Two-Dimensional Ising Model Simulation

PHY2027 – Scientific Programming in C – Final Project Report

Miguel de Sousa

December 12, 2025

1 Theory

[1] The Ising model, introduced by Ernst Ising in 1925, is a mathematical model consisting of a lattice of 'spin' variables σ_α which can take discrete values of ± 1. These spins represent magnetic dipole moments of atomic spins that can be in one of two states: up (+1) or down (-1). The model is used to study phase transitions and critical phenomena in statistical mechanics. [1]Any two of these spins have a mutual interaction Energy

$$-E_{\alpha\beta}\sigma_\alpha\sigma_\beta \tag{1}$$

[1]Additionally, an external magnetic field B can interact with each spin, contributing an energy term of

$$-B\sigma_\alpha \tag{2}$$

The total energy of a configuration of spins on a lattice can be expressed as

$$E = -\sum_{\langle\alpha,\beta\rangle} E_{\alpha\beta}\sigma_\alpha\sigma_\beta - B\sum_\alpha \sigma_\alpha \tag{3}$$

where the first sum is over all pairs of neighbouring spins. The Ising model exhibits a phase transition at a critical temperature T_c, below which the spins tend to align, resulting in spontaneous magnetisation, and above which the spins are disordered.

2 Implementation

2.1 Data Structure and Initialisation (Part a)

I began by defining a structure type to represent the 2D Ising lattice, which included all necessary parameters as structure members for the model such as its N-dimenionsionality, values for temperature T, external magnetic field B, total energy and magnetisation of the lattice, and a pointer to an array of spins.

Lines 18-20: Assigning a char type to Spin, with constants for SPIN_UP and SPIN_DOWN defined with this new type. The char data type forces each spin to occupy only 1 byte of memory, which is efficient for large n by n lattices. The Spin type is used throughout the code, so defining it at the start improves code readability.

Lines 22-30: Creating the IsingLattice structure type with members for lattice size N, pointer to spin array, temperature T, external magnetic field B, current step number, total energy, and total magnetisation. Structures are useful for grouping related data together, so its perfect for representing the lattice and its properties. I used double for T, B, energy, and magnetisation to allow for fractional values and higher precision in calculations. The step member is an unsigned long long to allow a large number of Monte Carlo steps. Proving its reuseability, the spin pointer defined in the sructure types is allocatd to my new Spin data type.

After defining the structure type, I implemented the initialisation function to set up the lattice with random n by n spins.

Lines 145-150: The create_function takes lattice parameters as arguments and it allocates enough memory for the new lattice instance using malloc and the size of the IsingLattice structure type. It checks if the allocation was successful, printing an error message and exiting if not. The structure type pointer is initalised within the fucntion as *lattice.

Lines 152-158: Assign the input parameters to the corresponding structure members in the IsingLattice using structure pointer notation.

Lines 160-164/169-178: Allocate memory for the spin array within the lattice structure using malloc. The size is n by n multiplied by the size of a char (1 byte) since each spin is represented as a char. It checks for successful allocation, freeing the previously allocated lattice memory location if the spin allocation fails. I then assigned each lattice spin array value in the lattice to a random value of SPIN_UP or SPIN_DOWN using a count controlled loop with boundaries at 0 and n by n. The rand() function acted as the randomiser, with modulus 2 giving eitehr values of 0 or 1, which were mapped to either SPIN_UP or SPIN_DOWN. The energy and magnetisation values where also initialised using the ebergy and magnetisation functions

Lines 166: Finally, the function returns a pointer to the newly created and initialised lattice instance.

2.2 Energy and Magnetisation Calculation (Part a)

I proceeded to create a total energy function for my ising model using the hamiltonian research stated in the theory section. The ising_hamiltonian takes the N value of the lattice and also initialises a local variable energy with a double type to store all energy components.

Lines 184-186: The code uses nested for loops to iterate through every spin in the lattice, the first iterating through rows and the second through columns. The s value assigned the new datatype Spin, retreives the current spin values for the lattice as a one dimensional representation of the two dimensions lattice.

Lines 188-194: I iterate through every spin value (δ) in the lattice and sum the energy contribution from the two possible sources: the nearest-neighbor interactions and the external magnetic field (B), if present. This task was made efficient by the initial mapping of the two-dimensional lattice onto a single one-dimensional array. This mapping allows for fast index lookups and ensures efficiency during the summation loop. Each value was added onto the double value created earlier and the energy returned from the function.

Once the total energy function was created, I began working on the function to output normalised values of magnetisation.

2.3 Metropolis Algorithm (Part b)

2.4 Visualisation Method (Part c)

For my simulation, I decided to take guided user inputs using switch cases for standard ising states. The program will start by giving users flexibility with choices.

Lines 40-115: The user is prompted with multiple input choices for N, T and B values. These are selected as standard values you would use to extract information from thes imualtion such as tenpearture being broken down into Low, Critical and High options to view the possible extreme states of the ising simulation. These inputs are all sanitised, checking if the users did in fact only input one value. Furthermore, default case denies any value that doesnt sit with the required format. Users also have a custom choice. To avoid datatype issues, a temporary integer CASE was created to handle all users selections as using varaibles such as T with double is not valid for these cases.

Lines 117-143: The program checks for a valid proceed option. If true, a new instance of an Ising Lattice is created with previously selcted N, T and B values. After the Lattice has been initalised the main simulation flow can begin, I create a loop defining the Monte Carlo Steps of the program (simulation time), with the inner loop iterating through every lattice value. Each step in the Monte Carlo acts as a sweep of the lattice where every spin experineces a possible flip change caused by the metropolis algorithm. Each step is updated directly on the lattice and snapshots are taken in increments of 200 steps until the program loop finishes. All information being passed on to the python program is handled by the write_csv function.

Lines 244–274: The write_csv first checks for the successful opening of the ising_data csv file before proceeding. The function takes all values of the ising lattice, including a nested loop sweep of the lattice spin values and writes them to a single csv file. This includes updated magnetisation, energy spin flip values.

3 Program Exploration and Results

3.1 Temperature Effect with Zero External Field ($B = 0$)

3.1.1 Low Temperature $\approx 1.5Jk_B^{-1}$

134-137: Once the simulation csv is complete, the user is notified and the lattice is freed from memory.
View Ising Low-Temperature Simulation (Online)

3.1.2 Critical Temperature $\approx 2.269Jk_B^{-1}$

View Ising Critical-Temperature Simulation (Online)

3.1.3 High Temperature $\approx 7.5Jk_B^{-1}$

View Ising High-Temperature Simulation (Online)

3.2 Effect of External Magnetic Field ($B \neq 0$)

3.3 Boundary and Neighbour Effects

4 Conclusion

References

[1] B. M. McCoy and T. T. Wu, *The Two-Dimensional Ising Model*, Harvard University Press, Reprint 2014.