



# Tutorial: Monte Carlo Simulation of the 2D Ising Model

August 26, 2019

In this tutorial, we will study the phase transition in the classical two-dimensional Ising model, with Hamiltonian

$$H = -J \sum_{\langle ij \rangle} \sigma_i \sigma_j,$$

where  $\sigma_i = \pm 1$ ,  $J$  is the coupling strength and  $\sum_{\langle ij \rangle}$  denotes a sum over nearest neighbours. We will consider simulations on a square lattice with periodic boundaries. In the thermodynamic limit, the critical temperature is known to be  $T_c/J \approx 2.269$ .

We will use and modify the two Python programs `ising_mc.py` and `plot_ising.py` throughout this tutorial in order to implement Monte Carlo (MC) methods that estimate  $T_c$  and compare with this known exact solution.

## 1 Monte Carlo algorithm

Consider the Monte Carlo program `ising_mc.py`, which is designed to perform a Monte Carlo simulation (using the single-spin-flip Metropolis algorithm) and record measurements of the system's energy  $E$  and magnetization  $M$ .

- a) Examine the section of the code that computes the two-dimensional `neighbours` array, which is used when calculating the system's energy. The code is already written such that `neighbours[i,0]` and `neighbours[i,1]` store the lattice location of spin  $i$ 's rightward and upward neighbours, respectively. Modify the code such that it will also store spin  $i$ 's leftward neighbour in `neighbours[i,2]` and its downward neighbour in `neighbours[i,3]`.

**Hint:** Don't forget to consider the periodic boundary conditions.

- b) Examine the `sweep()` function, which proposes a number `N_spins` single spin-flip Monte Carlo updates. Convince yourself that this code is implementing the single-spin-flip Metropolis algorithm.
- c) Implement a more efficient way of calculating the energy difference `deltaE` within the `sweep()` function. The given implementation calculates this energy difference by using the `getEnergy()` function, which involves iterating a loop `N_spins` times. However, you should be able to calculate `deltaE` by summing only four terms.

- d) Run your code with `n_eqSweeps=1000` and `n_measSweeps=10000`. The code will generate files in a directory called `Data` that will store the energy and magnetization corresponding to each of your sampled system configurations. (In Question 2, we will analyze and plot the resulting data.)

**Hint:** Set `animate = False` for this part so that the code runs faster.

## 2 Estimating the critical temperature

Recall from lecture that the specific heat  $C_V$  and susceptibility  $\chi$  can be expressed as

$$C_V = \frac{\langle E^2 \rangle - \langle E \rangle^2}{T^2}, \quad \chi = \frac{\langle M^2 \rangle - \langle M \rangle^2}{T},$$

where  $E$  is the energy and  $M = \sum_i \sigma_i$  is the magnetization. For our Monte Carlo calculations on finite lattices, there is no spontaneous symmetry breaking and therefore  $\langle M \rangle = 0$  at all  $T$ . As a result, we instead examine  $\langle |M| \rangle$  and calculate the susceptibility as

$$\chi = \frac{\langle M^2 \rangle - \langle |M| \rangle^2}{T}.$$

The quantities  $C_V/N$  versus  $T$  and  $\chi/N$  versus  $T$  both diverge at the critical temperature  $T_c$  in the thermodynamic limit  $N \rightarrow \infty$ . On a finite lattice, these quantities do not diverge but will acquire peaks near  $T_c$ .

- Use the code `plot_ising.py` to read in the Monte Carlo data from Question 1e and plot the estimators for  $\langle E \rangle/N$  and  $\langle |M| \rangle/N$ . Consider the values you find for these estimators in the limit of small  $T$ ; do they match your theoretical expectations?
- Modify `plot_ising.py` to calculate  $C_V$  and  $\chi$ . Plot  $C_V/N$  and  $\chi/N$  versus  $T$  and verify that there are peaks in these quantities near  $T_c$ .
- Use `ising_mc.py` to generate additional data for higher  $L$  and for more temperatures close to  $T_c$ . Modify `plot_ising.py` to plot your results for several different values of  $L$  and confirm that the peaks in  $C_V/N$  and  $\chi/N$  get closer to  $T_c$  as  $L$  increases.