Function ising\_model\_simulation()

% Parameters

J = 5; % Interaction strength in meV

N = 20; % Lattice size

Iterations = 4000; % Number of iterations

Temperatures = 5:5:200; % Range of temperatures

% Initialize storage for energy and magnetization as functions of temperature

Energies = zeros(size(temperatures));

Magnetizations = zeros(size(temperatures));

% Loop over temperatures

For idx = 1:length(temperatures)

T = temperatures(idx);

% Initialize lattice (randomly populated)

Lattice = initialize\_lattice(N);

% Simulate the system using the Metropolis algorithm

[avg\_energy, avg\_magnetization] = simulate(lattice, J, T, iterations);

% Store results

Energies(idx) = avg\_energy;

Magnetizations(idx) = avg\_magnetization;

End

% Plot energy vs temperature

Figure;

Plot(temperatures, energies, ‘-o’);

Xlabel(‘Temperature (K)’);

Ylabel(‘Average Energy’);

Title(‘Energy vs Temperature’);

% Plot magnetization vs temperature

Figure;

Plot(temperatures, magnetizations, ‘-o’);

Xlabel(‘Temperature (K)’);

Ylabel(‘Average Magnetization’);

Title(‘Magnetization vs Temperature’);

End

% Function to initialize the lattice

Function lattice = initialize\_lattice(N)

Lattice = ones(N, N); % Start with all +1 spins

% Randomly set half of the spins to -1

Indices = randperm(N^2, N^2 / 2);

Lattice(indices) = -1;

End

% Function to plot the lattice

Function plot\_lattice(lattice)

Imagesc(lattice);

Colormap(‘jet’);

Colorbar;

Title(‘Spin Configuration’);

Axis square;

End

% Function to calculate the total energy of the lattice

Function energy = calc\_energy(lattice, J)

N = size(lattice, 1);

Energy = 0;

For i = 1:N

For j = 1:N

% Periodic boundary conditions

Up = lattice(mod(i-2, N) + 1, j);

Down = lattice(mod(i, N) + 1, j);

Left = lattice(i, mod(j-2, N) + 1);

Right = lattice(i, mod(j, N) + 1);

Energy = energy – J \* lattice(i, j) \* (up + down + left + right);

End

End

Energy = energy / 2; % To avoid double-counting interactions

End

% Function to calculate the total magnetization of the lattice

Function magnetization = calc\_magnetization(lattice)

Magnetization = sum(lattice(☺); % Sum of all spins

End

% Metropolis step function (single spin flip)

Function [lattice, delta\_E] = metropolis\_step(lattice, J, T)

N = size(lattice, 1);

I = randi(N);

J = randi(N);

S = lattice(i, j);

% Periodic boundary conditions for nearest neighbors

Up = lattice(mod(i-2, N) + 1, j);

Down = lattice(mod(i, N) + 1, j);

Left = lattice(i, mod(j-2, N) + 1);

Right = lattice(i, mod(j, N) + 1);

% Calculate the change in energy if the spin is flipped

Delta\_E = 2 \* J \* s \* (up + down + left + right);

% Metropolis condition

If delta\_E < 0 || rand() < exp(-delta\_E / T)

Lattice(i, j) = -s; % Flip the spin

Else

Delta\_E = 0; % No change in energy if spin is not flipped

End

End

% Main simulation function

Function [avg\_energy, avg\_magnetization] = simulate(lattice, J, T, iterations)

N = size(lattice, 1);

Energy = calc\_energy(lattice, J);

Magnetization = calc\_magnetization(lattice);

% Initialize arrays to store energy and magnetization

Energies = zeros(1, iterations);

Magnetizations = zeros(1, iterations);

% Perform the simulation

For step = 1:iterations

[lattice, delta\_E] = metropolis\_step(lattice, J, T);

Energy = energy + delta\_E;

Magnetization = calc\_magnetization(lattice);

% After equilibration, start collecting data

If step > 1000

Energies(step) = energy;

Magnetizations(step) = magnetization;

End

End

% Average after equilibration

Avg\_energy = mean(energies(1001:end));

Avg\_magnetization = mean(magnetizations(1001:end));

End