XY Model and Monte Carlo Simulation of Hysteresis

Francisco Moura

1 Introduction

The XY model is a simple model of spins on a lattice. Each site i has a spin represented by an angle $\theta_i \in [0, 2\pi)$. The interaction energy between neighbors favors alignment of spins. We also include an external magnetic field H in the x-direction.

2 Hamiltonian

The Hamiltonian of the XY model in two dimensions is

$$E = -J \sum_{\langle i,j \rangle} \cos(\theta_i - \theta_j) - H \sum_i \cos(\theta_i),$$

where $J > 0$ favors ferromagnetic order, $\langle i, j \rangle$ denotes nearest neighbors, and H is the external field.

3 Monte Carlo Method

We use the Metropolis algorithm to simulate thermal fluctuations at temperature T.

- Choose a random site (i, j).
- Propose a new angle $\theta' = \theta + \delta$ with small random step.
- Compute the energy change $\Delta E = E' - E$.
- Accept the change if $\Delta E \leq 0$ or with probability $\exp(-\Delta E/T)$ if $\Delta E > 0$.

Repeating this procedure many times allows the system to explore the configuration space.

4 Magnetization

The average magnetization per site is

$$M_x = \frac{1}{N} \sum_i \cos(\theta_i), \quad M_y = \frac{1}{N} \sum_i \sin(\theta_i).$$

In this tutorial we study M_x, the component along the field.

5 Hysteresis Curve

To compute hysteresis, we vary the external field H cyclically, for example as

$$H(t) = H_{\max} \sin(\omega t).$$

At each step we perform Monte Carlo sweeps and measure the magnetization M_x. Plotting M_x versus H gives the hysteresis loop. At low temperature the loop shows memory effects due to spin ordering.

6 Python Code

The following Python script implements the XY model with Metropolis algorithm and produces a hysteresis animation.

```
# salvar_histerese_XY.py
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from pathlib import Path
import time

# ==============================
# Parameters
# ==============================
L = 32
T = 0.6
J = 1.0
H_max = 2.0
cycles = 2
frames_per_cycle = 160
sweeps_per_frame = 10
dtheta_max = 0.3
seed = 13

np.random.seed(seed)
N = L * L
total_frames = cycles * frames_per_cycle

theta = 2*np.pi*np.random.rand(L, L)

t_vals = np.linspace(0, 2*np.pi*cycles, total_frames)
H_vals = H_max * np.sin(t_vals)

def periodic(i):
    return i % L

def local_energy(theta, i, j, H):
    th = theta[i, j]
```

```
    nb_sum = (
        np.cos(th - theta[periodic(i+1), j]) +
        np.cos(th - theta[periodic(i-1), j]) +
        np.cos(th - theta[i, periodic(j+1)]) +
        np.cos(th - theta[i, periodic(j-1)])
    )
    return -J * nb_sum - H * np.cos(th)

def metropolis_sweep(theta, T, H):
    for _ in range(N):
        i = np.random.randint(0, L)
        j = np.random.randint(0, L)
        old_th = theta[i, j]
        E_old = local_energy(theta, i, j, H)
        new_th = old_th + (np.random.rand() - 0.5) * 2 * dtheta_max
        new_th = np.mod(new_th, 2*np.pi)
        theta[i, j] = new_th
        E_new = local_energy(theta, i, j, H)
        dE = E_new - E_old
        if dE > 0 and np.random.rand() > np.exp(-dE / T):
            theta[i, j] = old_th
    return theta

def magnetization(theta):
    mx = np.mean(np.cos(theta))
    my = np.mean(np.sin(theta))
    return mx, my

M_list = []
H_list = []
start_time = time.time()

for frame in range(total_frames):
    H = H_vals[frame]
    for _ in range(sweeps_per_frame):
        theta = metropolis_sweep(theta, T, H)
    mx, my = magnetization(theta)
    M_list.append(mx)
    H_list.append(H)

    if (frame+1) % 20 == 0 or frame == total_frames-1:
        print(f"Frame {frame+1}/{total_frames} - H={H:.3f} - Mx={mx:.3f}")

fig = plt.figure(figsize=(10,5))
ax_img = fig.add_axes([0.05, 0.12, 0.45, 0.76])
im = ax_img.imshow(theta, cmap="hsv", vmin=0, vmax=2*np.pi)
ax_img.set_xticks([]); ax_img.set_yticks([])
ax_img.set_title("Spin configuration (angles)")
```

```
ax_curve = fig.add_axes([0.57, 0.12, 0.38, 0.76])
ax_curve.set_xlim(-H_max*1.05, H_max*1.05)
ax_curve.set_ylim(-1.05, 1.05)
ax_curve.set_xlabel("Field H")
ax_curve.set_ylabel("Magnetization Mx")
line_full, = ax_curve.plot([], [], linewidth=1)
point_curr, = ax_curve.plot([], [], marker='o', markersize=6)

visual_theta = theta.copy()

def init():
    im.set_data(visual_theta)
    line_full.set_data([], [])
    point_curr.set_data([], [])
    return im, line_full, point_curr

def update(frame):
    global visual_theta
    H = H_vals[frame]
    for _ in range(2):
        visual_theta = metropolis_sweep(visual_theta, T, H)
    im.set_data(visual_theta)
    x = H_list[:frame+1]; y = M_list[:frame+1]
    line_full.set_data(x, y)
    point_curr.set_data(H_list[frame], M_list[frame])
    return im, line_full, point_curr

anim = animation.FuncAnimation(fig, update, frames=total_frames,
                               init_func=init, blit=True)

out_mp4 = Path("hysteresis_XY.mp4")
fps = 12

try:
    Writer = animation.writers['ffmpeg']
    writer = Writer(fps=fps, metadata=dict(artist='XY model'), bitrate=3000)
    anim.save(str(out_mp4), writer=writer)
except Exception as e:
    out_gif = Path("hysteresis_XY.gif")
    writer = animation.PillowWriter(fps=fps)
    anim.save(str(out_gif), writer=writer)
```