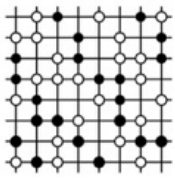
Report for final project 3

BY JINLONG HUANG

Jan 12 2017

In the final project 3, I used Metropolis Monte Carlo simulations to study the phase transition to a magnetized state occurring in a diluted 2D Ising model.

1. The description of diluted 2D Ising model:



In a diluted 2D Ising model, every site has three possible states: spin up, spin down and empty, corresponding to σ = {1, −1, 0}. It can describe two different species of atoms or molecules($\sigma_i = \pm 1$) partially covering a surface. The spins are coupled with a pair interaction $-J\sigma_i\sigma_j$. If the external magnetic field is zero, the hamiltonian H is zero too. So the energy is only interaction energy

$$E = -J \sum_{<i,j>} \sigma_i \sigma_j, \quad \sigma \in \{-1, 0, 1\} \tag{1}$$

We set $J = 1$(i.e., ferromagnetic inteeraction for spin pairs with $|\sigma_i| = |\sigma_j| = 1$). We consider the system as a diluted magnet where the sites with $\sigma_i = 0$ sites are *empty* and those with $\sigma_i = \pm 1$are *occupied* by magnetic moments(Ising spins).

2. Average magnetization as a function of time:

At low temperatures, the system is ferromagnetic. At sufficiently high temperatures, the magnetization of the system is zero. There is a critical value of the temperature called the *Curie temperature* at which a phase transition between the ferromagnetic and paramagnetic phases occurs.

The probability of a particular microstate is determined by *Boltzmann factor*:

$$P(s_1, s_2, \ldots, s_{N_s}) = \frac{e^{-E(s_1, s_2, \ldots, s_{N_s})/k_B T}}{\sum_{\text{configs}} e^{-E/K_B T}} \tag{2}$$

So the Average magnetization at some fixed temperature T is given by

$$<M> = \frac{\sum_{\text{configs}} M e^{-E/k_B T}}{\sum_{\text{configs}} e^{-E/K_B T}} \tag{3}$$

In my programs, I set $k_B = 1$.

3. Metropolis Monte Carlo for Ising model:

    a. The ordinary Metropolis MC algorithm:

Imagine a mountainous terrain on a d-dimensional plane, with P(x) representing the elevation of the surface at point **x**. A random walker is started at some point $\boldsymbol{x}_0$. The walker then steps successively from point to point and generates a sequence of points:

$$\begin{aligned} x_0 \;\rightarrow\; & x_{1_t} \rightarrow x_{2_t} \rightarrow \cdots \rightarrow x_{N_t} \rightarrow x_1 \rightarrow x_{1_2} \rightarrow \cdots \rightarrow x_{1_v} \\ & \rightarrow x_2 \rightarrow x_{2_2} \rightarrow \cdots \rightarrow x_{2_v} \\ & \rightarrow x_3 \rightarrow \cdots \rightarrow x_4 \rightarrow \cdots \\ & \rightarrow x_{N-1} \rightarrow \cdots \rightarrow x_N \end{aligned}$$

The first N_t steps is the thermalization steps, which can be viewed as being exploratory in nature to learn the highest mountains and peaks.

After thermalization, the generated points x_1, x_2, …, x_N are used in the Monte Carlo application. The internal points are discard so the sampling frequency $v \geqslant 1$.

At each step, supposed the walker is at some point x_{current}, the next point x_{next} is determined as follows:

- Choose a trial point x_{trial} at random in a nerghborhood of x_{current} with maximum step size δ.
- Calculate ratio $r = P(x_{\text{trial}})/P(x_{\text{current}})$ and generate a uniform random number n_r in [0, 1].
- Accept the trial point only when $r \geqslant n_r$.

Efficient Evaluation of Boltzmann Factors: Only consider four nearest interaction:$s_i = \pm 1$, $s_i \sum_{\text{neighors}\, j} s_j = -4, -2, 0, 2, 4$.

b. Metropolis MC for ordinary 2D Ising model:

- Generate an initial state *i*.
- Attempt to change the configuration from *i* to *j*.
- Compute $\Delta E = E_j(s_1, s_2, \ldots, s_{i,\text{trial}}, \ldots, s_{N_s}) - E_i(s_1, s_2, \ldots, s_i, \ldots, s_{N_s})$.
- If $\Delta E \leqslant 0$, accept the change and replace state *i* by state *j*. Go to 2.
- If $\Delta E > 0$, generate a uniform random number nr $\in$ [0, 1]. Calculate $w = \exp(-\Delta E/K_B T)$.
- If $r < w$, accpt the change and replace state *i* by state *j*. Go to 2.
- If $r > w$, then reject the change. Go to 2.

c. Metropolis MC for diluted 2D Ising model:

- Generate an initial state *i*.
- Attempt to change the configuration from *i* to *j* or k.
- Compute $E_i(s_1, s_2, \ldots, s_i, \ldots, s_{N_s})$, $E_j(s_1, s_2, \ldots, s_j, \ldots, s_{N_s})$ and $E_k(s_1, s_2, \ldots, s_k, \ldots, s_{N_s})$.
- $w = \exp(-E/k_B T)$. Compute w_i, w_j, w_k. maxw = max(w_j, w_k).
- If maxw $\geqslant 0$, accept the change and replace state *i* by state *j*. Go to 2.
- If maxw < 0, generate a uniform random number nr $\in$ [0, 1].

- If $r <$ maxw, accpt the change and replace state *i* by state *j*. Go to 2.
- If $r >$ maxw, then reject the change. Go to 2.

4. Two programs: ising2d.f90 and dIsing2d.f90

   My programs are based on the program ising2d.f90 developed by Professor Li Huang, but I added some new features and removed some functionality that I didn't need to use.

   a. ising2d.f90

      ising2d.f90 simulated the magnetization states of a 2 dimentional Ising model. It has a module ISING which is later imported in main program ising2d. Module ISING has four subroutines: initialize, spin_flip, calculate_properities and output.

      i. Subroutines "initialize" defines all necessary variable which will be used later on. It reads four values from ising_2d_input.dat: nt, tmax, dt and steps. nt is the number of temperature, tmax the highest temperature, dt the temperature spacing. Steps is the number of Monte Carlo steps. I didn't use the variable *bins* because I think by increasing the number of steps of Monte Carlo the error can be limited in a small range. Then I define a one dimentional array $pflip(i) = \exp(-i \times 2.0/\text{temperature})$ as the probability of flippint a state. Finally I assign every spin as a random integer -1 or 1.

      ii. Subroutine spin_flip(t) requires a input parameter t to identify the current temperature. It picks up a spin randomly and calculate spin_sum for four nearest spin around it. What's more, it computes the ratio between probability of flipped state and probability unflipped state. If the ratio is larger than a random number bewtween 0 and 1, the spin changes state.

      iii. Subroutines calculate_properties and output compute the total magnetization and average magnetizaiton respectively.

      iv. The main program ising2d.f90 has three sub moves:

         - Initialize all variables
         - Thermalize the model by flipping spins N_d times, where N_d stands for the total number of spins.
         - Calculate the average magnetizaiton from highest temperature to lowest temperature. For each temperature, it flips spins (the number of Monte Carlo * the number of Metropolis) times and adds all magnetization. After adding all magnetization, it computes the average magnetization and outputs temperature and average magnetization to ising_2d_output.dat.

   b. dIsing2d.f90

      The main structure of dIsing2d.f90 is the same as ising2d.f90. Only two main parts of it changed.

      i. Assign every spin as {-1, 0, 1}, not {-1, 1} by spin(i,j)=int(3.d0*nr)-1.

      ii. The Metropolis step:

         Assume the spin change state i to state j or k,

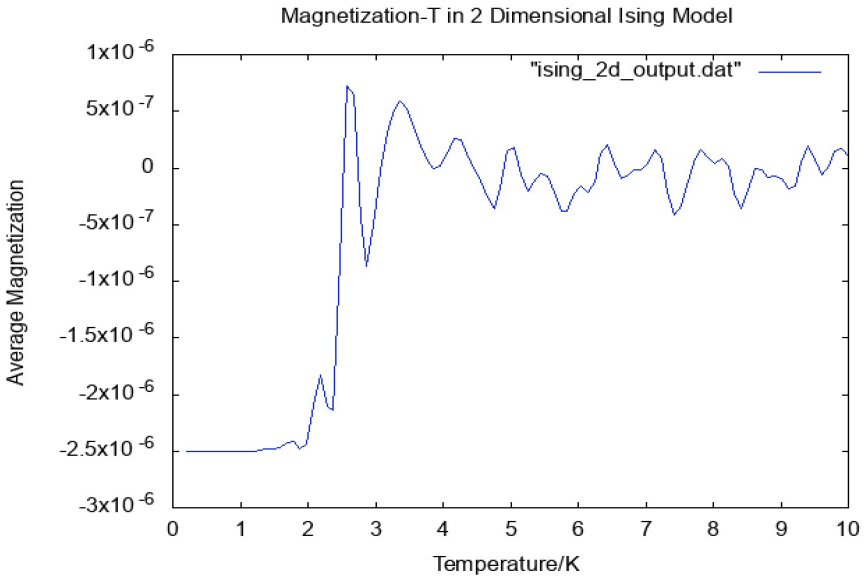
         - Compare the probability of state j and k using pflip(- j*spin_sum) and pflip(- k*spin_sum). If P(j) is larger.

- If $\frac{P(j)}{P(i)} \geqslant$ nr, change state *i* to state *j*.
- Else, no change.

5. Results and analysis:

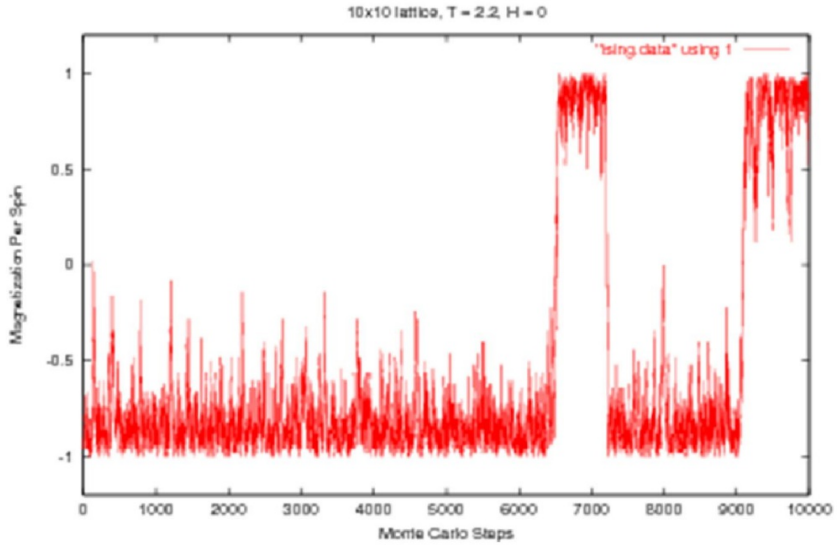
   i. Normal 2D Ising model:

   Setting nt = 50, tmax = 10.0, dt = 0.2 and steps = 1000



From the graph we can see that when the temperature is larger than 3K, the average magnetization fluctuated around 0, which means in this range of temperature the model didn't show any magnetization. When the temperature decrease to 2K, the average magnetization decrease dramatically, and stay stable when temperature is lower than 2K. So the *Curie temperature* is about 2.5K.

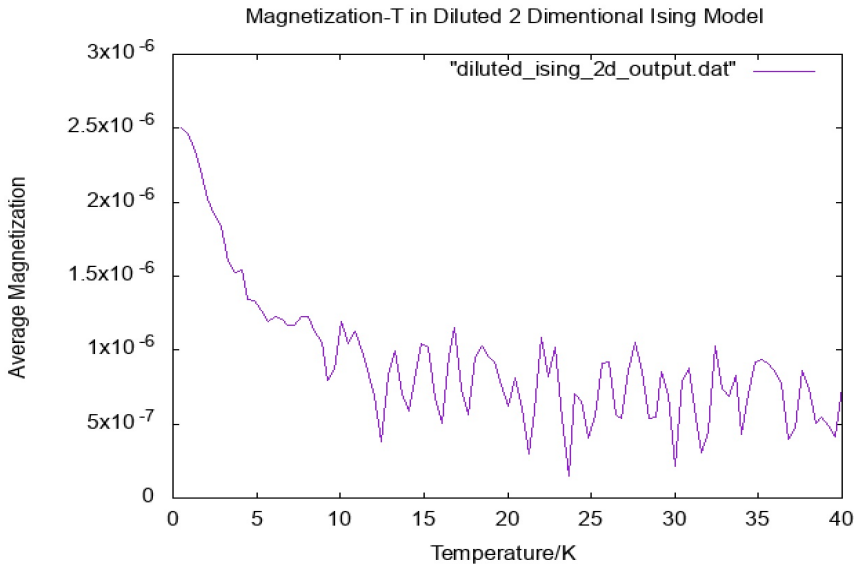
The fact that the average magnetization is negative when T<2K is consistent with results in this graph:



The graph depicts that when the number of Monte Carlo is set to 1000, the magnetization per spin is negative.

ii. Diluted 2D Ising model:

Setting nt = 80, tmax = 40.0, dt = 0.5 and steps = 1000



As the graph tells us that 1) there is no dramatically change of average magnetization when temperature decrease from 40K. The average magnetization increase gradually from T = 10K to T = 0K, which is what we expect in the diluted 2D Ising model. When the 2D Ising model is diluted, which means the system now can occupy much more states with expanded energy range, the change from no magnetization to magnetization is spanned in a larger range of temperature. 2) The diluted model has the same average magnetization as the normal one when $T \rightarrow 0K$.

But there are still two problems this graph reveals to us. 1) The average magnetization when T<10K is positive. 2)The average magnetization when T>10K doesn't fluctuate about 0. The first problem maybe tell us the new model has different positive-negative dependence on steps of Monte Carlo. The second problem maybe comes from a fixed round-off error.

6. Reflection:

1. Before I actually developed my own program, I go through every pieces of professor Li Huang 's program ising2d.f90. After I understood all the details in that program, I began to modify and add my own code into it. I believe this step learning from the existed code save me significant mount of time.

2. There are three important parameters that need to be careful when design a Monte Carlo program: step size for T, sampling frequency and the number of Monte Carlo. Using different those paramenters can result in a totally different output and graph. If the step size is too small, then most steps will be accepted no matter what the probability function is. If the step size is too large, too many steps will be rejected, and the walker will find it difficult to explore. The sampling frequency and the number of Monte Carlo decide the trade of between accuracy and speed.

3. The reject criteria I uesd in dIsing2d.f90 only involves the state with larger probability. There are also other reject criteria that can be used of course. For example, reject to change state even when the smaller probability ratio is less than the random number nr. I choose this criteria because I think this principle can result in a faster convergence to a thermal state, while still remain some probability to explore other mountains.

4. During the development of these programs, I stick to the principle that “testing, not debugging”. I added “print” statements into every important parts in these programs, sometimes a lot of them, and tested the program part by part. It works pretty well. A huge mount of time for debugging were saved. Now I feel I'm more professional at programming in fortran than I was.