ELEC5210 Midterm Project
2D Ising model with Monte Carlo simulation

name, SID

1 Ising Model Theory

Ising model, named after Ernst Ising who first investigated and solved the one-dimensional Ising model[1], is a physical model that generally describes a lattice system for spin-1/2 (any two-state system) particles with only two states for each particle: spin up and spin down. We can use $+1$ and -1 to denote the two states. There is interaction between neighboring particles, e.g. the four particles on the left, right, above and below one specific particle. The particles can also be subject to an external magnetic filed. The most common Ising model is two-dimensional square lattice system with zero B filed as it can be analytically solved and easy for numerical simulation. Ising model is the simplest physical model that has phase transition[2].

1.1 2D square lattice Ising model

For a 2D square lattice Ising model, assume N by N particles are aligned in a square lattice. The interaction strength between each pair of neighboring particles S_i and S_i is J_{ij}, and the interaction strength between one particle S_i and the external B filed is B_i. We can write down the Hamiltonian for the system[3]:

$$H = -\sum_{i,j} J_{ij} S_i S_j - \sum_i B_i S_i \tag{1}$$

Generally, we consider the Ising model with zero B filed, so equation (1) is reduced to a more common format:

$$H = -\sum_{i,j} J_{ij} S_i S_j \tag{2}$$

Here the interaction parameter J_{ij} can be positive (corresponding to ferromagnetic interaction) or negative (corresponding to antiferromagnetic interaction). Since the Hamiltonian corresponds to the energy which is supposed to be minimized for a stable system. For simplicity we can set $J_{ij} = \pm 1$ meaning the interactions between neighboring particles are uniform and for ferromagnetic system we take $J = 1$. Hence the Hamiltonian we will investigate below is finally:

$$H = -\sum_{i,j} S_i S_j \tag{3}$$

For ferromagnetic system, when the temperature is low, the particles tend to have identical spins to lower the total energy; when the temperature is gradually increased, small fluctuations will happen and some noises of the spin alignment will emerge in the system; when the temperature is high enough, the fluctuations due to the thermal movement becomes dominant and finally the system is randomized. There is a phase transition when the temperature is increasing and the temperature for the phase transition is called critical temperature T_c.

1.2 Onsager's solution

Onsager developed the analytical solution of the Helmholtz free energy F for 2D Ising model[4]:

$$-\beta F = \ln(2) + \frac{1}{8\pi^2}\int_0^{2\pi}\int_0^{2\pi} \ln\left[\cosh^2(2\beta) - \sinh(2\beta)\cos(\theta_1) - \sinh(2\beta)\cos(\theta_2)\right] \mathrm{d}\theta_1 \mathrm{d}\theta_2 \quad (4)$$

here we already take $J = 1$ and $\beta = \frac{1}{k_B T}$ where $k_B = 1$ for simplicity. From Onsager's solution, we can obtain various thermal dynamics parameters among which the most important one is the critical temperature:

$$T_c = \frac{2}{\ln(1+\sqrt{2})} \approx 2.269 \quad (5)$$

To investigate the characteristics of a system, a traditional thermal dynamics method is to calculate the partition function of the system, and then obtain every thermal dynamics characteristics from the partition function. But this method is not suitable for large scale system as the calculation of the partition function requires knowing microscopic configurations, which is nearly impossible for human since the system size is growing exponentially. Therefore, we seek the help from computers and thus the Monte Carlo method is introduced.

2 Monte Carlo

Monte Carlo method, named after a casino in Monaco, is a statistical method using probability (especially with random numbers) to simulate and solve real world problem[5]. The basic principle of Monte Carlo method is to generate random numbers for a specific state of a system, and then calculate certain values based on the randomly generated number, and this update is accepted or rejected based on some criteria. Here we have a simple example of estimating π using Monte Carlo method.

2.1 An example for estimating π

Within a 1×1 square area, N points with coordinates (x_i, y_i) are generated randomly. Then the distance between each point and the origin point is calculated and compared with 1. If the distance is smaller than 1, it means the point falls inside the quarter of circle area and if larger, the point falls outside the circle. We count the number of points falling inside of the circle (n), and when N goes to considerably large, we can extract π from the estimated area: $\lim_{N\to\infty} \frac{n}{N} = \frac{\pi}{4}$, thus $\pi \approx 4\frac{n}{N}$ when $N \to \infty$.

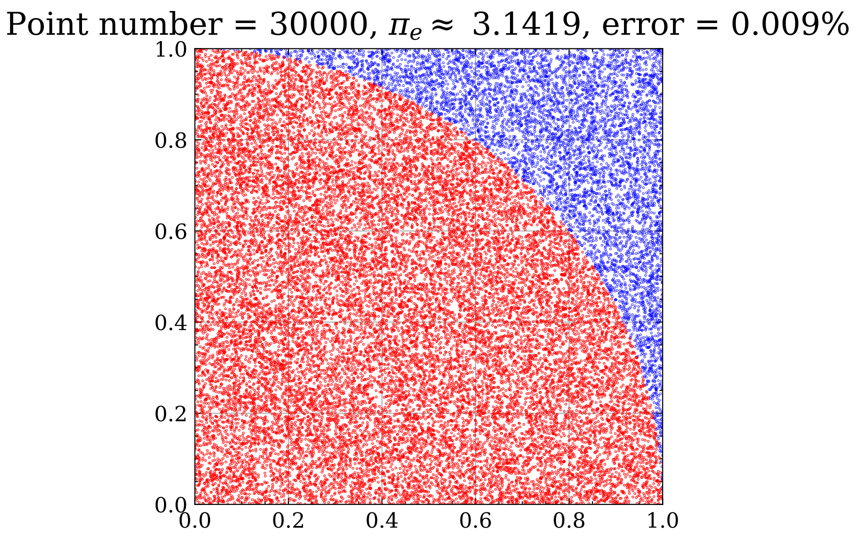

Figure 1: Monte Carlo estimation of π

Fig (1) illustrates the Monte Carlo simulation result of $N = 30000$, and the estimated π_e is very close to the real value.

2.2 Implementation on 2D Ising model

Consider a $N \times N$ lattice with total particle number N^2. Each particle has the spin of $+1$ or -1. We can calculate the interaction energy of each particle by the summation of its neighboring four particles, *i.e.*

$$E_{i,j} = -S_{i,j}(S_{i-1,j} + S_{i+1,j} + S_{i,j-1} + S_{i,j+1}) \tag{6}$$

Then we add up all the interaction energy for single particle to get the total energy.

2.2.1 Initialization

At first the system can be initialized into a uniform state meaning all the particles have the same spin. From thermal dynamics we know the microscopic state depends on system temperature. The higher the temperature is, the more random the system is. Thus we can throw a random number at one certain temperature T to flip a particle's spin and to see whether the total energy drops or not. Since the total energy is supposed to be minimized, if the total energy of the new state is lower than the older state, the new state is accepted; but if the new energy is higher than the older one, instead of simply rejecting it, we apply the Metropolis method[6].

2.2.2 Metropolis method

When the total energy of the new system is higher than that of the older one (assuming the energy difference is ΔE), we accept it with the probability $e^{-\beta\Delta E = e^{-\frac{\Delta E}{k_B T}}} = e^{-\frac{\Delta E}{T}}$. By "with the probability of $e^{-\Delta E}$" we mean generating a random number within $[0, 1)$ and compare it with $e^{-\Delta E}$; if the generated random number is smaller than $e^{-\Delta E}$ we accept this state.

2.2.3 Periodic boundary conditions

We cannot simulate real world crystal as the computation resources for simulation is finite so we must cut off at the edge of the square lattice. The particles inside the square lattice has four interacting neighbor particles, however the particles at the edge have only three or even two neighboring particles. Thus we need to apply periodic boundary conditions (PBC) for boundary particles. When the calculation happens on the boundary particles, we "borrow" one or two particles from the opposite border and treat them as the neighboring particles of the boundary particles in order to complete the calculation.
When we have completed all the configurations of the Ising model, we can do the simulation and extract some thermal dynamics parameters from the model, as listed below:

$$\langle E \rangle = \frac{1}{N^2} \sum_i E_i \tag{7}$$

$$\langle S \rangle = \frac{1}{N^2} \ln(\Omega) = \frac{1}{N^2} \ln(\mathrm{C}_{N^2}^{n+1}) \tag{8}$$

$$\langle C \rangle = \frac{1}{T^2} \left(\langle E^2 \rangle - \langle E \rangle^2 \right) \tag{9}$$

$$\langle M \rangle = \frac{1}{N^2} \sum_i S_i \tag{10}$$

here the Ω is the number of possible microscopic states and it can be calculated by the combination number of total particles number and number of spin up particles.

3 Simulation Results

Fig (2) shows the Ising model with a random initial state at different temperature. Noted that at low temperature, the system tend to be aligned at the same direction and several domain emerge. At the critical temperature, the fluctuations of the system is huge and the whole system can no longer remain ordered after iteration. When temperature is high, the system is completely in disordered phase.

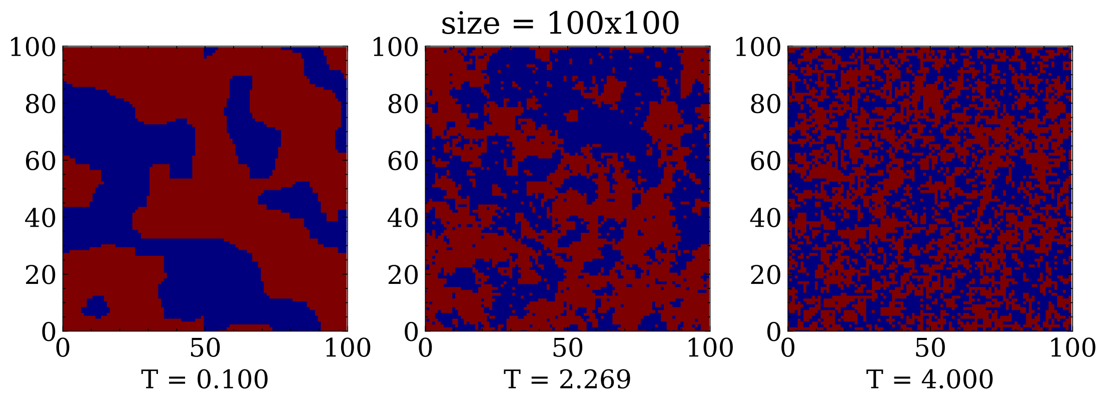

Figure 2: Ising model with random initial state

From Fig (3), we can verify the results further. The energy, entropy, specific heat and magnetization are plotted separately. We can see clearly the phase transition happens on the critical temperature T_c.

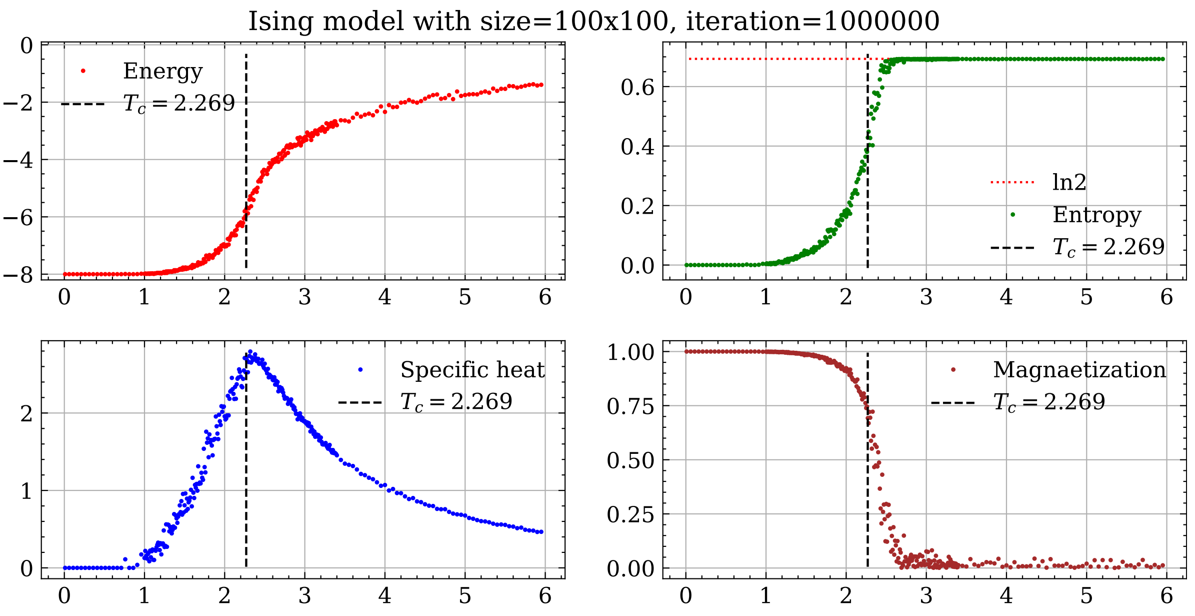

Figure 3: Ising model simulation results with uniform initial state

References

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

[2] Giovanni Gallavotti. *Statistical mechanics: A short treatise.* Springer Science & Business Media, 2013.

[3] Wikipedia contributors. Ising model. `https://en.wikipedia.org/wiki/Ising_model`, 2021. [Online; accessed 14-March-2021].

[4] Lars Onsager. Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Phys. Rev.*, 65:117–149, Feb 1944.

[5] Wikipedia contributors. Monte carlo method. `https://en.wikipedia.org/wiki/Monte_Carlo_method`, 2021. [Online; accessed 15-March-2021].

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

A Appendix: Codes

A.1 Monte Carlo estimation of π

```python
def mcpi(N=1000):
    points = np.random.uniform(0,1,(2,N))
    pointsInsideIndex = np.where(np.sum(np.square(points), axis=0) <= 1)[0]
    piEstimated = pointsInsideIndex.size / N * 4.0
    return points, pointsInsideIndex, piEstimated, (piEstimated-np.pi)/np.pi
```

A.2 Monte Carlo simulation of 2D Ising model[1]

```python
class IsingLattice:
    def __init__(self, temperature, initial_state, size):
        self.size = size
        self.T = temperature
        self.system = self._build_system(initial_state)

    @property
    def sqr_size(self):
        return (self.size, self.size)

    def _build_system(self, initial_state):
        if initial_state == 'r':
            system = np.random.choice([-1, 1], self.sqr_size)
        elif initial_state == 'u':
            system = np.ones(self.sqr_size)
        else:
            raise ValueError("Initial state must be 'r' or 'u'")

        return system

    def _bc(self, i):
        if i >= self.size:
            return 0
        if i < 0:
            return self.size - 1
        else:
            return i

    def energy(self, N, M):
        return -2*self.system[N, M]*(
            self.system[self._bc(N - 1), M] + self.system[self._bc(N + 1), M]
            + self.system[N, self._bc(M - 1)] + self.system[N, self._bc(M +
            ↪  1)]
        )

```

[1]This code is updated from this GitHub repo: `https://github.com/bdhammel/ising-model`

```python
    @property
    def internal_energy(self):
        e = 0
        E = 0
        E_2 = 0
        for i in range(self.size):
            for j in range(self.size):
                e = self.energy(i, j)
                E += e
                E_2 += e**2
        U = (1./self.size**2)*E
        U_2 = (1./self.size**2)*E_2
        return U, U_2

    @property
    def heat_capacity(self):
        U, U_2 = self.internal_energy
        return (U_2 - U**2)/(self.T)**2

    @property
    def entropy(self):
        N = self.size**2
        Np = int((N+np.sum(self.system))/2)
        return math.log(comb(N, Np, exact=True))/N

    @property
    def magnetization(self):
        return np.abs(np.sum(self.system)/self.size**2)


def run(lattice, epochs):
    for epoch in range(epochs):
        N, M = np.random.randint(0, lattice.size, 2)
        E = -1*lattice.energy(N, M)
        if E <= 0.:
            lattice.system[N, M] *= -1
        elif np.exp(-E/lattice.T) > np.random.rand():
            lattice.system[N, M] *= -1
```

A.3 Plotting simulation data

```python
def plotData(data, fileName):
    axName = ["Energy","Entropy","Specific heat","Magnaetization"]
    colors = ['red', 'green', 'blue', 'brown']
    fig, ax = pl.subplots(nrows=2, ncols=2, figsize=[8,5], dpi=300)
    fig.tight_layout(rect=[0, 0, 1, 0.98])
    ax = ax.flatten()
    fig.suptitle("Ising model with size=%dx%d, iteration=%d" %(size, size,
    ↪  iteration))
    ax[0].set_ylim([-8.2,0.1])
    ax[1].set_ylim([-0.05,0.75])
    ax[1].axhline(y=np.log(2), xmin=0.05, xmax=0.95, linestyle=':',
    ↪  color='red', label="ln2")
    for i in range(0, 4):
        ax[i].plot(data[0], data[i+1], '.', color=colors[i], markersize=2,
        ↪  label=axName[i])
        ax[i].axvline(x=2.0/np.log(1+np.sqrt(2)), ymin=0.05, ymax=0.95,
        ↪  color='black', linestyle='--', label='$T_c=%.3f$'
        ↪  %(2.0/np.log(1+np.sqrt(2))))
        ax[i].grid()
        ax[i].legend(loc='best', fontsize=10)
    # fig.show()
    figName = fileName+".pdf"
    fig.savefig(figName, bbox_inches='tight', pad_inches=0)
    pl.close()
```