Ising Model

Vasilis Tsioulos*

*Project 4: Computational mathematics II 2025

ABSTRACT

This study investigates the 2D Ising model, which is a classical model for a binary magnetic material. In order to simulate the problem, a Monte Carlo method is used, more specifically the Metropolis-Hastings algorithm, which is more than enough to study physical systems. The project itself is a challenging but rewarding problem, giving a spherical understanding about how Monte Carlo simulations work.

1 Introduction

1.1 2D Ising Model

The Ising model[1,2], is a mathematical model of ferromagnetism in statistical mechanics. The model that is going to be simulated is a 2D version of the model, were spins exist on a lattice with accepted values [-1, 1] and a Hamiltonian:

$$H = -J\sum_{\langle ij\rangle}\sigma_i\sigma_j - h\sum_j \sigma_j \tag{1}$$

where the first sum is over nearest neighbors, J is the interaction strength, h is the external magnetic field, and $\sigma_i\sigma_j$ are the spins.

For visualization[3], the spin lattice would look like:

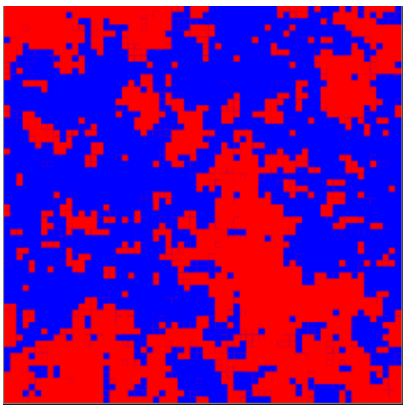

Figure 1. 2D Ising Model spin lattice

where the blue colored boxes have $spin = 1$ and red $spin = -1$

The second part of the Hamiltonian, the interaction term, is a term that based on the neighboring spins. This interaction arises because every spin has its own magnetic field due to the magnetic dipoles, and the only ones that can feel the magnetic field are its neighbors. Although the problem itself is hard to simulate, the Monte Carlo methods give a innovative way of simulating physics problems.

1.2 Monte Carlo Metropolis-Hastings

In order to solve the problem the Monte Carlo Metropolis-Hastings is used. In a system we define a point X of R^n space and a probability function $K(X|Y)$ describes how the system behaves. A condition which allows the system to achieve balance is called detailed balance. Every state of X, Y is descibed by the function:

$$\frac{A(X|Y)}{A(Y|X)} = \frac{f(X)Q(Y|X)}{F(Y)Q(X|Y)} \tag{2}$$

In Monte Carlo simulations we look for a convenient law of transient, where when the equilibrium is achieved, we have the required distribution $f(x)$.

2 Computational analysis

The process begins by initializing an $N \times N$ lattice. Next, the system undergoes thermalization, during which the temperature is gradually increased. As the temperature rises, the spins become disoriented and are measured. This disorientation leads to a weakening of the magnetic field. Once the temperature decreases again, the spins begin to align, resulting in a stabilization of the magnetic field. The variables were the size of the lattice N, the external field h, the interaction strength J, the temperature T and the number of sweeps. The first three function were initializing functions were they contain the calculation of energy, the metropolis sweep and the run of the simulation:

```python
@njit
def calculate_energy(lattice, J, h, N):
    energy = 0.0
    for i in range(N):
        for j in range(N):
            spin = lattice[i, j]
            neighbors = (
                lattice[(i+1)%N, j] +
                lattice[(i-1)%N, j] +
                lattice[i, (j+1)%N] +
                lattice[i, (j-1)%N]
            )
            energy += -J * spin * neighbors
            energy += -h * spin
    return energy / 2.0  # avoid double counting
```

```python

@njit
def metropolis_sweep(lat, J, h, T, N):
    for _ in range(N*N):
        i = np.random.randint(0, N)
        j = np.random.randint(0, N)

        spin = lat[i, j]
        neighbors = (
            lat[(i+1)%N, j] +
            lat[(i-1)%N, j] +
            lat[i, (j+1)%N] +
            lat[i, (j-1)%N]
        )

        delta_E = 2 * spin * (J * neighbors + h)

        if delta_E <= 0 or np.random.rand() < np.exp(-delta_E / T):
            lat[i, j] *= -1

@njit
def run_simulation(lat, sweeps, J, h, T, N):
    magnet = np.zeros(sweeps)
    energies = np.zeros(sweeps)

    for sweep in range(sweeps):
        metropolis_sweep(lat, J, h, T, N)
        energies[sweep] = calculate_energy(lat, J, h, N)
        magnet[sweep] = np.mean(lat)

    return energies, magnet
```

Listing 1. Initialization functions

The next important step is to simulate the ising model, do the thermalization and measure the magnetic field and the energy.

```
\begin{lstlisting}[style=pythonstyle, caption={Initialization functions}]
N = 100      # lattice size
h = 0         # external field
J = 1         # Interaction strength (ferromagnetic)
```

```
T = 2.5       # Temperature
energy = 0.0 # Energy
sweeps = 1000

def initialize_lattice(N):
    return np.random.choice([-1, 1], size=(N, N))

def simulate_ising(N, T, h, sweeps=1000):
    J = 1.0
    lat = initialize_lattice(N)

    # Thermalization
    for _ in range(sweeps):
        metropolis_sweep(lat, J, h, T, N)

    # Measurement
    energies, magnet = run_simulation(lat, sweeps, J, h, T, N)

    return np.mean(energies), np.std(energies), \
           np.mean(magnet), np.std(magnet), lat

def temperature_sweep(N, h, T_values, sweeps=1000):
    avg_E, std_E, avg_M, std_M = [], [], [], []

    for T in T_values:
        E_mean, E_std, M_mean, M_std, _ = simulate_ising(N, T, h, sweeps)
        avg_E.append(E_mean / (N * N))
        std_E.append(E_std / (N * N))
        avg_M.append(M_mean)
        std_M.append(M_std)
        print(f"T={T:.2f}  E={E_mean:.2f}  M={M_mean:.2f}")

    return np.array(avg_E), np.array(std_E), np.array(avg_M),
   np.array(std_M)

# Run the sweep
Ts = np.linspace(0.1, 3.0, 50)
E, E_err, M, M_err = temperature_sweep(N=N, h=0, T_values=Ts, sweeps=1000)

```

```
# Plotting
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.errorbar(Ts, E, yerr=E_err, fmt='o-', capsize=3)
plt.xlabel("Temperature T")
plt.ylabel("Energy per spin")
plt.title("Energy vs Temperature")

plt.subplot(1, 2, 2)
plt.errorbar(Ts, M, yerr=M_err, fmt='o-', capsize=3)
plt.xlabel("Temperature T")
plt.ylabel("Magnetization")
plt.title("Magnetization vs Temperature")

plt.tight_layout()
plt.show()
```

Listing 2. Initialization functions

Table 1. Energy (E) and Magnetization (M) at different Temperatures (T)

T	E	M
0.10	-19585.98	0.26
0.16	-19572.08	0.04
0.22	-19576.87	-0.20
0.28	-19992.20	-1.00
0.34	-19877.68	-0.92
...	...	...
2.88	-8683.37	-0.01
2.94	-8426.28	0.01
3.00	-8181.63	-0.01

In fig. 2 we can observe the energy rising exponentially with the temperature. When the temperature rises, the magnetization is unbalanced, until it hits the critical temperature, so the dipoles are disoriented and the magnetization is approaching zero.

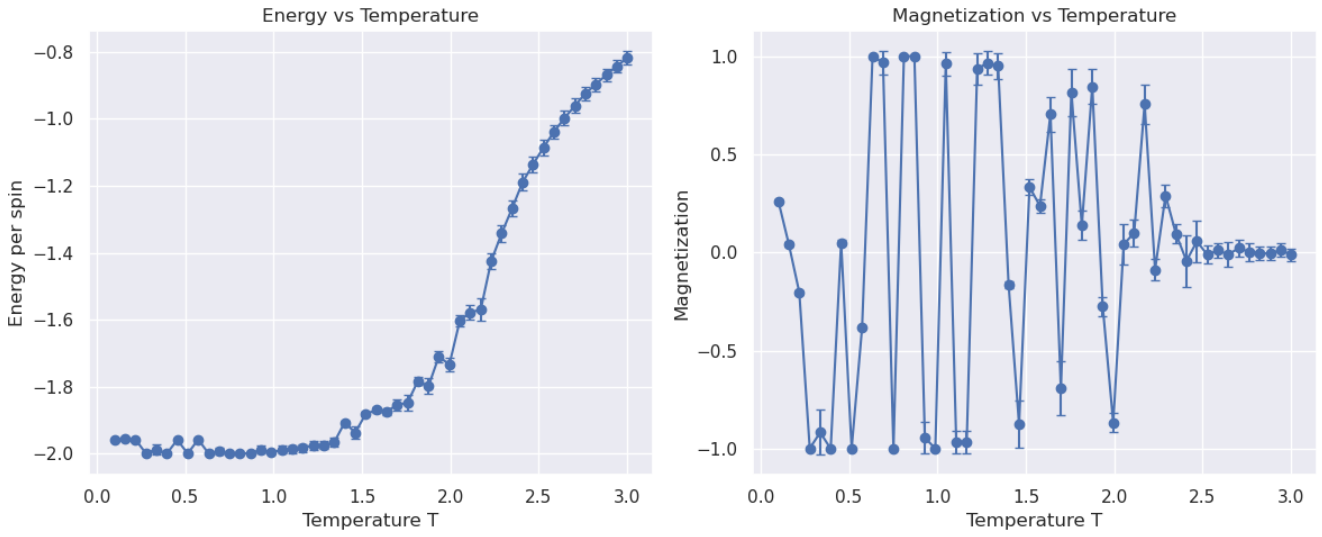

Figure 2. Energy and Magnetization vs temperature

By varying the external field, it is observed (as shown in the fig. 3) that the magnetization does not approach zero. However, the stronger the external field, the greater the resulting magnetization.

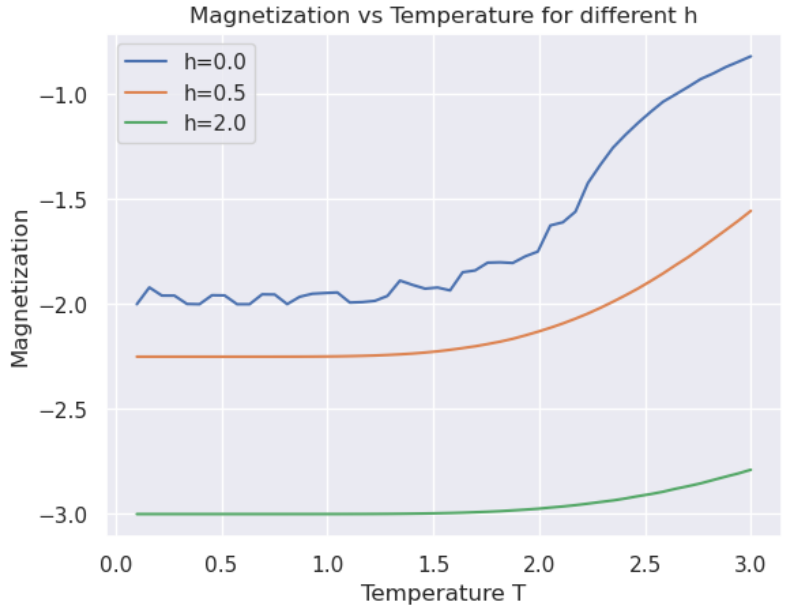

Figure 3. Magnetization vs temperature with different external fields

References

1. Kosmidis, K. Computational mathematics ii notes.
2. Jarzynski, J. Introduction to statistical mechanics – the ising model (n.d.). stanford.edu.
3. plius, M. Ising model (n.d.). IsingModel.