Self Learning Monte Carlo Method

Jie Xiong, Chizhou Wang

December 10,
Ecole Polytechnqiue

Contents

Abstract

Monte Carlo simulation is a useful tool in the research of condensed matter physics to solve many-body problems. However, traditional Hastings-Metropolis method fails to describe the long range interactions of a system, thus unable to perform well in Ising models at phase transition temperature. To solve this problem, an efficient global update algorithm is needed to accelerate the simulation. In our project, we implement the SLMC method proposed in literature [7], which uses machine learning method to help build clusters with higher acceptance ratio in order to speed up the simulation. This optimized updating method makes the simulation 6-10 time faster than naive Wolff global update.

1 Introduction

Ising model is a mathematical model of ferromagnetism in statistical physics.[6, 9, 10] This model consists of discrete spins arrayed in some kind of order, whose status is represented by quantum number +1 or -1. These spins can interact with their neighbours. Neighbouring spins which agree have lower energy than those disagree. The system tends to have lower energy, while thermal movement will disturb this tendency, which makes the system status follow the Boltzmann distribution.

Obviously, the complexity of an Ising model increases with its dimension. Also, besides the nearest-neighbour interactions, there could be more complex items in the Hamiltonian, such as the interaction among the four spins in the same plaquette. Although those interactions may follow simple equations, the massive quantity of the spins means that it's very difficult to solve most Ising models with straightforward numerical methods.

The Monte Carlo (MC) method is a powerful computational algorithm, using randomness to solve problems with many coupled degrees of freedom, like Ising models[11]. By creating a large number of configurations in a Markov chain according to a specific probability distribution, the MC method can obtain the exact expected value of physical variables with the mean value of them in all samples.

Usually, the MC method uses updates which follow the probability distribution to generate samples from a random status or a previous sample. For example, the local update chooses a random spin and flips it with the probability determined by the energy difference between the status before and after the flip according to the Boltzmann distribution. Close to the phase transition point of the system, the Hastings-Metropolis Update method becomes inefficient because the samples it generates by updates are highly correlated. A huge number of updates are needed to generate an independent sample. If samples are correlated to each other, the mean value of physical quantities will diverge from the expectation value of the system. In order to obtain accurate results with less calculation amount, we need an update method that can decorrelate the system from its starting point after fewer updates. The global update method, which flips a group of spins (or a cluster) at one update, is a possible solution to this conflict. However, since the global update may follow the specific probability, it is highly challenging to build an efficient and general update method which uses a simple calculation to generate clusters with a high probability to flip.

Such a global update method could be developed by machine learning, or self-learning[3]. Machine learning is able to uncover the unobvious relations between different properties based on a mass of samples, accordingly using a set of quantities from the data to predict some features of it. In the case of global update Monte Carlo method, the probability to flip a cluster is related to both the system status and the extension approach to building the cluster. In the Self-learning Monte Carlo method(SLMC), the configurations of a large number of samples generated by local updates are used to find the link between the correlation method to extend the cluster and the probability to flip it, which both depend on the Hamiltonian. Depending on this link, we can eventually find a simple approach to build the cluster with a high flipping rate at the same time.[7]

In this report, we will first introduce the physical model we are researching and the methods to deal with the model, including classical Monte Carlo methods and modified versions. Then we will briefly introduce our codes framework in section 3. Results of the numerical experiments and corresponding discussions are in section 4. Finally, we conclude and make comments on the outlook of this method.

2 Models and Methods

2.1 2D Ising model

To illustrate the application of SLMC in the Ising model, we first study a classical model on a two-dimensional (2D) square lattice with the plaquette interactions, whose Hamiltonian is written as

$$H = -J \sum_{\langle i,j \rangle} S_i S_j - K \sum_{ijkl \in \square} S_i S_j S_k S_l - h \sum_i S_i \tag{1}$$

where $S_i = \pm 1$ is the spin in the site i. J is the coefficient of nearest-neighbour(NN) interactions, while K is the coefficient of the interaction between four spins in a plaquette. h reflects the external magnetic field. In the case of ferromagnetic that we are studying, both J and K are positive. In our study we choose $J = 1$ and $K = 0.2$.

Such a system has a phase transition from the paramagnetic phase at high temperature(spin directions are random) to the ferromagnetic phase at low temperature(almost all spins follow the direction of the external field). Here we focus on the phase transition temperature T_c with $h = 0$. Even though we plan to study an Ising model without the external magnetic field, it is still meaningful to look into its magnetic susceptibility χ, which is an important characteristic of a spin system. It means the partial derivative of the magnetization to the external magnetic field at a fixed temperature.

$$\chi \equiv \frac{1}{N} (\frac{\partial M}{\partial h})_T \tag{2}$$

Analogous to the magnetic susceptibility, another physical quantity showing how the system reacts to the environment change is the specific heat

$$c_V \equiv \frac{1}{N} (\frac{\partial E}{\partial T})_h \tag{3}$$

which describes how much heat the system absorbs as the temperature changes.

2.2 Autocorrelation function

Before studying different kinds of MC methods, we should first know how to evaluate their efficiency. As we know in the introduction part, the Ising model gives the expected physical quantities by averaging massive independent samples. MC methods generate such samples by updating the system status and forming a Markov chain containing the status of every step. Pragmatic samples are then selected from this chain with an interval long enough to make the two samples independent. We want to decrease the number of steps and accelerate the calculation, so a MC method had better decorrelate the system from the previous status within as few update steps as possible. To describe the correlation, we first choose a physical value O in the system, such as the magnetization. Then we consider the evolution of this value in a Markov chain $\cdots \to A(t-1) \to A(t) \to A(t+1) \to \cdots$. This can be written as $\cdots \to O(t-1) \to O(t) \to O(t+1) \to \cdots$, where $O(t) \equiv O(A(t))$. Here t represents the step, since it can be viewed as a time variable. In fact, it directly affects the calculation we need for each method. With the values in the chain, the autocorrelation function is defined as

$$\mathbf{A}_O(\Delta t) \equiv \langle O(t) O(t + \Delta t) \rangle - \langle O(t) \rangle^2 \tag{4}$$

Due to ergodicity, we use the time-average method instead of the ensemble-average method to calculate a quantity in the MC method. Therefore, autocorrelation function has a form

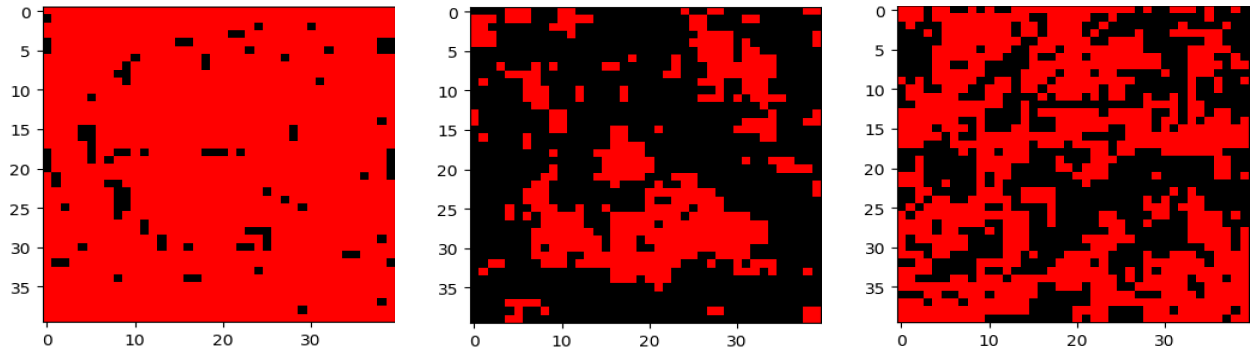
$$\mathbf{A}_O(\Delta t) \equiv \frac{1}{N-\Delta t}\sum_{t=0}^{N-\Delta t} O(t)O(t+\Delta t) - [\frac{1}{N}\sum_{t=0}^{N} O(t)]^2 \tag{5}$$

We find that if the system at $t+\Delta t$ is more correlated to its previous status at t, $\langle O(t)O(t+\Delta t)\rangle$ will be closer to its maximum value $\langle O(t)^2\rangle$. One thing to note is that in a low temperature case where M is condensed at +1/-1 or in a high temperature case where M is almost 0 in the chaotic system, the autocorrelation function based on the magnetization is always close to zero. It is easy to understand if you put a constant $O(t)$ in the auto autocorrelation function. Therefore, any kind of MC update can easily decorrelate the system from its original status at temperature significantly higher or lower than T_c. That's why we say the spin system is highly correlated from step to step near T_c.

If the Markov chain is not correlated, which is the ideal case, the autocorrelation function should decay exponentially as

$$\mathbf{A}_O(\Delta t) = e^{-\frac{\Delta t}{\tau}} \tag{6}$$

where τ is called correlation time. It shows the ability of an update method to decorrelate the system from its previous status. In actual work, this exponential decay is valid from Δt is equal to zero until the value of the autocorrelation function is very small (usually small enough to make a new independent sample). And then, autocorrelation function begins to fluctuate.



(a) T=2.2 (b) T=2.5 (c) T=3.0

Figure 1: Three instantaneous statuses of the Ising system we study. The system size is 40. The temperature is 2.2, 2.5, and 3.0. Red and black sites represent two spin directions. At a temperature near T_c(Fig.1b), we can see large-scale components in the figure. The profile of some components may stay nearly unchanged after millions of local updates.

2.3 Monte Carlo methods

In general, any MC methods are the realization of the Metropolis-Hastings algorithm with two steps. First, from configuration A, a different configuration B is generated with the probability of $Q(A \to B)$. Second, the transformation from A to B is accepted with the probability of $\alpha(A \to B)$, which is called the acceptance ratio. In order to make the probabilities of generating A and B satisfy the Boltzmann distribution, the acceptance ratio should satisfy the detailed balance principle (DBP) [4]

$$\alpha(A \to B) = \min\{1, \frac{Q(B \to A)}{Q(A \to B)}\frac{W(B)}{W(A)}\} \tag{7}$$

where $W(A) = e^{-E(A)}$ and $E(A)$ is the energy of status A. Every kind of MC update applied to a system with Boltzmann distribution must satisfy this equation. Based on this principle, we can look further into some traditional MC methods and develop some more efficient SLMC methods.

2.3.1 Local update

The local update in a system with N spins randomly chooses a spin with the probability of $1/N$, so we have $Q(A \to B) = Q(B \to A) = 1/N$. The probability to accept the update from A to B is

$$\alpha(A \to B) = \min\{1, \frac{W(B)}{W(A)}\} \tag{8}$$

one is easy to prove from the definition that this acceptance ratio satisfies 7.

One local update makes little difference to the system since it is only likely to flip one spin. Here we combine N local updates operated on the system with N spins into one step. The calculation amount for one step is proportional to N, and one global update we will discuss below needs the calculation amount of the same order. So it would be fair to evaluate them by comparing their correlation time.

As we mentioned in the last subsection, the correlation time will significantly increase near T_c. This problem is especially serious for local updates and results in critical slowing-down. The main cause is that local update is only good at updating short-wavelength components at the order of the interaction distance shown in Hamiltonian. This is not a problem when the temperature is high and the structure of the system is trivial(Fig.1c) or when the temperature is low and each spin is tightly bound to the same direction(Fig.1a). However, in the vicinity of the phase transition temperature, we can see many 'islands' of sites with the same spin forming, evolving and disappearing(Fig.1b). Large-scale structures start to appear in the spin system and long-wavelength components become more important. Because of these long-wavelength correlations, the local update algorithm tends to diffuse only very slowly through the configuration space of the model, which both elongates the calculation time and degrades the quality of the output samples. Since the diffusion process here is stochastic, the number of updates it needs to process the long-wavelength components is much lager than the number of sites in the components. So, we need a MC method that considers long-wavelength correlations in the update. This leads to the development of global update.

2.3.2 Wolff-cluster update

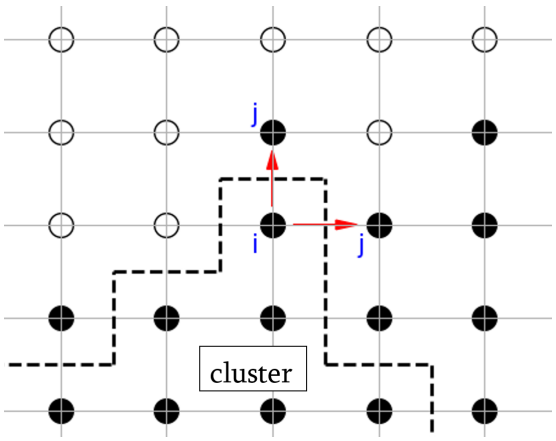


Figure 2: The extension of the cluster from a site i to its neighbours j. Only the link between two sites with the same spin can be activated.

In global updates, the construction of a cluster already takes DBP into account. For example, the Wolff-cluster update[11] extends a cluster from a random spin according to NN interaction. It constructs a cluster stochastically and flips all the spins in the cluster to obtain the new status B. The procedures to construct a cluster are as follows,

(1) Randomly choose a spin, add it to the cluster.

(2) For each spin i in the cluster, consider each nearest neighbour j outside the cluster, neighbouring site j is activated with the probability of

$$p(i \to j) = \max\{0, 1 - e^{-2\beta J S_i S_j}\} \tag{9}$$

(3) Extend the cluster by adding all activated spins into it. Then go back to (2);

(4) Finish the construction if the extension stops or all the spins have been examined.

Then we consider the probability of constructing such a cluster transforming the system from status A to B and from status B to A. According to (4) we see that site j cannot be activated if the spins of site i and j disagree. So a possible cluster only contains the spins in one direction(+1 or -1). One update between $A \to B$ and $B \to A$ will flip a cluster of spins from +1 to -1, while the other one flips the same cluster from -1 to +1. Since interactions inside the cluster concerning only spins in the same direction are equivalent for +1 or -1 spins, the probabilities to construct the cluster in the two cases above differ only at interactions across the border, which blocks the extension of the cluster. So, we just calculate the ratio between the probabilities of the two clusters that they do not extend at the boundary.

$$\frac{Q(A \to B)}{Q(B \to A)} = \prod_{\langle i,j\rangle, i\in c, j\notin c} \frac{1 - p_{i\to j}(A)}{1 - p_{i\to j}(B)} = \prod_{\langle i,j\rangle, i\in c, j\notin c} e^{-2\beta J S_i^A S_j^A} \tag{10}$$

Here c means the cluster. If the Ising model only have NN interaction, (5) will be equal to $\frac{W(B)}{W(A)}$. Then, we are able to satisfy DBP by making the acceptance ratio equal to 1 and flip every cluster we construct. Wolff-cluster update can be expanded to any Ising model whose Hamiltonian contains only two-body interactions(not necessarily NN interaction), just by concerning all the sites in connection with i in step 2.

Unfortunately, Hamiltonian with many-body interactions, like the plaquette interaction, cannot be easily included in the extension of the cluster. In order to study these systems, we first consider only NN interactions and construct the cluster with the method mentioned above. Then the acceptance ratio is used to correspond to the other interactions we leave behind. For example, in the case of the Ising model with plaquette interactions, we have

$$\alpha(A \to B) = \min\{1, e^{\Delta E}\} \tag{11}$$

where $\Delta E = K \sum_{ijkl\in\square}(S_i^B S_j^B S_k^B S_l^B - S_i^A S_j^A S_k^A S_l^A)$.

2.3.3 Self-learning global update

To extend the application of the Wolff update, we use the NN interaction component of the Hamiltonian as the approximation to it. However, it is not an ideal method. The remaining of the Hamiltonian is still significant, suggesting that the acceptance ratio in 11 should be a small value. If the cluster we construct fails to be accepted, this update will change nothing and the calculation amount will be wasted. It can seriously elongate the correlation time.

To avoid this, we need a better method to build the cluster. Of course, it should only contain two-body interactions if we do not want to totally change the creation method for the cluster. We can expand the Hamiltonian into two body interactions at different distances. This new Hamiltonian is called an effective Hamiltonian, or H_{eff}:

$$H_{eff} = E_0 - J_1 \sum_{\langle ij\rangle_1} S_i S_j - J_2 \sum_{\langle ij\rangle_2} S_i S_j - \cdots \tag{12}$$

where $\langle ij \rangle_n$ means nth nearest interaction. Whether these two-body-interaction terms form a set of complete basis remains to be proved. Nevertheless, the idea of operator expansion motivates us to approximate the original Hamiltonian with only several terms of the effective Hamiltonian. Then, the effective Hamiltonian is used to extend the cluster. To satisfy (7), we should flip the constructed cluster with the probability of

$$\alpha(A \to B) = \min\{1, \frac{W(B)}{W(A)} \frac{W_{eff}(A)}{W_{eff}(B)}\} \tag{13}$$

where W_{eff} is the energy calculated from H_{eff}.

We hope in most cases the difference between H_{eff} and H is small enough that the cluster we construct is almost always accepted. Since the difference may be influenced by the system configuration, the H_{eff}s for the same H may differ under different conditions, like the temperatures or the methods we used to construct the system. To find optimized values of J_ns can be seen as to predict the system energy from some sums of two-body spin products shown in (12). This exactly fits the region of machine learning, or specifically the linear model of it. Usually, machine learning is used to uncover some characteristics or future evolution of a system from the parameters we have about it. But here we already know the exact solution of the energy of the system. The problem is that it is not the form which can be easily applied to global updates. So we try to use machine learning method to find the H_{eff} which is able to predict the energy with limited information(two-body interaction items).

Besides, we hope the effective Hamiltonian has fewer terms so that it is easier to extend the cluster. Therefore, we test different H_{eff}s with different numbers of expansion terms and see how close they approach the exact energy in real samples. According to the calculation result, we will keep the terms which contribute the most to the effective Hamiltonian. It is expected that the terms reflecting the closer interactions are more important. This can be evaluated by calculated coefficients we obtain.

The self-learning update method needs a large amount of samples to train effective parameters $\{J_n\}$. As we mention above, one set of $\{J_n\}$ is based on a specific temperature. This results in a dilemma. We do not want to use the samples generated by inefficient local update method at T_c to train effective Hamiltonian, which is bound to give bad results, so the iteration method is used. Two optimized methods are proposed here. The first one is called 'Ensemble Average' method. We first generate samples with the local update method at a temperature T higher than T_c, and use them to train a first effective Hamiltonian H_{eff}. Then we use this H_{eff} to create samples at T_c and train effective parameters H_{eff} at T_c. Since the samples generated by different H_{eff}s may have different structure, we are afraid that only one iteration is not enough to train a stable effective Hamiltonian. Therefore, we use this effective parameter to generate more samples with self-learning update method and train new H_{eff}. We expect that after each iteration at T_c, effective Hamiltonian will approach closer to the real one and thus self-optimize itself. This idea should be test in the real calculations. The second approach is 'Temperature Descent' method, as suggested in the literature. Samples are generated at a higher temperature T_i with the local update method, where it works well. Effective Hamiltonian is trained from the first batch of samples. Then, this procedure is repeated iteratively at a set of temperatures, which form a gradient sequence from T_i to T_c. Finally, we will get the effective Hamiltonian at critical temperature. One should note that the training of the effective Hamiltonian at T_c is done in a self-optimizing way, and that's why we call this global update method "self learning update".

These two methods are used to verify each other. The numerical experiments show that they give compatible results of effective Hamiltonian.

2.3.4 Restricted self-learning update

Wolff update sometimes contains restrictions on the size of clusters[1]. This is to avoid creating large clusters with low acceptance ratio. Sometimes it's also because one wants to confine the cluster to a subset of the lattice to implement parallelism. We can apply it to SLMC, which creates the restricted self-learning Monte Carlo method(RSLMC).

In restricted self-learning update method, before we activate a link to a site which is outside the cluster, we first examine if the distance (Manhattan distance and periodic boundary condition are used here) between it and the original site of the cluster exceeds a determined limit. If so, this link will not be activated. These cutoffs need to be compensated for in the acceptance ratio. The ratio of the probabilities between constructing a cluster that could transform the system from status A to B and the contrary can be computed from this modified model

$$\frac{Q(A \to B)}{Q(B \to A)} = \frac{W_{eff}(B)}{W_{eff}(A)} \prod_{\langle ij \rangle \in r} e^{2\beta J S_i^A S_j^A} \tag{14}$$

where r is the collection of blocked spin pairs across the cluster boundary. Pay attention that any spin pairs violating the restriction should be added to the collection, even if the two spins in the pair are opposite to each other and the link cannot be activated. Because the clusters in A and B have different spin direction, and a link across the boundary is always activated in one of them. Eventually, the acceptance ratio is

$$\alpha(A \to B) = \min\{1, \frac{W(B)}{W(A)} \frac{W_{eff}(A)}{W_{eff}(B)} \prod_{\langle ij \rangle \in r} e^{-2\beta J S_i^A S_j^A}\} \tag{15}$$

Like what we do for the local update, we count N/A_r RSLMC updates as one MC step, so the correlation time of each method can be compared.

3 Framework of Codes

In this section, we will briefly introduce the framework of the codes for this project. The source codes are available in our Github repository. Our goal is to make the codes more concise, logical and well-organized.

To realize conciseness and logic, we separate the codes into two parts based on their functions. The first part aims to construct a modified Ising model used in this project and implement different update methods, which, of course, is the core part. The second part aims to realize specific computing tasks, for example, to calculate how physical quantities vary with temperature in the current research model or to determine the critical temperature based on the binder cumulant method. In this part of codes, we directly call the class files of the previous part to perform calculation tasks. In this way, the codes show good logic.

In order to organize the codes well , the codes in the first part are wrapped into several classes and written in .py files. Each of the class files either constructs the model or realizes an update method; the codes in the second part are edited into different .py files, each of which realizes a specific computing task. Furthermore, due to the fact that Monte Carlo training process is time-consuming, production and plotting are strictly separated. Calculation results are written in files, which are loaded in the plotting process. To make the codes more readable, a lot of explanatory instructions have been added to the code files, and one can view the description of a function or a class (input parameters, output, aim of the function, etc.) with the command "help".

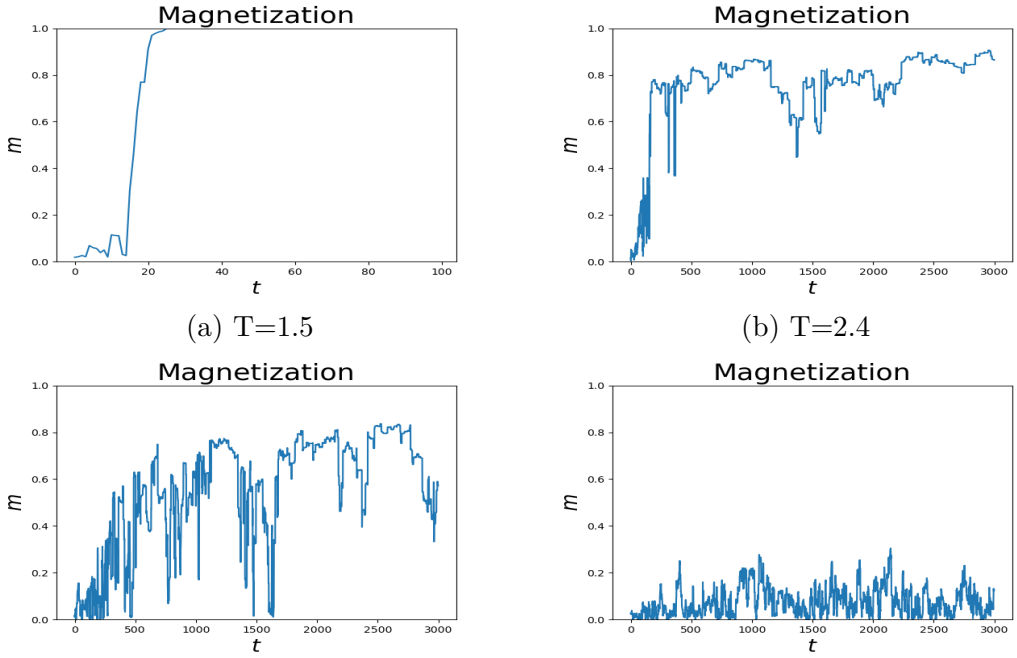
In Appendix B, we will illustrate more about the codes files in the Github Repository.

4 Results and Discussions

4.1 Overview of Ising systems under different temperatures

First of all, we study the evolution of the Ising system under different temperatures and the physical quantities as functions of the temperature. This will give us an overview of the Ising model with plaquette interactions, and enable us to move to further studies.

In this section, most data is generated by the modified Wolff-cluster update we mentioned in the last section. One thing to note is that under low temperature when most spins condense in the same direction, one Wolff update may construct a large cluster and flip nearly all the spins in the $L \times L$ square. This will flip the magnetization from close to +1 to nearly -1. The problem is that the evolution of magnetization will shift rapidly and the mean value of it will be 0, which is against reality. However, in this situation the energy will not have a sharp shift since there are no odd-order items of spins in the Hamiltonian. We can find that the system is symmetric for positive and negative spins, and anything other than magnetization itself is not influenced by a minus added to all the spins. So we choose to use the absolute value of the magnetization in our study, and it will be the same case for SLMC and RSLMC.



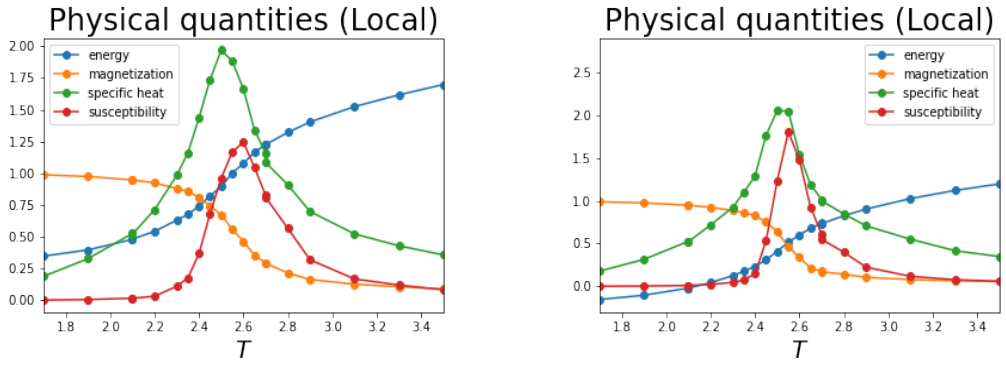
(a) T=1.5 (b) T=2.4

(c) T=2.5 (d) T=3.0

Figure 3: The value of the magnetization as the function of steps.

These figures (Fig.3) show the evolution of the magnetization as the system updates. The size of the system is 40 and we used Wolff-cluster update. Four representative figures of T=1.5, 2.4, 2.5, 3.0 is chosen to illustrate the update process from a random status with $m = 0$. Like the Ising model with only NN interactions, at a very low temperature, almost all spins are in the same direction with m close to 1 in tens of updates, and at a high temperature, the magnetization fluctuates near 0 without any obvious warm-up processes. Of course, here high and low are compared to T_c. When the temperature reaches Tc from the lower side, the magnetization fluctuates near an average value after hundreds of steps of warm-up. The mean value decays and the fluctuation magnitude increases as temperature goes up. Near Tc, the magnetization becomes quite unstable. It may stay near a value for some steps and jump to another value far away from the previous one. A mean value is still available but the numbers of warm-up steps and the steps needed to calculate the average are very big. We can see from this tendency that

the randomness of the system near Tc will take us a lot of computing resources to study its evolution and characteristics.



(a) L=25 (b) L=40

Figure 4: The evolution of physical quantities with temperature is generated by local updates. 4a shows the system with $L = 25$ and 4b shows that with $L = 40$. Larger system size results in a steeper phase transition.

Then we move to the physical quantities under the full range of temperature. We choose the system size of 25 and 40. The phase transition temperature T_c is obvious. When the temperature increases and crosses T_c, the magnetization decreases rapidly from 1 to 0, but the energy increases much more slowly. It is reasonable that the specific heat has its peak near T_c, where the system energy changes with T most significantly. The susceptibility shows a sharper peak near T_c, and the peak gets more like a delta function as the system size increases and approaches the real situation. That is because when the temperature crosses T_c and the system transforms from chaos to order, the spins will choose to follow the external magnetic field. So here a weak field near 0 can decide the system spin direction. By contrast, at a high temperature, spins are disturbed by thermal motion. Also, at a low temperature, all spins are bound to the same direction by the interactions among them. In these two cases the change of magnetic field near $h = 0$ is unlikely to make observable changes to the magnetization of the Ising system.

4.2 Self-learning results of H_{eff}

Based on the general characteristics of the system, we can train the effective Hamiltonian near T_c. To begin with, we choose the first three nearest two-body interactions and set $\{E_0, J_1, J_2, J_3\}$ as the effective parameters we would like to obtain. The constant energy E_0 is never mentioned in any MC method, so we can ignore it.

The result concluded from Tab.1 is that the interaction coefficient J_n becomes about one order of magnitude smaller as n increases by one. It is easy to understand because the farthest interaction in our Hamiltonian is just the interaction of four spins in a square. We can induce that those interactions of higher orders are weaker, and the NN interactions still dominate in this system. Since J_2 and J_3 are small values, considering them in SLMC updates would only waste time in some far links with very low activating ratios. The improvements they make for the accuracy of H_{eff} and the acceptance ratio of the final cluster will be small or even negative, which can be seen from the mean errors in Tab.1. So we had better only consider the first item of nearest interaction as the approximation to the Hamiltonian. Fig.5 shows that this linear fitting is quite ideal. Further experiments on the codes also illustrate that SLMC with $n = 3$ gives similar results as $n = 1$ method and spends much more time calculating.

The values of $\{J_n\}$ converge from the results obtained by local updates at $T = 5$ to its mean at $T = 2.5$ in almost one step. Then it begins to fluctuate near the mean(Fig.6a).

L=25	J_1	J_2	J_3	mean error
$n = 3$(SLMC)	1.24535	-0.09117	-0.01483	0.00257
$n = 3$(RSLMC)	1.25577	-0.08498	-0.01430	0.00264
$n = 1$(SLMC)	1.10701			0.00254
$n = 1$(RSLMC)	1.10735			0.00264
L=40	J_1	J_2	J_3	mean error
$n = 3$(SLMC)	1.24484	-0.08619	-0.01345	
$n = 1$(SLMC)	1.10707			
$n = 1$(RSLMC)	1.10679			
L=60	J_1	J_2	J_3	mean error
$n = 1$(SLMC)	1.10690			
$n = 1$(RSLMC)	1.10681			

Table 1: The training results of parameters in H_{eff}.

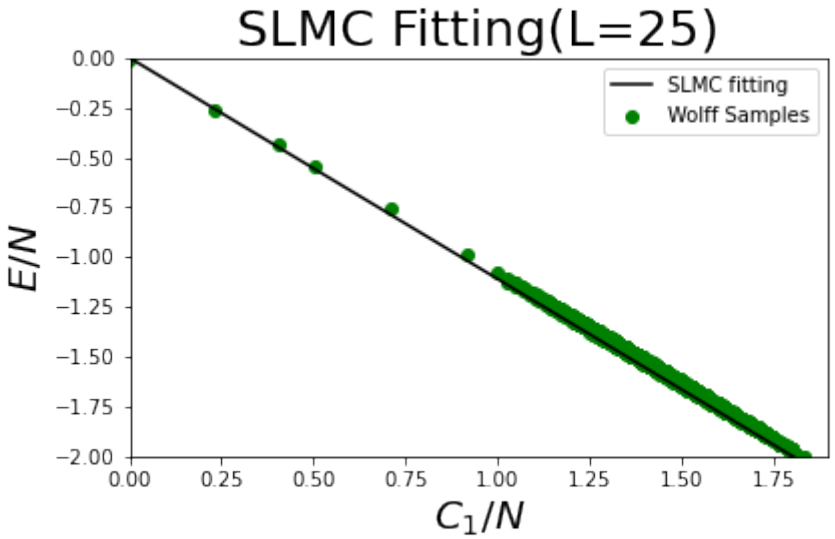
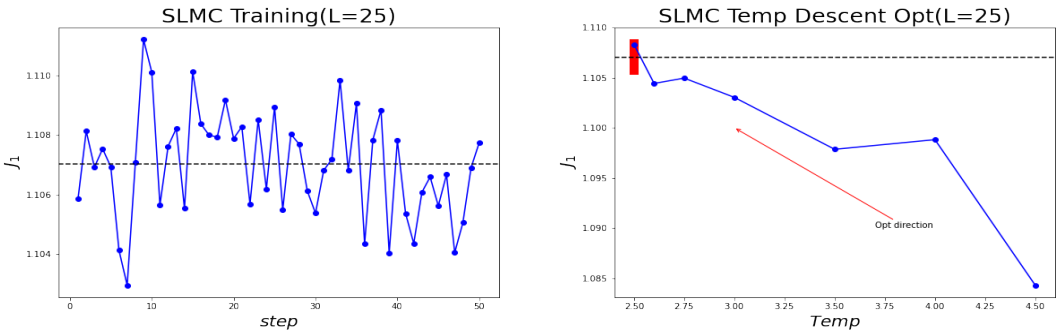


Figure 5: The SLMC fitting of J_1. C_1 is the sum of the products of nearest spins S_iS_j. All the data spots are almost on the line. The figures for models of other size or for RSLMC is similar.

This proves that our choice of skipping all the temperatures between $T = 2.5$ and $T = 5$ is acceptable. The J_1 generated by iterations from $T = 5$ to $T = 2.5$ is also shown in Fig.6b.

From Tab.1, we can see that the effective parameters we obtain from different system sizes with SLMC or RSLMC are very close to each other. We can also compare the results with those from [7]. The J_1s for $n = 1$ models only show the difference within 0.1%, while in $n = 3$ models this value is within 1%. The J_2s and J_3s' distinctions between different sets of data are a bit bigger since they are small and easy to suffer from the statistical error.

In general, the H_{eff} parameters from the training can be verified by previous research. Also, we prove that RSLMC generates the same result as SLMC, though it slightly increases the mean error. The effect of restriction can be compensated by the flipping ratio. During the calculation, we can find that under large L, RSLMC runs faster than SLMC because of the small cluster size.



(a) The value of J_1 in 50 SLMC optimizing. Their difference is within 0.01. We can use their average as the final J_1.

(b) This J_1 is iterated from $T = 5$ to $T = 2.5$ in sever intermediate temperatures one by one.

Figure 6: The value of J_1 at $T = 2.5$ is obtained by two methods. The final results are almost the same(1.0701 in 6a vs 1.0828 in 6b).

4.3 Correlation time of different methods

After creating the H_{eff}, we can compare the autocorrelation functions of local update, naive Wolff-cluster update which uses NN interaction with $J = 1$ to extend the cluster, and SLMC update7. This figure shows the exponential part of the function with the size of 25 and $T = 2.5$. We choose the exponential part of the functions. After that part, they begin to fluctuate at a low value. It is clear that SLMC updates can decorrelate the system much faster than local and Wolff updates do. By matching each function to the exponential decay, we can get the correlation time of each method at T_c. The τ of SLMC update is about 1/5 of that of Wolff-cluster update and 1/60 of local update's τ_L.

However, we only normalized τ in an approximate method. One site which is likely to be reached in one update doesn't necessarily adds the same amount of calculation time to different methods, since their algorithms are different. So, the time for each step in different methods can be different, only not in order of magnitude. The advantage of SLMC over naive Wolff-cluster update and local update is still for sure.

Wolff update shows the great advantage of global update over local update at T_c, while SLMC optimizes the method to build the clusters for global updates and improves the efficiency one step forward. As is shown in Tab.1, under the system size of 25, self-learning optimizing can generate a H_{eff} with its mean error within 0.26%. We can approach the Hamiltonian with the plaquette interactions by only NN interaction and get such close results. So the correction in the flipping ratio is very small, and the clusters we generate can be accepted with high probabilities.

4.4 Correlation time as a function of size

Then we promote the correlation time to larger systems. They are τ_L from the local update, τ_W from the naive Wolff update, τ_S from the SLMC update, and τ_R from the RSLMC method. Fig.8 shows the correlation time of four update methods as the functions of system size L. We can match the indexes α for L in these functions, which is shown in the figure. Remember that we have already made each step contain updates with the calculation proportional to N, which is the number of sites in the system. So if we would like to find a function of calculation time vs. system size L, the index may roughly be $\alpha + 2$.

As we mentioned above, τ in different methods cannot give an accurate comparison between their calculation speeds. In the experiments for systems with their size smaller than 60, RSLMC didn't show a lower speed than SLMC, even though its correlation time is larger.

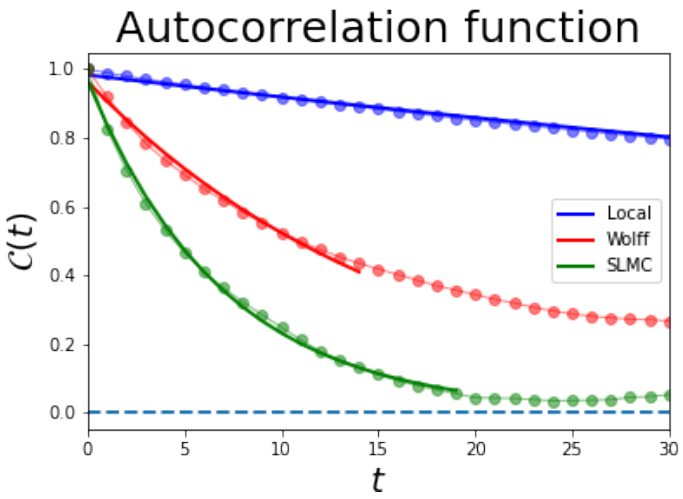


Figure 7: Autocorrelation functions of three update methods. The steps are normalized by A/N, where A is the maximum number of spins that can be flipped in one update and N is the number of sites in the system.

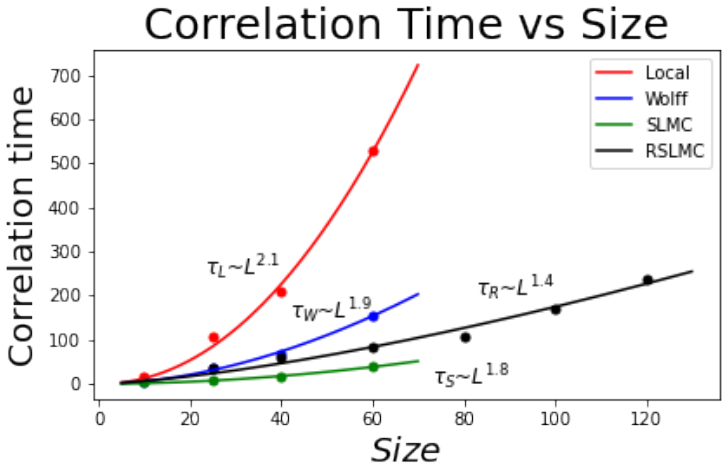


Figure 8: The behaviour of correlation times τ of local(τ_L), Wolff(τ_W), SLMC(τ_S) and RSLMC(τ_R) updates in systems of different sizes. The restriction size r for RSLMC is 10 in $L = 25$, 15 in $L = 40$, 25 in $L = 60$, 35 in $L = 80$ and 40 in others.

The three indexes in the expressions of $\tau_L(2.1)$, $\tau_W(1.9)$, $\tau_S(1.8)$ are close to each other, and their magnitude relations are just like that of τ, which means that global update and self-learning method also arrest the increase of τ with system size. Compared to them, τ_R has an obviously smaller index. The velocity that its correlation time increases with system size is much slower. Among these methods, local updates cannot deal with long-wavelength components in the system, so its speed decreases most seriously as system size increases. For SLMC updates, a larger cluster size in a bigger system would increase the length of the boundary, where the correction of the flipping ratio comes from. It leads to a smaller acceptance ratio of a cluster in SLMC updates, though SLMC only has a small difference between H and H_{eff}. However, RSLMC avoids this problem by limiting the cluster size. This will be discussed in detail in the next section. Of course, RSLMC needs more flips to decorrelate the samples and ignores some very far interactions, but usually the restriction size is big enough to cover most components in the system near T_c. So, RSLMC is an ideal method for large systems with their sizes above 60. The best restriction size L can also be achieved by self-learning.

4.5 The acceptance ratio of clusters

According to our research above, an important quantity affecting the efficiency of a global update method is the acceptance ratio of the clusters. In this section, we look into the acceptance ratios of SLMC and RSLMC in Ising systems with different L.

L	25	40	60	80	100	120
Restriction r(RSLMC)	10	15	25	35	40	40
local	0.1671	0.1707	0.1735			
Wolff	0.1565	0.1582	0.1462			
SLMC	0.5015	0.3013	0.1995			
RSLMC	0.5733	0.5875	0.5652	0.5826	0.5781	0.5826

Table 2: The acceptance ratio of clusters generated in different experiments. SLMC update in large systems with L 60 is so inefficient that we cannot get the results within tolerable time.

The results in Tab.2 verify the hypothesis that the acceptance ratios in SLMC decrease as L becomes larger, and RSLMC is able to solve this problem by limiting the cluster size. Actually, the acceptance ratio almost stays unchanged under different system sizes. That efficiently avoids the waste of time in creating a big cluster which would be abandoned with a very high probability when the Ising system is large. The insensitivity of RSLMC to system size can be very useful in practical research. A larger model can be more precise in simulating real systems since the periodic boundary condition always brings some errors, which can be illustrated by the evolution of physical quantities vs. temperature under different L in section 4.1.

A noticeable phenomenon is that the acceptance ratio of Wolff update shows little change as L becomes larger. Like SLMC, the acceptance ratio of Wolff update comes from the energy correction between H_{eff} and H on the boundaries of the clusters before and after the flip. So if the cluster size enlarges with L, the acceptance ratio of Wolff should also decrease. One explanation is that such a decay has its minimum, and the acceptance ratio of Wolff update is close to this minimum. So, it doesn't show an obvious tendency of decreasing as L changes from 25 to 40 and to 60. To prove this, we need the data of Wolff and SLMC updates in larger systems. Unfortunately, the training of H_{eff} with SLMC in larger systems is so time-consuming that we fail to achieve it.

The acceptance ratio and correlation time show a dilemma between them. The acceptance ratio for Wolff update almost doesn't change as L increases, while for SLMC the ratio decreases rapidly with the increase of L. By contrast, the τ_W of Wolff update is more sensitive to L than SLMC's τ_S. It means that SLMC has other advantages over Wolff in solving large systems besides more accurate H_{eff}(although both of them are not good at this). One possible explanation is that analogy to local update, Wolff update cannot efficiently deal with the long-wavelength components in a big system because of the inaccurate approximation to H. So, both local and Wolff updates show terrible performance under large L.

5 Conclusion and Outlook

We have reproduced the application of SLMC and RSLMC on the Ising model with plaquette interactions. Additionally, we looked further into the two-body interaction expansion of Hamiltonian. Here, self-learning largely improves the accuracy of H_{eff}, which contributes to a global update method with high acceptance ratios, small correlation time, and simple cluster construction method concerning only NN interaction. Restriction of the cluster size enables it to apply to large systems, and reduce the complexity from $O(L^{4.1})$ in the local updates to $O(L^{3.4})$ in RSLMC. The detailed mechanisms under the

behaviors of global updates with a large system size are worth further study. Pitifully we haven't operated experiments on them due to limitations on computing resources and time.

The combination of machine learning and expansion of Hamiltonian can help us transfer more systems with complex interactions into simplified models with only two-body interactions. For example, SLMC can contribute to the study of strongly correlated fermion systems lacking efficient global update methods[8]. The study can also in turn benefit our understanding of the original models.

However, the supervised machine learning we used here is quite naive. The extension of the Hamiltonian into two-body interactions is also too simple. In order to build more efficient links between spins and system characteristics, it is more proper to turn to the neural network, such as the restricted Boltzmann machine (RBM)[5]. Finding the hidden patterns and variables are just the strength of the neural network. The spins are the input and the results can be treated like a wave function. If we look further, the development of quantum computing is very likely to booster the study in SLMC, since the neural network can be treated as a quantum system. That will make many problems with unimaginable complexity accessible to us.

Acknowledgement

We thank Prof. Michel Ferrero and Prof. Arnaud Couairon for their help and instruction during our study. We would like to acknowledge CPHT for the computing resources.

Important Link

Source-code: `https://github.com/JaySchon/PHY571-Project`

References

[1] G. T. Barkema and J. F. Marko. Accelerating diffusive nonequilibrium processes in discrete spin systems. *Phys. Rev. Lett.*, 71:2070–2073, Sep 1993.

[2] K. Binder. Finite size scaling analysis of ising model block distribution functions. *Zeitschrift für Physik B Condensed Matter*, 43:119,140, June 1981.

[3] Jerome H Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. springer open, 2017.

[4] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[5] Li Huang and Lei Wang. Accelerated monte carlo simulations with restricted boltzmann machines. *Phys. Rev. B*, 95:035105, Jan 2017.

[6] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258, 1925.

[7] Junwei Liu, Yang Qi, Zi Yang Meng, and Liang Fu. Self-learning monte carlo method. *Phys. Rev. B*, 95:041101, Jan 2017.

[8] Junwei Liu, Huitao Shen, Yang Qi, Zi Yang Meng, and Liang Fu. Self-learning monte carlo method and cumulative update in fermion systems. *Phys. Rev. B*, 95:241104, Jun 2017.

[9] R. Peierls. On ising's model of ferromagnetism. *Mathematical Proceedings of the Cambridge Philosophical Society*, 32(3):477–481, 1936.

[10] Satya Pal Singh. The ising model: Brief introduction and its application. In Subbarayan Sivasankaran, Pramoda Kumar Nayak, and Ezgi Günay, editors, *Metastable, Spintronics Materials and Mechanics of Deformable Bodies*, chapter 8. IntechOpen, Rijeka, 2020.

[11] Ulli Wolff. Collective monte carlo updating for spin systems. *Phys. Rev. Lett.*, 62:361–364, Jan 1989.

A Determine Critical Temperature with Binder Cumulant Method

Binder and Heerman[2] define the reduced fourth-order cumulant (i.e. the Binder cumulant) for a lattice of linear size L as:

$$U_L = 1 - \frac{\langle M^4 \rangle_L}{3 \langle M^2 \rangle_L^2} \tag{16}$$

where $\langle M^2 \rangle_L$ is the mean of the squares of the ordered parameter (in our case, magnetization) with lattice size L and $\langle M^4 \rangle_L$ is the mean of the fourth powers of the ordered parameter.

It is easy to prove that Binder cumulant converges to 0 at high temperature as the system is in a disordered phase, magnetization fluctuates around zero obeying Gaussian distribution. It is more difficult to prove that U_L tends to $U_\infty = 2/3$ when $T < T_c$. More surprisingly, Binder and Geerman showed that at critical temperature T_c, binder cumulant is a constant independent of size (ignore scaling correction). This is of great help for us to determine the critical temperature T_c of the system.

If we can plot the Binder cumulant value of different sizes versus temperature, then the cross point of these curves corresponds exactly to T_c. We implement this idea with Wolff cluster global update method, and the result is shown in Figure 9. Note that we do a normalization and set Binder cumulant to be 1 at a very low temperature. We conclude from the graph that for the researched system (modified Ising model with the plaquette interaction, take $J = 1$, $K = 0.2$), the critical temperature is approximately 2.500. Further experiments show that this value shows somewhat stability with size, i.e. it doesn't change much from $L = 10$ to $L = 60$. The critical temperature determined here shows consistency with the result obtained from Figure 4.

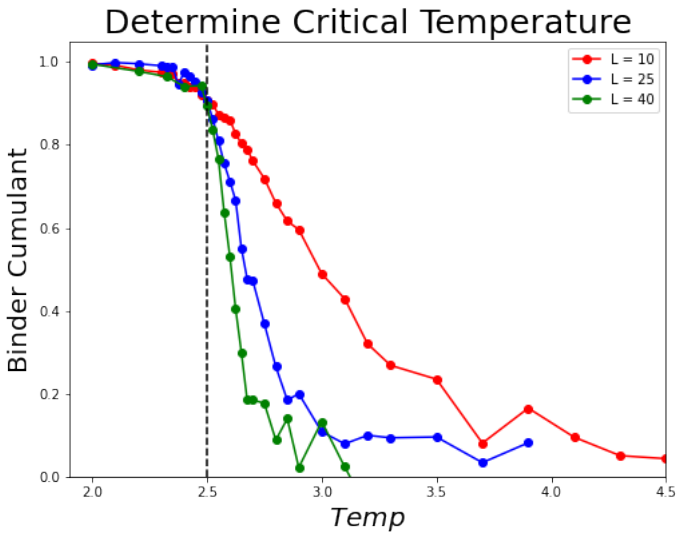


Figure 9: Binder Cumulant Value vs Temperature for different sizes (J = 1, K= 0.2). The cross point of the three curves is marked as the dashed line, poiting to T = 2.500

B Codes Description

The codes are available in the Github Repository. In this section, we will give some illustrations to the codes.

B.1 Folders

- In the codes folder, there are well wrapped "class" files or “function” files, which are called in the calculation tasks.

  1. Configuration.py: A class file to construct the modified 2D Ising model;
  2. Hamiltonian.py: A function file to calculate the Hamiltonian or effective Hamiltonian in the project;
  3. LocalUpdate.py: A class file to Complete local update algorithm based on Hastings-Metropolis method;
  4. WolffUpdate.py: A class file to implement Wolff cluster update algorithm for the system with original Hamiltonian;
  5. SelfLearningUpdate.py: A class file to implement Self-Learning Update Method with effective Hamiltonian and Wolff update algotithm;
  6. RestrictedSelfLearningUpdate.py: A class file to implement Self-Learning Update Method with cluster size restriction;
  7. SLMC_Training_Lib.py: A function file which will be used in the SLMC/RSLMC training process, including making samples, self-optimizing, etc.

  The four update classes will take spins configuration as an input parameter, and return the spins configuration after one update.

- In the testing folder, one will find some sub-folders which aims to realize one specific calculation task. Plotting and production files are strictly separated.

- In the data folder, calculation results (including plot files and maybe data files) are presented in this folder.

B.2 Calculation Functions

- **Physical_quantities_vs_Temp**

  To calculate how the physical quantities (which we are interested in, including energy, magnetization, specific heat capacity, and susceptibility) vary with temperature using the local update method and Wolff cluster Update method.

  The definition of specific heat capacity is $C_v = \beta N(\langle m^2\rangle - \langle|m|\rangle^2)$, while the definition of susceptibility is $\chi = k_B\beta^2 N(\langle e^2\rangle - \langle e\rangle^2)$. In all the calculation, we set $k_B = 1$.

  To run the file and get calculation results, one should put Physical_Quantities_vs_Temp_plotting.py and Physical_Quantities_vs_Temp_training.py in the codes folder and use the command:

```
1    >> python Physical_Quantities_vs_Temp_training.py
2    >> python Physical_Quantities_vs_Temp_plotting.py
```

  The training process will take some time, and one should be patient with Monte Carlo Simulation Calculations.

  Of course, one is welcome to change parameters set in the file and explore more about the model.

- **Binder Cumulant vs Temp**

  The definition of Binder cumulant is: $U(T,C) = \frac{3}{2}(1 - \frac{\langle M^4\rangle}{3\langle M^2\rangle})$. It can be used to determine the critical temperature of Ising model, as suggested in the literature.

  To calculate Binder cumulant vs temperature, one should first run "Binder_Wolff" file:

```
1    >> python Binder_Wolff.py
```

This file will produce some .dat files, which are used in the plotting. For convenience, we put our calculation results in the folder. If one doesn't want to spend a long time waiting for calculation, he/she can run directly:

```
>> python Binder_Wolff_plot.py
```

Then the graph will be saved in the same folder. One should note that you must put both .py files in the codes folder, and it is the same with the following tasks.

- **SLMC & RSLMC Training**

  To complete SLMC or RSLMC training and obtain optimized effective Hamiltonian at T = T_c, one should first generate some samples at T larger than T_c using Local Update Algorithm. Run with the command:

```
>> python Local_samples_training.py
```

  After a long time of samples production, one will have effective parameters (J as defined in the literature) written in the .dat files. In the file, we choose to generate Local Update samples of size = 10, 25, 40, 60, 80, however, one can have their own choice for this parameter. Correspondingly, one should change the "interval" value in the calculation. Because different configuration sizes have different correlation times.

  After that, one can run SLMC_training. There are two approaches for the training, as introduced in our report. One is "ensemble average", the other is "temperature descent". Usually, the temperature descent method will cost less time than the other.

  To run the first option, one uses the command:

```
>> python SLMC_Training_Ensemble_Average.py
```

  To run the second option, one uses the command:

```
>> python SLMC_Training_Temp_Descent.py
```

  Please note that in either case, the calculated result will be printed and displayed in your terminal instead of written into data files. Therefore, one is encouraged to save output results by yourself. This can be done easily, for example, in a Linux system, one uses the command:

```
>> python SLMC_Training_Ensemble_Average.py |tee
   result.txt
```

  Moreover, you have to do data dealing by writing a job script by yourself. For example, you have to calculate the average value of effective parameters in the ensemble average method.

  And the same procedure for RSLMC training.

  Our training process is done using only the first-order expansion, namely, we only keep J_1 in the effective Hamiltonian. If one wants to try third-order expansion, simply change this line in the file

```
eff_param = np.loadtxt('Local_data_fitting_eff_param(
   n=1)(L=%i).dat'%size[i])
```

  to the following:

```
eff_param = np.loadtxt('Local_data_fitting_eff_param(
   n=3)(L=%i).dat'%size[i])
```

Third-order Hamiltonian parameters will be read and training is done by keeping all these three parameters.

- **Acceptance probability**

  If one is interested in the flipping probability when construction the clusters in SLMC/RSLMC update method, run with the command:

```
python Acceptance_prob_SLMC_RSLMC.py
```

  In this file, we calculate the flipping probability for SLMC with sizes 25, 40, 60 and for RSLMC with sizes 25, 40, 60, 80, 100, 120. One is free to change the setting, however, correspondingly you have to change eff_param which is obtained from (R)SLMC training process and "restriction value" for RSLMC.

  The calculated result will be displayed in the terminal, you have to save it by yourself.

- **Autocorrelation function**

  The definition is autocorrelation function is given in the original literature. Here, we introduce a normalized coefficient, which will not change the value of correlation time. Let $O_i = |M_i| - \langle |M| \rangle$, and then autocorrelation function of $|M|$ is given by $C(\tau) = \frac{\frac{1}{N-t}\sum_i O_i O_{i+t}}{\langle O^2 \rangle} \propto e^{-\tau/\tau_{correl}}$.

  In our experiment, we find that the fitting correlation time varies if we only do one calculation of the autocorrelation function. Therefore, we choose to do a collection of calculations (for example, iter = 30) and take the average value of correlation time as the final result.

  To calculate autocorrelation function and correlation time, one run the command:

```
>> python Autocorrelation_Local.py
>> python Autocorrelation_Wolff.py
>> python Autocorrelation_SLMC.py
>> python Autocorrelation_RSLMC.py
```

  After completing the calculation, one opens the jupyter notebook file and runs each cell. We plot the Correlation Time vs Size and autocorrelation function for each update method.

To end this section, we want to express that one is always welcome to do more tests with the codes and give suggestions or feedback to us.