

# 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  $\sigma_\alpha$ , which can take discrete values of  $\pm 1$ . 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:

$$-E_{\alpha\beta}\sigma_\alpha\sigma_\beta \quad (1)$$

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

$$-B\sigma_\alpha \quad (2)$$

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

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

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-dimensionality, 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  $N \times N$  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  $N$ , a pointer to the 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 it’s 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 reusability, the spin pointer defined in the structure is allocated to my new Spin data type.

After defining the structure type, I implemented the initialisation function to set up the lattice with random  $N \times N$  spins.

Lines 145-150: The create function takes lattice parameters as arguments and 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 initialised within the function as \*lattice.

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

Lines 160-164/169-178: Memory is allocated for the spin array within the lattice structure using malloc. The size is  $N \times N$  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  $N \times N$ . The rand() function acted as the randomiser, with modulus 2 giving either values of 0 or 1, which were mapped to either SPIN\_UP or SPIN\_DOWN. The energy and magnetisation values were also initialised using the energy 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, retrieves the current spin values for the lattice as a one-dimensional representation of the two-dimensional lattice.

Lines 188-194: I iterate through every spin value ( $\delta$ ) in the lattice and sum the energy contribution from the two possible sources: the nearest-neighbour 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 was 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)

The Metropolis algorithm is the main function governing the evolution of the system. It decides whether a randomly chosen spin is likely to flip state based on probability.

Lines 222-224: Using rand(), random  $i$  and  $j$  locations can be chosen to pinpoint a spin value within the lattice in an unbiased fashion.

Lines 226: Instead of calculating the total energy again, I used the chosen spin location and the change in energy function to figure out the energy change if the spin flip did occur.

Lines 228-241: If the energy change is less than 0 for the spin, then the energy change will always be favourable, and you can flip that particular lattice spin and return 1. Nothing is accepted at absolute zero temperature, therefore that can be ignored as an option. This leaves the interesting part of the flip probability, energy levels greater than 0. In this case, it follows a Boltzmann factor whose value is stored in a double. Using the rand function again, I created a randomised  $r$  value for each iteration of this algorithm, and when  $r$  is less than the acceptance probability (Boltzmann factor), then the lattice energy flip is successful.

## 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 the simulation, such as temperature 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 user did, in fact, only input one value. Furthermore, the default case denies any value that doesn't sit with the required format. Users also have

a custom choice. To avoid datatype issues, a temporary integer CASE was created to handle all user selections as using variables 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 selected  $N$ ,  $T$ , and  $B$  values. After the Lattice has been initialised, 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 experiences 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, and spin flip values.

134-137: Once the simulation CSV is complete, the user is notified and the lattice is freed from memory.

### 3 Program Exploration and Results

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

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

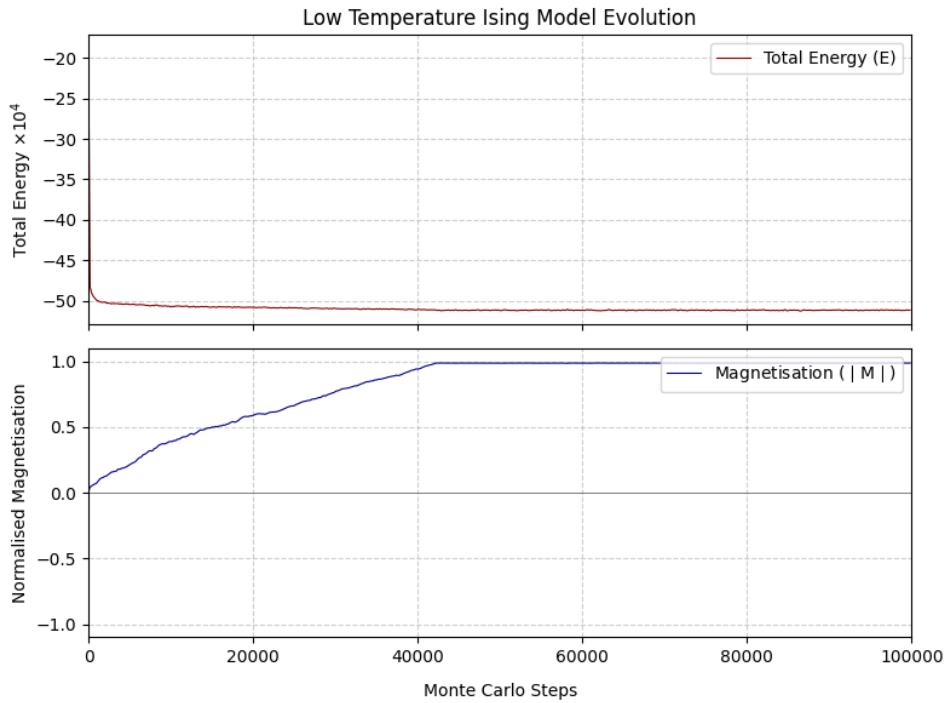


Figure 1: Ising Model Simulation at Low Temperature ( $T \approx 1.5Jk_B^{-1}$ )

##### [View Ising Low-Temperature Simulation \(Online\)](#)

At a temperature of 1.5, the Ising model simulation is below its critical temperature and therefore lies within the ferromagnetic phase. In this phase, the spins tend to align in the same direction, resulting in a net magnetic moment. As the simulation progresses, we see the formation of large domains of aligned spins, leading to a normalised magnetisation approaching 1. Thermal fluctuations are minimal at this low temperature, so the spins remain mostly stable once they align.

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

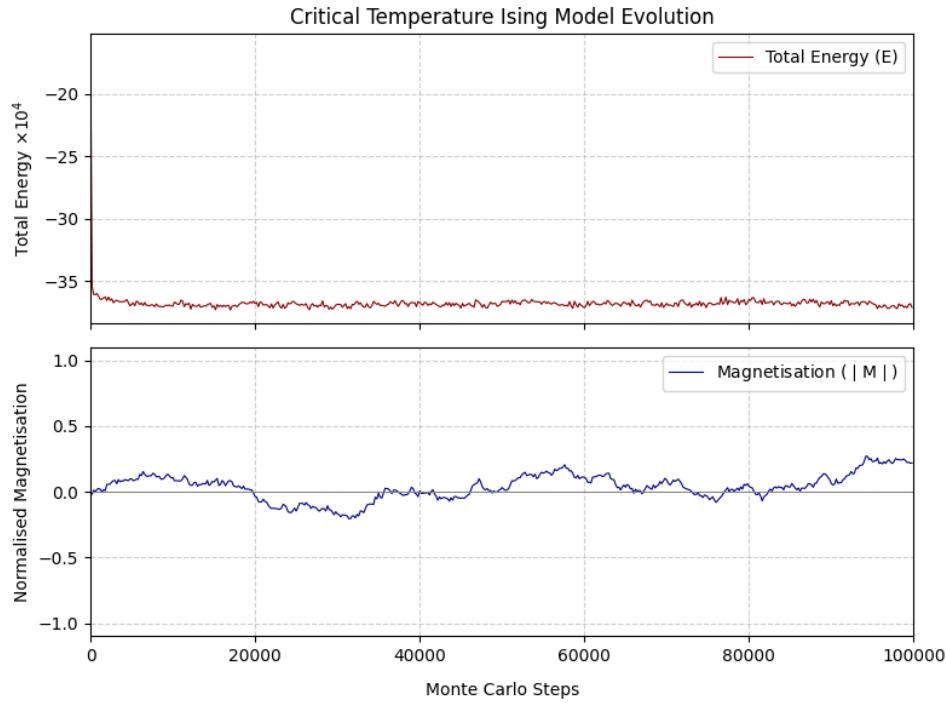


Figure 2: Ising Model Simulation at Critical Temperature ( $T \approx 2.269 Jk_B^{-1}$ )

#### [View Ising Critical-Temperature Simulation \(Online\)](#)

At the critical temperature of approximately 2.269, the Ising model undergoes a phase transition from the ferromagnetic to the paramagnetic phase. In this simulation, we see many fluctuations in spin alignment as the system has high sensitivity to thermal energy. Large clusters or domains of aligned spins form and disintegrate frequently, leading to a normalised magnetisation that hovers around zero. This is a characteristic behaviour at the critical point, which remains the same even if the simulation is scaled up to larger lattice sizes.

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

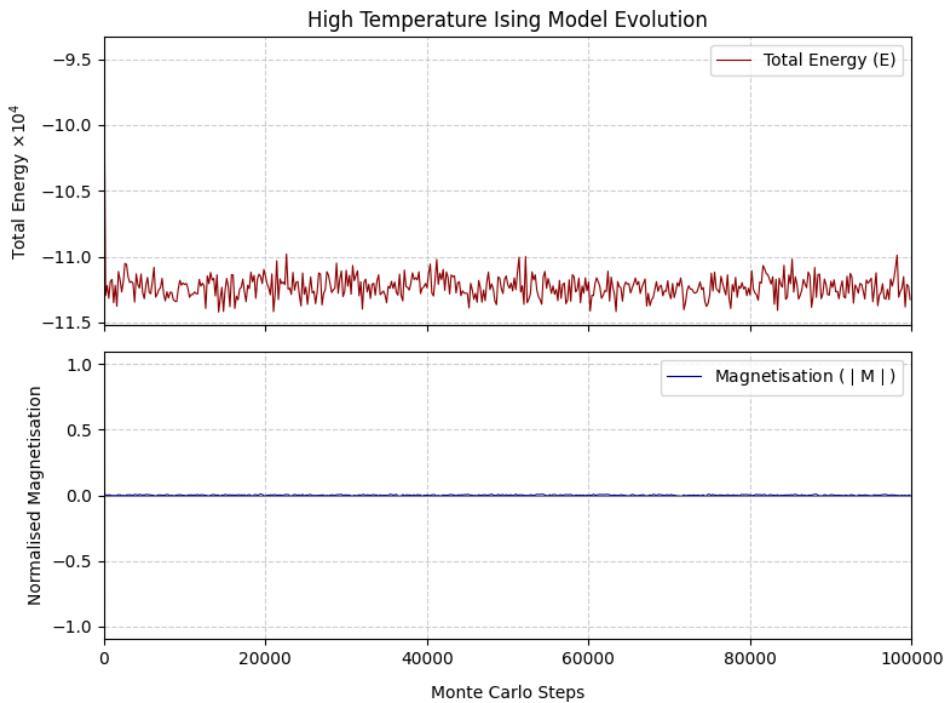


Figure 3: Ising Model Simulation at High Temperature ( $T \approx 7.5Jk_B^{-1}$ )

#### [View Ising High-Temperature Simulation \(Online\)](#)

At a high temperature of 7.5, the Ising model is well above its critical temperature and remains in the paramagnetic phase. In this phase, thermal fluctuations dominate, causing the spins to be oriented randomly with no possibility for order. As the simulation runs, we can see that the spins do not form any significant clusters, resulting in a normalised magnetisation that stays close to zero. This high temperature causes a disordered state for the remaining duration of the simulation.

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

#### [View Ising Field Simulation \(Online\)](#)

When an external magnetic field is applied to the Ising model, it influences the alignment of the spins of the lattice. In this simulation, I observed the changes when a moderate external field ( $B = 1.0$ ) is applied at the critical temperature of the model. The external field biases the spins to align in the direction of the field, leading to a net magnetisation even at the critical temperature, where normalised magnetisation should hover around zero. Interestingly, the presence of the external field suppresses the large fluctuations seen in the previous critical simulation; however, the system still shows the creation and destruction of small clusters, just of a bigger size than those seen in the high-temperature simulation without a field. This means that the external magnetic field stabilises the spin and remains at the same value for the simulation duration, unlike the low-temperature simulation.

## 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.