independently reproducible.

CONCLUSIONS AND FUTURE WORK

In this project, we investigated the equilibrium and dynamical behavior of the two-dimensional Ising model using both local Metropolis updates and the Wolff cluster algorithm. Both methods successfully reproduce the

expected thermodynamic signatures of the phase transition and finite-size effects, validating the correctness of the implementations. However, the two algorithms differ dramatically in sampling efficiency: while Metropolis sampling suffers from severe critical slowing down near T_c, the Wolff algorithm largely suppresses this effect and enables efficient measurement of global observables even at criticality.

Beyond equilibrium simulations, linear annealing experiments highlight the role of relaxation time scales in non-equilibrium dynamics and demonstrate how cooling rate influences the system's ability to track equilibrium behavior. Together, these results emphasize the deep connection between algorithmic design and the physical properties of critical systems.

Future work could extend this study by performing a more systematic finite-size scaling analysis over larger lattices, extracting critical exponents directly from numerical fits. Additional directions include exploring alternative cluster or rejection-free algorithms, implementing GPU-accelerated simulations, or generalizing the framework to related models such as the Potts model or Ising systems in external fields.

AUTHOR CONTRIBUTIONS

Juncheng Ding developed the theoretical framework of the project and coordinated the overall structure and final integration of the report.

Junzhe Xie implemented the Wolff cluster algorithm and contributed to the corresponding methodological and results sections of the manuscript.

Tianchen Yu implemented the Metropolis Monte Carlo algorithm and contributed to the corresponding methodological and results sections of the manuscript.

Shangjun Qiu performed numerical simulations and data analysis for the Wolff algorithm, including result generation and visualization, and contributed to the associated sections of the report.

Zhu Liu performed numerical simulations and data analysis for the Metropolis algorithm, including result generation and visualization, and contributed to the associated sections of the report.

USE OF AI TOOLS (DISCLOSURE)

AI-based tools were used to assist with drafting text, debugging code, and improving clarity of presentation. All physical interpretations, numerical results, and simulations were independently verified by the authors to ensure correctness and consistency with established theory.

ACKNOWLEDGMENTS

We thank the course instructors and teaching staff for valuable guidance and feedback throughout the project. This work was completed as part of a computational physics course, and we acknowledge the use of standard open-source scientific computing libraries.

* dingjuncheng@berkeley.edu
† xiejunzhe22@berkeley.edu
‡ yutianchen2025@berkeley.edu
§ shangjunqiu@berkeley.edu
¶ liuzhu@berkeley.edu

[1] Lars Onsager. Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Physical Review*, 65(3-4):117–149, 1944.

[2] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[3] Ulli Wolff. Collective monte carlo updating for spin systems. *Physical Review Letters*, 62(4):361–364, 1989.

[4] Chen-Ning Yang. The spontaneous magnetization of a two-dimensional ising model. *Physical Review*, 85:808–816, 1952.

[5] Robert H. Swendsen and Jian-Sheng Wang. Nonuniversal critical dynamics in monte carlo simulations. *Physical Review Letters*, 58(2):86–88, 1987.

[6] Alan D. Sokal. Monte carlo methods in statistical mechanics: Foundations and new algorithms. *NATO ASI Series C*, 361:131–192, 1997.

[7] M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics*. Oxford University Press, 1999.

[8] Michael E. Fisher. Scaling theory for critical points. *Physical Review Letters*, 28(16):1022–1025, 1972.

[9] M. N. Barber. Finite-size scaling. *Phase Transitions and Critical Phenomena*, 8:145–266, 1983.

[10] C. M. Fortuin and P. W. Kasteleyn. On the random-cluster model. i. introduction and relation to other models. *Physica*, 57(4):536–564, 1972.

[11] Xiaoyi Li and Alan D. Sokal. Rigorous lower bounds on the dynamic critical exponent of the swendsen–wang algorithm. *Physical Review Letters*, 63(8):827–830, 1989.