Chapter 1

Monte Carlo simulations for spin systems

The purpose of this chapter is to give a brief introduction to Monte Carlo simulations and their applications in the numerical analysis of phase transitions in statistical physics models. We will start by illustrating the general idea of importance sampling Monte Carlo methods, followed by the different type of update schemes (Metropolis, Wolff-cluster, etc.) to sample non-uniform distributions efficiently. We will also pay attention to the topic of statistical analysis of the generated data via autocorrelation analysis and statistical error analysis. For illustration purposes, we will primarily focus our attention on the simplest spin model, i.e.,the two-dimensional Ising model (without external field) and discuss the finite-temperature critical phenomena and phase transition. Finally, we will demonstrate the calculation of the critical exponents of the two-dimensional Ising model using finite-size scaling analysis of relevant physical observables. Although we start with a classical statistical model, these ideas can be generalized and are applicable in simulations of quantum-spin systems, where they become the *Quantum Monte Carlo* (QMC) methods.

1.1 Introduction to Monte Carlo methods

Monte Carlo simulations are an important and broad class of stochastic methods that utilize randomness to solve deterministic problems efficiently. To show their utility, we will start with a simple and illustrative example. Consider the calculation of a thermal expectation value of an observable $Q(x)$ in statistical physics;

$$\langle Q \rangle = \int_0^L \mathrm{d}x\, \rho(x) Q(x), \qquad \int_0^L \mathrm{d}x\, \rho(x) = 1, \tag{1.1.1}$$

where $\rho(x)$ is the underlying probability distribution.

The most naive way to estimate this integral numerically is to use *Euler's method* to

discretize the integration range into N pieces and perform a discretized sum

$$\langle Q \rangle \approx \Delta x \sum_{j=1}^{N} \rho(x_0 + j\Delta x)\, Q(x_0 + j\Delta x) \tag{1.1.2}$$

where $\Delta x = L/N$. Such grid-based methods are very accurate for low-dimensional integrals, but as we go to higher-dimensional integrals, both the error-scaling and computational costs grow significantly.

A more efficient method for performing high-dimensional integrals is known as *Monte Carlo integration* where the points are uniformly sampled across the integration range instead of a grid. If the uniformly sampled random points are denoted by the set $\{x_1, x_2, x_3, \ldots, x_N\}$, then the integral in Eq. (1.1.1) can be estimated as

$$\langle Q \rangle \approx \frac{L}{N} \sum_{i=1}^{N} \rho(x_i)\, Q(x_i) \tag{1.1.3}$$

The error in a Monte Carlo estimate goes as $1/\sqrt{N}$ in any number of dimensions, and hence it is more efficient for estimating $6\mathcal{N}$-dimensional integrals characteristic of statistical mechanics of $\mathcal{N}$ particles. However, this straightforward unbiased Monte Carlo integration only works well in practice if the integrand isn't sharply peaked in some small region of the integration range.

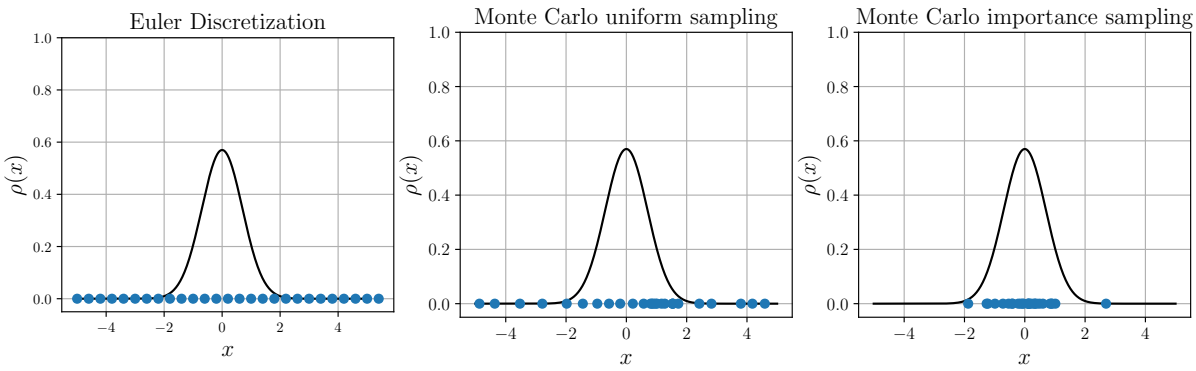


Figure 1.1: Comparision of sampling techniques.

If $\rho(x)$ is sharply peaked in a small region, then uniform sampling of points can result in large statistical fluctuations as only a small fraction of points will fall within the dominant range. Therefore, the next improvement can be performed by sampling the points according to a probability distribution $W(x)$ which is peaked in the same region as $\rho(x)$. This gives the estimate of the expectation value as

$$\langle Q \rangle \approx \frac{L}{N} \sum_{i=1}^{N} \frac{\rho(x_i)}{W(x_i)} Q(x_i)\, W(x_i) \approx \frac{1}{N} \sum_{i=1}^{N} {}^{(W)} \frac{\rho(x_i)}{W(x_i)}\, Q(x_i) \tag{1.1.4}$$

where $\sum^{(W)}$ denotes points being sampled from the distribution $W(x)$, and we write $L\sum W(x_i) \to \sum^{(W)}$. Often, a good solution is to use $W(x) = \rho(x)$, and the expectation value becomes an arithmetic average of $Q(x)$ over the configurations sampled by $\rho(x)$

$$\langle Q \rangle \approx \frac{1}{N} \sum_{i=1}^{N} {}^{(\rho)} Q(x_i) \tag{1.1.5}$$

This technique is known as the *Monte Carlo Importance Sampling* method since we are only sampling the points lying in the "important" region of the probability distribution $\rho(x)$.

In statistical mechanics, the probability distribution is generally the Boltzmann distribution $\rho(\vec{x}, \vec{p}) = e^{-\beta H(\vec{x},\vec{p})}$, and we can use Monte Carlo importance sampling to calculate expectation values of physical observables. However, in the above discussion, we have ignored the problem of sampling points according to a given probability distribution. We discuss this in the following subsection.

1.1.1 Markov Chains and Detailed Balance condition

In order to calculate integrals via the method of importance sampling, we need a way to sample points according to the probability distribution $\rho(x)$. The theory of Markov Chains provides us with the necessary tools to generate a Markov Chain process which evolves towards a desired equilibrium distribution.

In physicists' language, a Markov chain is a discrete chain of events $C_1 \to C_2 \to C_3 \to \ldots \to C_N$ that evolves stochastically and satisfies the Markovian property, i.e., the probability of $C_{i-1} \to C_i$ transition is independent of its history. Put together, this implies the probability of obtaining the above sequence is

$$P(C_1 \to C_2 \to \ldots \to C_N) = P(C_1) \cdot P(C_2|C_1) \cdot P(C_3|C_2) \ldots P(C_N|C_{N-1}) \tag{1.1.6}$$

Roughly speaking, if the Markov chain doesn't repeat itself and can reach *any* configuration starting from *any other* configuration, then it is *ergodic* and settles onto a stationary distribution. By designing an appropriate Markov chain, it is possible to obtain any desired stationary distribution $\rho(C)$.

Let us now assume we have a set of all possible configurations $\{X\} = \{X_1, X_2, \ldots, X_n\}$ in the configuration space. Assume we start with some configuration $X_{i(0)}$ and stochastically generate a Markov chain $X_{i(1)}, X_{i(2)}, X_{i(3)}, \ldots, X_{i(M)}$. We can do the same for an ensemble of configurations initially distributed according to $\rho(X)$. At the update 0 , the number of configurations X_i in the initial ensemble is $N_0(X_i) \propto \rho(X_i) \Rightarrow N_0(X_i) = \mathcal{N}\rho(X_i)$. The given Markov chain must have an update scheme which stochastically evolves the

ensemble to the next set of states. The number of configurations after the update 1 is

$$N_1(X_i) = N_0(X_i) + \sum_{j \neq i} \left[N_0(X_j)\, P(X_j \to X_i) - N_0(C_i)\, P(X_i \to X_j) \right] \tag{1.1.7}$$

The first term in the sum represents configurations updating into X_i and the second term represents X_i updating out to other configurations. If we want the ensemble to remain distributed according to the initial ensemble distribution $\rho(X)$, then, for all $i = 1, 2, \ldots M$,

$$\sum_{j \neq i} \left[\rho(X_j)\, P(X_j \to X_i) - \rho(C_i)\, P(X_i \to X_j) \right] \stackrel{!}{=} 0 \tag{1.1.8}$$

One possible solution of this condition is to satisfy it term-by-term $\forall\ j$

$$\rho(X_j)\, P(X_j \to X_i) - \rho(C_i)\, P(X_i \to X_j) \stackrel{!}{=} 0 \tag{1.1.9}$$

which can be written as a ratio

$$\frac{P(X_j \to X_i)}{P(X_i \to X_j)} = \frac{\rho(X_i)}{\rho(X_j)}, \tag{1.1.10}$$

also known as the *detailed balance condition*. Although here we start with an ensemble distributed according to the probability distribution ρ, for practical purposes, neither do we need to start from the same distribution, nor do we require an ensemble of configurations. In practice, the master equation Eq. (1.1.7) takes care of the excess or deficiency and equilibrates after a characteristic *equilibration time* of Markov chain updates.

The transition probability $P(X_i \to X_j)$ can further be written as a product of the update attempt and the proposal acceptance probabilities. Since the proposal probabilities should be uniform, the detailed balance condition in terms of acceptance probabilities becomes

$$\frac{P_{\text{accept}}(X_j \to X_i)}{P_{\text{accept}}(X_i \to X_j)} = \frac{\rho(X_i)}{\rho(X_j)} \tag{1.1.11}$$

Starting from the detailed balance condition, one can define stochastic algorithms, such as Monte Carlo simulations, which generate configurations according to a desired distribution. One such common algorithm is the *Metropolis algorithm* with the solution to Eq. (1.1.11) as

$$P_{\text{accept}}(X_i \to X_j) = \min \left[\frac{\rho(X_j)}{\rho(X_i)}, 1 \right] \tag{1.1.12}$$

For statistical mechanics, the desired equilibrium distribution is the Boltzmann distribution which gives rise to the well-known Metropolis acceptance probability

$$P_{\text{accept}}(E_i \to E_j) = \min \left[e^{-\beta(E_j - E_i)}, 1 \right]. \tag{1.1.13}$$

1.2 Metropolis and the Ising model

As discussed in the last section, we can generate samples distributed according to the Boltzmann distribution if we choose the acceptance probability as defined in Eq. (1.1.13) for the Markov chain. These drawn configurations can then be utilized to calculate thermal expectation values via importance sampling.

We will discuss this simulation method in the context of the 2-dimensional ferromagnetic Ising model. The Ising model is the paradigmatic model for systems exhibiting continuous phase transition from a ferromagnetic to a paramagnetic phase at a critical temperature T_c. In the absence of an external field ($h = 0$) and only nearest-neighbor interactions, the energy of the ferromagnetic Ising model is

$$E = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j \tag{1.2.1}$$

where $\langle i, j \rangle$ indicates nearest-neighbors i, j and the expression for a 2-dimensional square lattice can be more suggestively written as

$$E = -J \sum_{i} \sigma_{i_x,i_y} \left[\sigma_{i_x,i_y+1} + \sigma_{i_x+1,i_y} \right] \tag{1.2.2}$$

For a Markov chain transition $\sigma_i \to -\sigma_i$, the difference in energy between the configurations is given by

$$\Delta E = 2J \, \sigma_{i_x,i_y} \left[\sigma_{i_x,i_y+1} + \sigma_{i_x+1,i_y} + \sigma_{i_x,i_y-1} + \sigma_{i_x-1,i_y} \right] \tag{1.2.3}$$

This leads to the Metropolis acceptance probability for the Ising model spin flips being given by

$$P_{\text{accept}}(\sigma_i \to -\sigma_i) = \min \left[e^{-\beta \Delta E}, 1 \right] \tag{1.2.4}$$

with ΔE being defined by Eq. (1.2.3). In a nutshell, a *Metropolis Monte Carlo* simulation of the Ising model consists of performing such spin flip proposals by selecting the spin σ_i to be flipped at random and generating a Markov chain distributed according to the Boltzmann distribution. A single *Monte Carlo sweep* then consists of L^2 such spin flip proposals so that roughly each lattice site gets an equal chance to flip, where L denotes the side length of the square lattice. In algorithmic language, a Monte Carlo sweep is defined as follows:

1. Randomly choose a spin σ_i on the lattice.
2. Propose the spin flip $\sigma_i \to -\sigma_i$.
3. Calculate the difference in energy $\Delta E = E_{\text{final}} - E_{\text{initial}}$.
4. Accept the proposed move with a probability of $\min \left[e^{-\beta \Delta E}, 1 \right]$.
5. Steps 1 to 4 are then repeated $\mathcal{N} = L^2$ times.

The Boltzmann sampling performed using Markov Chain Monte Carlo (MCMC) can then be utilized in calculating expectation values of observables as simple statistical averages over the sampled points

$$\langle \mathcal{O} \rangle \approx \frac{1}{N} \sum_{\{\sigma\}} \mathcal{O}(\sigma) \tag{1.2.5}$$

where we have suppressed the superscript $^{(\rho)}$ on the sum denoting sampling over the distribution $\rho(\{\sigma\}) = e^{-\beta E(\{\sigma\})}$. Note that such Monte Carlo averages are always supposed to be taken over *equilibrated* configurations in a Monte Carlo simulation. As discussed earlier, the master equation (1.1.7) ensures the equilibration of the Markov chain's probability distribution after a few steps by taking care of appropriate excess or deficiencies during the initial steps. This initial equilibration period where no measurements are performed is known as the *equilibration time* or the *thermalization time*, often denoted by t_{eq}.

The Metropolis MCMC method is thus a conceptually simple and quite universally applicable algorithm. Its applications range from simulations of simple Ising chains to highly non-trivial spin systems. However, the Metropolis algorithm comes with its fair share of problems which we will discuss later. In the next section, we will discuss the relevant statistical physics of the Ising model and the measurement of physical observables.

1.3 Physical observables and derived quantities

To set the scenery for calculating observable expectation values, we will first begin by discussing the relevant physical observables for the Ising model. Since we are dealing with spin systems, a natural quantity of interest is the magnetization, which acts as the *order parameter* of the Ising phase transition at $T_c \neq 0$. The magnetization per site is defined by the observable

$$m \equiv \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \sigma_i \tag{1.3.1}$$

where $\mathcal{N} = L^2$ is the total number of lattice sites. One can explicitly check that an operation which rotates all spins $\sigma_i \to -\sigma_i \ \forall\, i$ is a symmetry of the Ising Hamiltonian (Eq. (1.2.1)). Although we are technically not yet dealing with quantum mechanics, this operation can be denoted as $F = \prod_{i=1}^{N} X_i$ where X_i denotes a flip of spin at site i. Since applying this operation twice is equivalent to identity ($F^2 = \mathbb{1}$), we often say that the Ising model possesses a global $\mathbb{Z}_2$ symmetry.

As a result of this global $\mathbb{Z}_2$ symmetry, both negative and positive magnetization are equally probable. However, in the thermodynamic limit $\mathcal{N} \to \infty$ (infinitely large lattice), if the system attains a state with spins predominantly in one direction below $T < T_c$, then it cannot spontaneously (in a finite time) fluctuate through a series of local spin flips into a state with the opposite magnetization. This is known as the *spontaneous symmetry-breaking* of the global $\mathbb{Z}_2$ symmetry in the Ising model, and can be physically thought of

as arising from the limit of a very weak external field. However, in a finite system, such fluctuations are possible and it is possible for $\langle m \rangle = 0$ for all temperatures T, making it difficult to study the phase transition which appears in the thermodynamic limit.

Therefore, in simulations of finite lattice sizes, **we instead define the order parameter as $\langle |\boldsymbol{m}| \rangle$.** As the system size increases, the $\langle |m| \rangle$ curve sharpens close to T_c and approaches the thermodynamic limit symmetry-broken $\langle m(T) \rangle$ curve.

Similarly, the other relevant physical observable is the energy density which is simply defined as the energy per site

$$e = \frac{E}{\mathcal{N}} = \frac{-J}{\mathcal{N}} \sum_{\langle i,j \rangle} \sigma_i \sigma_j \tag{1.3.2}$$

We can also use magnetization and energy density observables to calculate other derived quantities of interest. In the context of magnetization, we can calculate the magnetic susceptibility

$$\chi \equiv \left. \frac{\partial \langle M \rangle}{\partial h} \right|_{h=0} \tag{1.3.3}$$

i.e. the linear response of magnetization in the weak field limit, i.e., $E' = \lim_{h \to 0}(E - hM)$. It is easy to show that the expression for susceptibility simplifies to

$$\chi = \frac{1}{T}\left(\langle M^2 \rangle - \langle |M| \rangle^2 \right) = \frac{L^2}{T}\left(\langle m^2 \rangle - \langle |m| \rangle^2 \right) \tag{1.3.4}$$

where we have replaced the $\langle m \rangle$ with $\langle |m| \rangle$ for reasons discussed previously. Similarly, the specific heat capacity is another quantity of interest which is defined as

$$C_V = \frac{\partial \langle E \rangle}{\partial T} = \frac{L^2}{T^2}\left(\langle e^2 \rangle - \langle e \rangle^2 \right) \tag{1.3.5}$$

Finally, we are also interested in the derived *Binder cumulant* which is defined as the kurtosis of the order parameter $\langle |m| \rangle$

$$U_L = 1 - \frac{\langle m^2 \rangle_L}{3 \langle m^2 \rangle_L}, \tag{1.3.6}$$

and the intersection of U_L curves for different lattice side L is used to accurately determine the phase transition point.

Therefore, the measurements of the set of physical observables $\{|m|, m^2, m^4, e, e^2\}$ is sufficient to determine the Q vs. T curves for all the relevant observables $(e, |m|)$ and derived quantities (χ, C_V, U_L). Using Monte Carlo simulations, we can calculate the expectation values and statistical errors in these quantities, and use the simulation data to extract the critical exponents, which we will discuss in later sections.

1.4 Autocorrelation functions

All MCMC algorithms are based upon the central idea of performing local or cluster updates to arrive at the next configuration (or the next state in a Markov chain). Naturally, it is reasonable to expect that the observable measurements are temporally correlated and these correlations are quite significant near the transition points. The only way to get around this is to perform the measurements of the observables after a few time steps so as to ensure that subsequent measurements are statistically uncorrelated. The *autocorrelation function* is a quantitative measure of temporal correlations between measurements. For an observable $\mathcal{O}$, the autocorrelation function is defined as

$$A_{\mathcal{O}}(t) = \frac{1}{N-t} \sum_{k=0}^{N-t-1} \frac{(\mathcal{O}(k) - \langle \mathcal{O} \rangle) \cdot (\mathcal{O}(k+t) - \langle \mathcal{O} \rangle)}{\langle \mathcal{O}^2 \rangle - \langle \mathcal{O} \rangle^2} \tag{1.4.1}$$

where $\mathcal{O}(k)$'s are the measurements performed at each MC sweep, and N is the total number of MC sampling sweeps. Moreover, the averages in Eq. (1.4.1) are calculated only over the first $N - t$ measurements.

$$\langle \mathcal{O}^n \rangle = \frac{1}{N-t} \sum_{k=0}^{N-t-1} [\mathcal{O}(k)]^n \tag{1.4.2}$$

The autocorrelation function is suitably normalized to give values $\in [-1, 1]$. For an ergodic simulation, we expect $A_{\mathcal{O}}(t) \to 0$ as $t \to \infty$; infact, $A_{\mathcal{O}}(t)$ is expected to fall off as an exponential

$$A_{\mathcal{O}}(t) \sim e^{-|t|/\tau}, \tag{1.4.3}$$

and τ is called the *autocorrelation time*. The autocorrelation time is roughly the number of MC steps after which the temporal correlations drop to $1/e \approx 0.37$. A general rule of thumb in MCMC simulations is to **perform measurements of the observables after every 2τ steps.** For an observable $\mathcal{O}$, we define the *integrated autocorrelation time* τ_{int} as a discretized integration sum

$$\tau_{\text{int},\mathcal{O}} = \frac{1}{2} \sum_{t=-\infty}^{\infty} A_{\mathcal{O}}(t) = \frac{1}{2} + \sum_{t=1}^{\infty} A_{\mathcal{O}}(t) \tag{1.4.4}$$

However, this natural estimator isn't *quite* correct because the autocorrelations $A_{\mathcal{O}}(t)$ for $t \gg \tau_{\text{int}}$ contain much noise and little signal. To fix this, we implement the *automatic windowing algorithm* by introducing a cutoff M on the sum

$$\tau_{\text{int},\mathcal{O}} = \frac{1}{2} + \sum_{t=1}^{M} A_{\mathcal{O}}(t) \tag{1.4.5}$$

M is chosen as the smallest integer such that $M \geq 6\tau_{\text{int},\mathcal{O}}(M)$. The algorithm can be summarized as follows

1. Start with some small value of M, say $M = 50$, and evaluate $\tau_{\text{int},\mathcal{O}}(50)$.
2. Check if $50 \geq 6\tau_{\text{int},\mathcal{O}}(50)$.
3. If not, increase the value of M and repeat until you find a M where $M \geq 6\tau_{\text{int},\mathcal{O}}(M)$.

We take the autocorrelation time τ_{int} of a given (L, T) simulation as the maximum of autocorrelation times for all observables except magnetization m. This is because the lack of symmetry-breaking in finite lattices causes the time taken to tunnel between positive and negative m states to be very long, leading to extremely high autocorrelations.

$$\tau_{\text{int}} = \max\left\{\tau_{\text{int},|m|},\ \tau_{\text{int},m^2},\ \tau_{\text{int},m^4},\ \tau_{\text{int},e},\ \tau_{\text{int},e^2}\right\} \tag{1.4.6}$$

The first run of the simulation is only used to calculate the integrated autocorrelation time τ_{int}, followed by a second run where measurements are taken after every $2\tau_{\text{int}}$ steps. These statistically independent measurements can then be used to compute the Monte Carlo expectation values and errors.

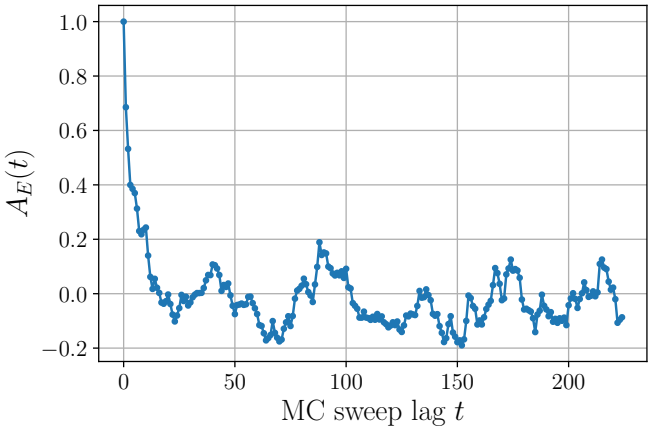


Figure 1.2: Exponential decay of energy density autocorrelation function.

1.5 Estimation of expectation values and errors

Once we have a set of uncorrelated Monte Carlo observable measurements, it is a simple task to calculate the mean and standard deviation of the uncorrelated observable measurement data $(|m|, e)$. However, for quantities that are not measured directly in the simulation but are computed as non-linear combinations of basic observables (χ, C_V, U_L), deriving a propagated error is a highly non-trivial task. Therefore, we utilize *Jackknife analysis* to create large blocks/bins of the data and perform the required computations.

1.5.1 Jackknife binning analysis

We begin by binning the N measurements of observable $\mathcal{O}$ into N_B non-overlapping bins of length k such that $N_B k = N$, and construct a shorter time series of bin averages. The

bin average for the j^{th} bin is defined as

$$\mathcal{O}_j^{(B)} \equiv \frac{1}{k}\sum_{i=0}^{k-1}\mathcal{O}_{jk+i} \tag{1.5.1}$$

Below is a visual representation of the binning process

$$\{\underbrace{\{\mathcal{O}_0, \mathcal{O}_1 \ldots \mathcal{O}_{k-1}\}}_{\overline{\mathcal{O}}_0^{(B)}}, \underbrace{\{\mathcal{O}_k, \mathcal{O}_{k+1} \ldots \mathcal{O}_{2k-1}\}}_{\overline{\mathcal{O}}_1^{(B)}}, \ldots\ldots, \underbrace{\{\mathcal{O}_{(N_B-1)k}, \mathcal{O}_{(N_B-1)k+1} \ldots \mathcal{O}_{N-1}\}}_{\overline{\mathcal{O}}_{N_B-1}^{(B)}}\}$$

$$\Downarrow$$

$$\{\overline{\mathcal{O}}_0^{(B)}, \overline{\mathcal{O}}_1^{(B)}, \overline{\mathcal{O}}_2^{(B)} \ldots, \overline{\mathcal{O}}_{N_B-1}^{(B)}\}$$

Knowing the bin averages for the observables $\mathcal{O} \in \{|m|, m^2, m^4, e, e^2\}$, the bin estimates of the derived quantities are estimated as

$$\chi_j^{(B)} = \beta\, L^2 \left(\overline{m^2}_j^{(B)} - \left[\overline{|m|}_j^{(B)}\right]^2\right) \tag{1.5.2}$$

$$C_V{}_j^{(B)} = \beta^2\, L^2 \left(\overline{e^2}_j^{(B)} - \left[\overline{e}_j^{(B)}\right]^2\right) \tag{1.5.3}$$

$$U_L{}_j^{(B)} = 1 - \frac{\overline{m^4}_j^{(B)}}{3\left[\overline{m^2}_j^{(B)}\right]^2} \tag{1.5.4}$$

We denote these derived quantities by ρ such that $\rho \in \{\chi, C_V, U_L\}$. The means over the bin averages (for $\mathcal{O}$) and bin estimates (for ρ) are also calculated

$$\overline{\mathcal{O}} = \overline{\overline{\mathcal{O}}_j^{(B)}} = \frac{1}{N_B}\sum_{i=0}^{N_B-1}\overline{\mathcal{O}}_j^{(B)} \tag{1.5.5a}$$

$$\overline{\rho} = \overline{\rho_j^{(B)}} = \frac{1}{N_B}\sum_{i=0}^{N_B-1}\rho_j^{(B)} \tag{1.5.5b}$$

For the Jackknife error analysis, we begin by constructing the same number (N_B) of Jackknife bins containing all data but the j^{th} bin of the previously mentioned binning method. The Jackknife averages for these new bins are defined as

$$\overline{\mathcal{O}}_j^{(J)} = \frac{N\overline{\mathcal{O}} - k\overline{\mathcal{O}}_j^{(B)}}{N-k} \tag{1.5.6}$$

$$\overline{\rho}_j^{(J)} = \frac{N\overline{\rho} - k\rho_j^{(B)}}{N-k} \tag{1.5.7}$$

Since we are using the same data again and again while forming the Jackknife bins, these bins are trivially correlated, and as a result, the Jackknife bin variance is much smaller

than is expected. This can be corrected by multiplying the sample variance by another factor of $(N_B - 1)^2$ resulting in

$$\delta\overline{\mathcal{O}} = \sqrt{\frac{N_B - 1}{N_B} \sum_{j=0}^{N_B-1} \left(\overline{\mathcal{O}}_j^{(J)} - \overline{\mathcal{O}}\right)^2} \tag{1.5.8a}$$

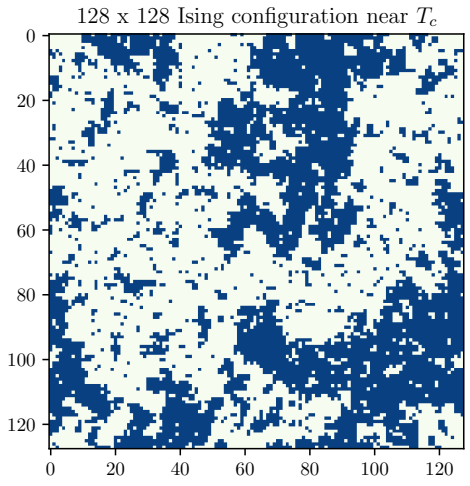
$$\delta\rho = \sqrt{\frac{N_B - 1}{N_B} \sum_{j=0}^{N_B-1} \left(\overline{\rho}_j^{(J)} - \overline{\rho}\right)^2} \tag{1.5.8b}$$

Therefore, using the calculated values from Eq. (1.5.5) and Eq. (1.5.8), we get our mean and error estimates of the observable expectation values and the derived physical quantities

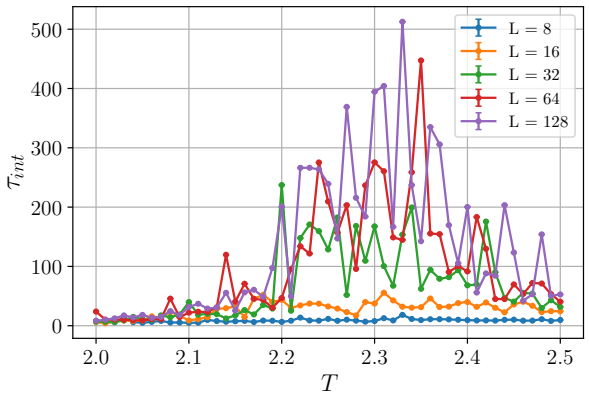
$$\text{Estimate of } \langle\mathcal{O}\rangle \to \overline{\mathcal{O}} \pm \delta\overline{\mathcal{O}}, \qquad \text{Estimate of } \rho \to \overline{\rho} \pm \delta\rho$$

1.6 Critical slowing down near phase transition

Phase transitions, in general, are associated with a discontinuity in the first derivative (for first-order phase transitions) or a divergence in the second derivative (for second-order phase transitions). For example, the classical Ising model is characterized by a second-order phase transition at a critical temperature T_c. Above this temperature ($T > T_c$), the spins are randomly aligned and correspond to the *paramagnetic* phase. Below this temperature ($T < T_c$), the spins majorly align in one of the directions to form the *ferromagnetic* phase of the system.



(a) Domain formation near T_c ($T = 2.30$).



(b) High autocorrelation times near T_c for Metropolis.

Figure 1.3: Critical slowing down near the transition point in the Ising model.

In the paramagnetic phase, at high temperatures, the spin configuration is random and there are little to no spatial correlations. However, as the temperature decreases, we enter

the paramagnetic phase and spins tend to align themselves with their neighbors. As a result, we form large *domains* or *clusters* of spins and we have a high degree of spatial correlation (Fig. 1.3a), often measured by the parameter ξ, known as the correlation length. As one approaches the critical temperature T_c from the above, the onset of strong correlations results in *domain flips* which cause high fluctuations in magnetization and energy. Since the Metropolis algorithm performs local updates and the spins have a strong spatial correlation, we need to wait for a long autocorrelation time to get independent subsequent measurements (Fig. 1.3b). This is known in literature as *critical slowing down*.

The phenomena of critical slowing down is primarily dependent on the algorithm used. Since local update (such as Metropolis) take a long time to flip the domains and generate statistically independent configurations, they have a significantly higher computational cost as a tradeoff to their conceptual simplicity.

Therefore, it should be intuitively clear that some sort of non-local updates should be able to alleviate this problem by flipping entire clusters of spins. This class of algorithms is generally known as *Cluster-update algorithms*. Although the critical slowing down can be significantly reduced with cluster algorithms, they are still far less general applicable than local update algorithms.

1.7 Wolff Cluster algorithm