Computer Simulation I – Computational Methods, Lab 3

The aim of this lab (and the subsequent assignment) is twofold:

- to simulate the physics of the Ising model using the Metropolis Monte Carlo algorithm and object-oriented programming in Python
- to explore the capabilities and limitations of generative artificial intelligence (AI) as an aid to computer simulation of physical systems

1. We will use **Microsoft Copilot** as our AI helper during this lab. To access this, open a web browser and navigate to `copilot.cloud.microsoft`. Log in using your TCD email/username and password. You should see a dialogue box where you can type in a **prompt**: a text string that will be used to generate a response by Copilot.

2. Type in the following prompt and press Enter/Return:
   Write a Python class to simulate the 1D Ising model using the Metropolis Monte Carlo algorithm

3. Read through the code produced by the AI and try to understand how it works. To help you, use the following AI prompt:
   Explain how each part of the code works

4. Open your favourite IDE (e.g. Spyder, Jupyter, VSCode etc.) and copy-paste in the AI-generated code. Look for a button like this to copy all of the code to clipboard:

   

   Try running the code, and experimenting with different values of the parameters.

5. Now let's try to improve the code. If they are not included already, add methods to:
   - Calculate the total magnetisation of the spin lattice
   - Calculate the total energy of the spin lattice
   - Print nicely to screen some information about the spin lattice, including:
     - number of spins
     - temperature
     - total magnetisation
     - total energy

   Test your new methods to check they work as expected.

6. Modify the class to simulate the 1D Ising model in the presence of an external magnetic field h. That is, given a spin state $\mathbf{S} = [S_0, S_1, S_2, \ldots]$ where $S_i = \pm 1$ is the spin on lattice site i, the total energy is now given by

$$E(\mathbf{S}) = -J \sum_{i=0}^{N-1} S_i S_{i+1} - h \sum_{i=0}^{N-1} S_i$$

   Here, J is the coupling energy (usually we choose units where $J = 1$); h is the external field, which should be passed as an argument to the `__init__` method and stored as one of the object's attributes.
   (Hint: you can use the AI to help you do this!)

7. Monte Carlo simulation works by averaging over a large number of random samples. Add a method (if not included already) that runs some number of sweeps through the lattice, computes the magnetisation and the energy after each sweep, then averages the results (and returns or stores the values appropriately!). Experiment to find out how many sweeps you need to perform before the magnetisation and energy converge (i.e. averaging over even more sweeps does not appreciably change their values).
   (Hint: the AI can help here too! Also check out ising_proc.py from Lecture 6.)

8. Plot the average magnetisation and the average energy of the 1D Ising model as a function of the external magnetic field in the range $0 \leq h \leq 4J$, for temperature $k_B T = J$. Does this show the expected behaviour for a paramagnet?

9. Plot the magnetisation as a function of temperature for various different values of the magnetic field h, using log-log scale. Confirm that the average magnetisation $\langle M \rangle$ obeys Curie's law for high temperature:
   $$\langle M \rangle \sim T^{-1}$$
   What determines the temperature scale above which this law holds?

10. Modify your class to explore the following questions:
    - The magnetic susceptibility yields the change in average magnetisation upon a small change in external field
      $$\chi = \frac{\partial \langle M \rangle}{\partial h}$$
      It can also be computed from the fluctuations of the magnetisation, i.e. its variance across Metropolis samples, as
      $$\chi = \frac{1}{k_B T} \left(\langle M^2 \rangle - \langle M \rangle^2 \right)$$
      Verify that these two relations give the same result.
    - How does the choice of initial state affect the result? To explore this, modify your `__init__` method to initialise the lattice in a state of your choosing. Explore the effect of choosing a random initial state versus a polarised state (e.g. all spins $S_i = +1$).
    - Modify your class to allow the user to choose whether the sweeps are performed sequentially (i.e. flipping each spin in the lattice in turn) or by choosing sites at random. How does this choice affect the results and the performance of the algorithm?