PH30056 Comp Phys B – Coursework 2: Ising Model

General guidance on project

This document includes some background information on the Ising model, and the instructions for this coursework assignment. The idea is to use a C++ program to investigate the physical features of this model. You will *upload the report and a working program to Moodle*. We understand that one program will not be sufficient to show your answers to all parts of this coursework: you should hand in a version that illustrates what you did. The report should be written at a level suitable for a third-year physics student who is not taking this unit. On the report, you should identify yourself by your candidate number: *you should not include your name*.

The deadline for the report and the uploads to Moodle is **4pm on Wednesday 11 May**. The page limit (7 pages, excluding possible Appendices), report template and marking scheme are the same as in the DLA project (available on Moodle).

Remember that you need to manage your time during the semester. Make sure that you leave time to write up your report.

The report should have a clear structure, with a title and abstract, as well as sections dealing with the various exercises within the coursework. You need to explain your methods as well as showing your results. Please do *not* include your entire code. If necessary, you may want to include extracts from the code in your report (it may be easiest to put these into an appendix).

You are expected to work individually on Coursework 2, but we encourage you to discuss it with other students. **The report that you submit must be your own work and you must acknowledge any help that you have received.** Plagiarising any part of anyone else's report is forbidden. It is in your own interest to ensure that no one has the opportunity to copy your work.

Part of developing good project management is to write *tidy and readable code*. This will help you as you go, and it will also help anyone else who tries to read the code (including lab demonstrators, and whoever marks your report). Another important project management idea is *version control*. This involves making backup copies of your files as you move forward with the exercises. That way you can always return to the older versions of the code if you need to.

Version history:
AS, March 2022

Section 1 – Ising Model Introduction

Definition of the model

In the Ising model, there are N sites on a square lattice (grid). We take periodic boundary conditions. On each site there is a "spin" which can be up or down. If spin i is up then we write $s_i = 1$ and if it is down we write $s_i = -1$. The energy of spin i is

$$E_i = -h_i s_i$$

where

$$h_i = \sum_{j \text{ n.n. of } i} (J s_j).$$

Here "n.n." stands for nearest neighbours, and J is a (positive) number that indicates the strength of the interactions between the spins. So the energy of site i is negative if s_i has the same sign as the total spin of its neighbours. The quantity h_i is like a local magnetic field for spin i, since we expect spins to have low energy when they are aligned with magnetic fields.

The total energy of the system depends on all the spins in the system. We define it to be

$$E = \frac{1}{2} \sum_i E_i$$

This allows us to write the energy as a sum over all pairs of nearest neighbour sites:

$$E = \sum_{\langle ij \rangle} (-J s_i s_j) \tag{1}$$

The notation $\sum_{\langle ij \rangle}$ conventionally means a sum over all nearest neighbour pairs. (These angle brackets have nothing to do with any kind of average.) Note that s_i has no units, and J has units of energy.

Physical interpretation of the model

The variable s_i represents the orientation of an electron spin. In most real magnets, spins can point in any direction, so it is a rather crude approximation to restrict s_i to just two possible values. But there are some special materials whose atomic structure means that electron spins always point along one specific symmetry axis. In this case, one recalls that electrons have spin-$\frac{1}{2}$ so it's reasonable to have two spin states, which point either parallel or anti-parallel to the symmetry axis. In any case, taking $s_i = \pm 1$ makes the model much simpler to analyse, so it's useful as a starting point for a more general theory.

In real materials, aligning the spins tends to lower the energy because of the *exchange interaction*. The low energy comes from symmetry properties of the electronic wavefunction, which mean that electrons whose spins are aligned less likely to be close to each other, reducing the repulsive Coulomb forces between them.

Quantities to measure

Some interesting quantities in the Ising model are the energy per spin (E/N) and the magnetisation (per spin)

$$M = \frac{1}{N} \sum_i s_i \tag{2}$$

If almost all the spins are "up" then $M \approx 1$; if they are almost all down then $M \approx -1$. If the spins are roughly half-and-half then $M \approx 0$.

Note that the system has a symmetry: the energy of the system does not change if we flip the states of all spins (that is, replace $s_i \to -s_i$ for all i). At equilibrium, this will mean that the probability of finding magnetisation M is the same as finding magnetisation $-M$.

Boltzmann distribution and Metropolis Monte Carlo

To understand the properties of this system at thermal equilibrium, we need to recall some results from statistical mechanics. If we specify values for all the spins s_i, this defines a *microstate*. We write S for a general microstate. In equilibrium at temperature T, the probability of finding the system in microstate S must be

$$p_{\mathrm{eq}}(S) = \frac{1}{Z(T)} \mathrm{e}^{-E(S)/k_B T}$$

where $E(S)$ is the energy of microstate S, and $Z(T) = \sum_S \mathrm{e}^{-E(S)/k_B T}$ is the partition function. (The sum runs over all possible microstates: there are 2^N possibilities since each spin has two possible states.)

Metropolis Monte Carlo (MC) is a method for updating the spins in the system. After many updates, the system reaches a state in which the probability of a particular microstate appearing is equal to $p_{\mathrm{eq}}(S)$. This means that we can estimate equilibrium averages by averaging over a large number of microstates that have been generated by Metropolis MC.

The rule of Metropolis MC is that we pick a spin at random. Suppose we pick spin i. We calculate the change in the system energy, ΔE_i, that would occur if spin i changed its state from s_i to $-s_i$. If $\Delta E_i \leq 0$ (energy decreases) then we "accept" this change, and s_i gets changed to $-s_i$. If $\Delta E_i > 0$ (energy increases) then we accept this change with probability $p_i = \mathrm{e}^{-\Delta E_i/k_B T}$. Otherwise we "reject" the change and leave s_i in its original state. Then we repeat this procedure many times. A single iteration of the procedure is called an *attempted flip* since we attempt to change ("flip") spin i, but this change might not be accepted.

The fact that flips which increase the energy are not always accepted means that the system is biased to low energy. However, there are many states with high energy, so moves that increase the energy are more likely to be proposed. This means that the system settles into a steady state, which can be proven to be consistent with the Botlzmann distribution. The large number of high-energy states is related to the system entropy.

The Metropolis MC method is implemented in the example code which has been provided to you. Section 3 of this document explains what you should investigate, using this code.

Theory – Mean field, susceptibility, and specific heat capacity

Some extra information about theoretical calculations on the Ising model is given in Section 4 of this document.

Section 2 – Ising Model Program

This section explains how to get the code working, and how it is structured.

Getting started

The first step is to set up a new C++ project in Visual Studio. For instructions as to how to do this, you may want to go back to the DLA handout. You will need to install the nupengl library as before. You will need to create two new classes called `Window` and `IsingSystem` and an inline class called `rnd`

The code for the relevant `.cpp` and `.h` files (including the main program file) is provided on Moodle.

Program overview

The program is similar to the one from the DLA project. It creates an `IsingSystem` and it uses a global pointer to access that system. There is a `handleKeyPress` function that controls what happens when you press a key. There is also an `update` function which tells the `IsingSystem` to update itself and then tells openGL to wait a while before calling the `update` function again.

You will probably not need to edit the `rnd` and `Window` classes. The main business happens in the `IsingSystem` class:

- The most important *member variable* is the `grid`, which works like a 2d array. The idea is that `grid[x][y]` contains the value (± 1) of the spin s_i at co-ordinates (x, y). This array is accessed by the `setGrid` and `readGrid` functions. The variable `gridSize` stores the size of the grid. This is a fixed constant (you can change it within the code but the program can't change its value). The variable `inverseTemperatureBeta` controls the temperature (see Section 3).
- There is a single constructor function that takes a `Window` pointer as input. This function mostly just initialises the variables in the class. It also calls a function called `Reset` which also sets up the `grid` and the initial temperature.
- The most important functions are `Update` which causes the system to update itself (it attempts one spin flip per spin); and `DrawSquares`, which draws the system on the screen. *Note: as before, it is important that these two functions are kept separate: the drawing function should not change any system variables and the update function should not do any drawing.*
- There are several other member variables which we do not discuss in detail: the `rgen` object is of type `rnd` and deals with random numbers; there are variables like `isActive` that determine whether the system is running or paused, and `slowNotFast` which determines whether it is running slowly or quickly. There are also various functions that allow these variables to be changed (eg `pauseRunning` and `setSlow`).
  As a general point, it is a good idea to use functions to modify member variables; this tends to make the program more flexible.

You are encouraged to take a look at the various functions and see if you can understand how they work. There should be plenty of comments in the code that help with this.

Section 3 – Ising Model Exercises

These are the instructions for the Ising Model project. We assume that the program is compiled and runs ok (see Section 2). Start the program. The window shows "down" spins as green ($s_i = -1$) and and "up" spins as blue ($s_i = +1$). The system starts in a paused state with all spins "down" (-1). The system size is 40×40 sites. At each step a spin is chosen at random, and its state is updated according to the Metropolis Monte Carlo rules (see Section 1).

The parameters of the model are the energy J and the temperature T. However, the behaviour of the system depends only on the ratio $T_0 = Tk_{\rm B}/J$ which has no units (it is dimensionless). In fact, within the code, we use a variable `inverseTemperatureBeta` which is $\beta = (1/T_0) = \frac{J}{k_{\rm B}T}$. (This variable is printed to the screen.) Note that the use of dimensionless variables is a general feature of computational modelling.

The Metropolis algorithm operates by "MC sweeps": in each sweep there are N attempted flips, where N is the total number of spins.

Exercise 1 – convergence to equilibrium

The program starts with $\beta = 0.25$ and all spins down (-1). Start the program (in the paused state) and hit the 'u' key a few times. This will cause a few MC sweeps to happen, and some of the spins will change state. After a few sweeps, you can probably see that roughly half of them are blue ($+1$) and half are green (-1). Your first goal is to characterise the process by which the spins end up in this "half-and-half" state.

Write a function in the `IsingSystem` class that calculates the magnetisation M defined in Eq. (2) of section 1. (Note the sum in that equation runs over all sites in the grid: within the program you will need to loop separately over the rows and the columns.)

Modify the program so that if you hit the 'm' key then this magnetisation gets calculated and printed to the screen.

Calculate (and print to a file) the magnetisation after each of the first 10 sweeps. By using several simulations all with the same initial condition, make a figure showing how the average magnetisation behaves as the number of sweeps increases.

Do the same for the energy defined in Eq. (1) of Section 1. In fact the best quantity to measure is probably either the dimensionless energy E/J or the dimensionless energy per spin $E/(NJ)$. (There is a function `setPosNeighbour` in the `IsingSystem` class which you can use to work out which are the neighbours of each site. Note in particular that this function takes care of the periodic boundaries.)

Repeat this test for different temperatures with β between 0.2 and 0.7. (You may need to use simulations that are longer than 10 sweeps.) What do you notice about the time taken to converge to a steady state? Can you explain the trends?

Exercise 2 – measuring equilibrium averages

As described in Section 1, the Metropolis MC algorithm allows us to calculate averages of observable quantities at equilibrium. Our examples of observable quantities are the energy and the magnetisation.

The idea is to wait for the simulation to converges to a steady state; then let the simulation generate m configurations; calculate the observable quantities for each; and estimate the equilibrium average by averaging these m measurements.

To do this, we need to fix the number of measurements m, as well as the number of MC sweeps n_0 that we wait at the beginning (to allow the system to converge to its steady state); and the number of MC sweeps n that we allow between each of our m measurements. You need to choose these numbers in a sensible way, so that you have an accurate and efficient measurement. (In general, accuracy requires that m and n should be reasonably large: for efficiency it is useful if n is not too small. See, Lecture 7.)

Use this method to estimate equilibrium averages of the dimensionless energy per spin $\langle E/(NJ)\rangle$, the magnetisation $\langle M\rangle$, and the modulus of the magnetisation $\langle |M|\rangle$.

Obtain these estimates for dimensionless temperatures T_0 between 1.0 and 4.0. Can you explain the trends? (It is suggested that you use simulations where the initial condition has either all spins up or all spins down.)

Exercise 3 – comparison with mean-field theory

The mean-field theory and the exact solution for the Ising model are discussed in Section 4 of this document, and in lectures.

Compare the exact solution for the magnetisation (16) with your numerical estimates. Note that the exact results are calculated by assuming that the system is very large.

For the mean-field theory, we can get theoretical predictions by solving Eq. (15). This can be done numerically. The predictions of this theory for the energy and magnetisation are provided on Moodle. (Make sure that you read the data correctly. It is available in both .TXT and .CSV formats). Compare these with your numerical results. The theory is approximate so it does not give always the right numbers, but to what extent does it give the right trends? Does the theory predict the correct limiting behaviour at very low and very high temperatures?

Exercise 4 – specific heat capacity and magnetic susceptibility

When comparing simulations with experiments, two interesting quantities are the heat capacity c of a magnetic system, and its magnetic susceptibility χ. These are defined as

$$c = \frac{\partial}{\partial T}\langle E/N\rangle \tag{3}$$

$$\chi = \frac{\partial}{\partial h}\langle M\rangle, \tag{4}$$

where h is an external magnetic field that is applied to the system. (More precisely, $h = m_0 B$ where B is the magnetic field and m_0 is the dipole moment of a single spin. Note that h has units of energy.)

In laboratory experiments, these quantities are (fairly) easy to obtain by changing the tem-

perature or magnetic field, and measuring changes in the energy or magnetisation. In computer simulations, there is a nicer way to measure these quantities: it can be shown that

$$c = \frac{1}{Nk_BT^2}\mathrm{Var}(E) \tag{5}$$

$$\chi = \frac{N}{k_BT}\mathrm{Var}(M), \tag{6}$$

where $\mathrm{Var}(E)$ is the *variance* of the energy, and $\mathrm{Var}(M)$ is the variance of the magnetisation. These equations mean that we can choose to measure variances instead of trying to measure derivatives with respect to h or T. Equation (6) is derived in Section 4 of this document, using properties of the Boltzmann distribution.

Recall that the variance of a quantity X can be calculated as

$$\begin{aligned}\mathrm{Var}(X) &= \langle (X - \langle X\rangle)^2\rangle \\ &= \langle X^2\rangle - \langle X\rangle^2. \end{aligned} \tag{7}$$

You should be able to calculate $\langle E^2\rangle$ and $\langle M^2\rangle$ from your simulations, and hence you can calculate c and χ. (To be precise, you can calculate dimensionless quantities that are proportional to c and χ.)

Calculate these quantities and show them in your report. Are your results consistent with the definition of c as the derivative of the energy with respect to temperature? Theoretical predictions of c and χ from mean-field theory are included on Moodle. Compare the trends in your data with those predicted by this theory.

Exercise 5 – correlation function

An interesting function to measure in the Ising model is the "spin-spin correlation function"

$$G_{ij} = \langle s_i s_j\rangle \tag{8}$$

where i and j are two different sites.

All sites in the system are equivalent so G_{ij} depends only on the *relative positions* of i and j. (For example, if i and j are nearest neighbour sites then it doesn't matter which pair of neighbouring sites you choose, you should always get the same value for the average in Eq. (8).)

Pick a pair of neighbouring sites and calculate G_{ij} for that pair (you will need to take n measurements by analysing n different configurations of the system, as you did when calculating the magnetisation). Then check that you can get to the same answer (but more quickly) if you average over all pairs of neighbours within each configuration. (It may be more efficient to take the average within the C++ program instead of writing all your measurements to a file.)

Investigate how the correlation function depends on the distance between sites i and j. You should be able to convince yourself that if $G_{ij} > 0$, this means that spins i and j are more likely to be equal to each other. [If $s_i = s_j$, what can you say about $s_i s_j$? What if $s_i \neq s_j$? Remember that the allowed vales of s_i are ± 1.] You might expect that if spins i and j are far apart then their states are *independent*, in which case $G_{ij} = \langle s_i\rangle\langle s_j\rangle$. Is this true? How does G_{ij} behave when the system is close to the phase transition?

Section 4 – Ising Model Theory

This section contains some theoretical results that you will want to use when interpreting and analysing your results.

Magnetic susceptibility and specific heat capacity

To calculate the magnetic susceptibility χ we need to introduce a magnetic field to the model. We do this by defining the energy of the system *in a magnetic field* as

$$E_h = E - NhM \tag{9}$$

where $E = -J\sum_{\langle ij\rangle} s_i s_j$ is the original energy defined in Section 1, M is the magnetisation, N is the total number of spins, and h is the magnetic field (but see the note on units just below Eq. 4 in Exercise 4).

The average magnetisation is obtained from the Boltzmann distribution as

$$\langle M\rangle = \frac{1}{Z(h,T)}\sum_S M(S)\,\mathrm{e}^{-[E(S)-NhM(S)]/k_BT} \tag{10}$$

where S is a microstate of the system, the sum runs over all possible microstates, and

$$Z(h,T) = \sum_S \mathrm{e}^{-[E(S)-NhM(S)]/k_BT}.$$

From here, it is not hard to check that

$$\frac{\partial}{\partial h}\ln Z(h,T) = \frac{1}{Z(h,T)}\sum_S M(S)\frac{N}{k_BT}\,\mathrm{e}^{-[E(S)-NhM(S)]/k_BT} \tag{11}$$

$$= \frac{N}{k_\mathrm{B}T}\langle M\rangle \tag{12}$$

Differentiating Equ. (11) again with respect to h one has

$$\frac{N}{k_\mathrm{B}T}\frac{\partial}{\partial h}\langle M\rangle = \frac{1}{Z(h,T)}\sum_S M(S)^2\left(\frac{N}{k_BT}\right)^2\mathrm{e}^{-[E(S)-NhM(S)]/k_BT} - \frac{1}{Z(h,T)^2}\left[\sum_S M(S)\frac{N}{k_BT}\,\mathrm{e}^{-[E(S)-NhM(S)]/k_BT}\right]^2 \tag{13}$$

We recognise the right hand side as $(\langle M^2\rangle - \langle M\rangle^2)(N/k_BT)^2$ and so

$$\chi = \frac{\partial}{\partial h}\langle M\rangle = \frac{N}{k_BT}\mathrm{Var}(M)$$

To prove that the specific heat $c = (\partial/\partial T)\langle E/N\rangle$ is equal to $(1/Nk_BT^2)\mathrm{Var}(E)$, the steps are almost the same, except that you can set $h = 0$ from the beginning which shortens the equations a bit. (You may find it simpler to write $\beta = 1/k_BT$ and use the chain rule to write $(\partial/\partial T) = (-k_B\beta^2)(\partial/\partial\beta)$.)

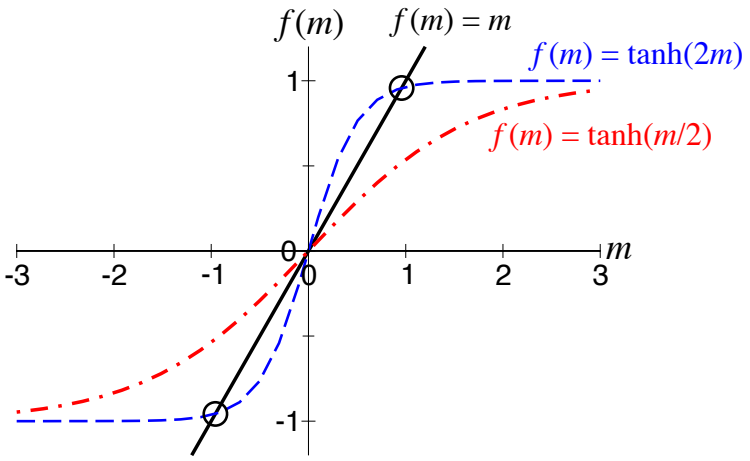

FIG. 1. Graphical solution of equations of the form $m = \tanh(am)$. If a is large, this corresponds to low temperatures in Eq. (15). There is always a solution at $m = 0$, where the black line crosses the two coloured lines. For $a = 2$ there are also solutions at $m \approx \pm 0.96$, indicated by circles.

Mean field theory

In order to understand how phase transitions can appear in the Ising model, we can use mean-field theory (also called Curie-Weiss theory in this context). We imagine that for low temperatures, the average magnetisation $m = \langle M \rangle$ may be different from zero, and we want to estimate the value of m. Note that since all spins have the same average behaviour, the average magnetisation is simply equal to the average of a single spin: $\langle M \rangle = \langle s_i \rangle$.

Now recall that for a single spin in a magnetic field h, the average spin

$$m = \langle s_i \rangle = \tanh(h/k_B T). \tag{14}$$

(Eq. (14) can be proved by considering the two possible states that the spin can be in and using the Boltzmann distribution with energy $E_0 = -hs_i$.)

Recall from the Section 1 that spin i feels a "local field" h_i that is given by the sum of the spins of its neighbours, multiplied by J. We write z for the number of neighbours of each spin (on the square lattice $z = 4$ but the argument is more general than this.) Then, the average local field is $\langle h_i \rangle = zJ\langle s_j \rangle$, where $\langle s_j \rangle$ is the average spin for a neighbour of spin i. On average, there is nothing special about any particular spin, so $\langle s_j \rangle = \langle s_i \rangle$, and we already have $\langle s_i \rangle = m$. Therefore $\langle h_i \rangle = zJm$.

The *mean-field approximation* assumes that spin i behaves as if it was a single spin, with an applied magnetic field equal to $\langle h_i \rangle$ (this is the "mean field"). In that case, replacing h by $\langle h_i \rangle$ in Eq. (14), one obtains

$$m = \tanh(mzJ/k_B T). \tag{15}$$

Note that this equation required a strong assumption, so we can't expect it to give results that are exactly correct. But it is simple and can help us to understand what is happening in the system.

To get a prediction for the average magnetisation m, we need to solve Eq. (15). One possible solution is $m = 0$ but for low temperatures, there are also two other solutions, as

shown graphically in Fig. 1. These equations cannot be solved analytically but if we define $T_c = zJ/k_B$, we can show that for $T > T_c$ the only solution is $m = 0$ but for $T < T_c$ there are multiple solutions. Hence T_c is the critical temperature (phase transition temperature) below which we can predict that $m \neq 0$. Also, if $T < T_c$ but $T_c - T$ is small then $m \approx \pm\sqrt{3(T_c - T)/T_c}$.

Exact Solution of the $2d$ Ising model

In a famous calculation, Lars Onsager obtained an exact solution for the free energy of the Ising model in two dimensions [Phys. Rev. **65**, 117 (1944)]. He showed that

$$T_c = \frac{2J}{k_B \ln(1+\sqrt{2})} \approx 2.27 J/k_B$$

Later it was also shown that the magnetisation of the Ising model for $T \leq T_c$ is equal to

$$m = \pm \left[1 - \frac{1}{\left(\sinh\left(\frac{T_c}{T} \ln(1+\sqrt{2}) \right) \right)^4} \right]^{1/8} \tag{16}$$

(Note that all these exact results are valid only for very large lattices, this was an important part of Onsager's calculation.)

The Ising model can be solved exactly in 1 and 2 dimensions, but there is no exact solution in 3 or more dimensions. On the other hand, a nice feature of mean-field theory is that it can be used in any dimension, and it becomes increasingly accurate as the dimension gets higher.

Version history:
AS, March 2022