11 Statistical Physics

11.2 Monte Carlo simulation of the Ising Model (10 units)

Knowledge of the Statistical Physics course will be useful for this project.

1 Introduction

The Ising model is a simple model for a magnet. It consists of N "spins", which are variables $\sigma_1, \sigma_2, \ldots, \sigma_N$. Each spin (for example σ_i) takes values in $\{-1, +1\}$. A *configuration* is specified by giving the value of every spin; we denote such a configuration by $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_N)$.

To each configuration $\boldsymbol{\sigma}$ we associate an energy $E(\boldsymbol{\sigma})$ which for the one-dimensional (1d) model is

$$E(\boldsymbol{\sigma}) = -J\left[\sigma_N\sigma_1 + \sum_{i=1}^{N-1}\sigma_i\sigma_{i+1}\right] - h\sum_{i=1}^{N}\sigma_i.$$

The first term in the square brackets is present because we have arranged the spins on a circle so the Nth spin is adjacent to the first one ("periodic boundary conditions"). In the canonical ensemble at temperature T, configuration $\boldsymbol{\sigma}$ occurs with a probability given by the Boltzmann distribution

$$p(\boldsymbol{\sigma}|T) = \frac{1}{Z}\exp\left(-\frac{E(\boldsymbol{\sigma})}{T}\right)$$

where Z is the partition function (normalisation constant).

A common aim in statistical mechanics is to compute averages of observable quantities with respect to p. For example we might compute the average energy $U(T) = \sum_{\boldsymbol{\sigma}} E(\boldsymbol{\sigma})p(\boldsymbol{\sigma}|T)$, where the sum runs over all possible configurations. For the 1d Ising model this sum can be done analytically. In two dimensions, the average energy can be computed analytically for the special case $h = 0$, but not otherwise. In three dimensions, the average energy cannot be computed analytically.

In this project, we consider a general numerical method for computing averages in the canonical ensemble. (In principle this method can be used in any dimension, here we consider $d = 1$.) The idea is to define a dynamical process by which the Ising model evolves, as a function of time. This process is random – within its steady state, configuration $\boldsymbol{\sigma}$ appears with probability $p(\boldsymbol{\sigma}|T)$. Hence one can use the random process to estimate averages such as $U(T)$. This method is very useful in systems that cannot be solved analytically. It is related to Markov chain Monte Carlo (MCMC) methods in statistics.

1.1 Theoretical analysis

The project focusses on the magnetisation of the system. The results in this section can be derived using methods from part II statistical physics. The magnetisation of a configuration is

$$\hat{M}(\boldsymbol{\sigma}) = \frac{1}{N}\sum_i \sigma_i$$

Its average (with respect to the Boltzmann distribution) is

$$M(T) = \sum_{\boldsymbol{\sigma}} \hat{M}(\boldsymbol{\sigma})p(\boldsymbol{\sigma}|T)$$

where the sum runs over all configurations. Note that $M(T)$ depends on J, h, N as well as T. The (scaled) magnetic susceptibility is $\chi(T) = T\frac{\partial M}{\partial h}$. This quantity is related to the variance of $\hat{M}$, as

$$\chi(T) = N \sum_{\boldsymbol{\sigma}} [\hat{M}(\boldsymbol{\sigma}) - M(T)]^2 p(\boldsymbol{\sigma}|T)$$

In the limit $N \to \infty$, it can be shown by transfer matrix methods that

$$M(T) = \sinh(h/T) \frac{\cosh(h/T) + \sqrt{\sinh^2(h/T) + e^{-4J/T}}}{\sinh^2(h/T) + e^{-4J/T} + \cosh(h/T)\sqrt{\sinh^2(h/T) + e^{-4J/T}}} \,. \tag{1}$$

For $h = 0$ we have also

$$\chi(T) = e^{2J/T} \,. \tag{2}$$

2 The Simulation

In the following, you will use a computer to generate a sequence of configurations, which correspond to the Ising model evolving (stochastically) as a function of time. The time t is a positive integer and the t-th configuration is denoted by $\boldsymbol{\sigma}(t)$.

To generate the configuration $\boldsymbol{\sigma}(t+1)$, first set $\boldsymbol{\sigma}(t+1) = \boldsymbol{\sigma}(t)$; then choose a random integer i between 1 and N; and finally replace $\sigma_i(t+1)$ by either -1 or 1, according to

$$\begin{aligned} \text{Prob}[\sigma_i(t+1) = +1] &= \frac{1}{1 + e^{-2\kappa_i/T}} \\ \text{Prob}[\sigma_i(t+1) = -1] &= \frac{1}{1 + e^{2\kappa_i/T}} \end{aligned} \tag{3}$$

with

$$\kappa_i = h + J(\sigma_{i+1} + \sigma_{i-1}).$$

This update yields the new configuration $\boldsymbol{\sigma}(t+1)$, which may be identical to $\boldsymbol{\sigma}(t)$, or it may differ in exactly one spin. (It is easy to verify that the probabilities of the two possible values of $\sigma_i(t+1)$ sum to unity. In the formula for κ_i then σ_{N+1} should be interpreted as σ_1 and σ_0 should be interpreted as σ_N, to take care of the periodic boundaries.) The variable κ_i is sometimes called the "local field": you can check that the difference in energy between configurations with $\sigma_i = -1$ and $\sigma_i = 1$ is $2\kappa_i$.

After many updates, the probability that the system ends in configuration $\boldsymbol{\sigma}$ converges to $p(\boldsymbol{\sigma}|T)$, this is discussed in a later section. It is useful to define a rescaled time $\tau = t/N$. If $\tau = 1$, this means that N updates have been performed.

2.1 The program

You will write a program that stores a configuration of N Ising spins, and performs random updates using the rule described above. The configuration should be stored as a sequence of 1's and -1's in an array of size N. You can verify that while the parameters J, h, T all appear in the algorithm, the behaviour of the system only depends on J/T and h/T. These parameters will have the same value for every update (they are independent of time). In order to do the updates, you will need to generate random numbers: the MATLAB manual has some information on this.

Here is an outline of the program that you should write. It depends on a few parameters whose values will be discussed at the end

(1) Choose an initial condition with $\sigma_i = 1$ for all i.

(2) Define a parameter n_w and perform $n_w \times N$ updates to "warm up" the system. [The initial configuration was our choice, but the idea is that if n_w is reasonably large then we the final configuration will be distributed (approximately) as $p(\boldsymbol{\sigma}|T)$.] After this warming up, the rescaled time is $\tau = n_w$. Set a counter $k = 1$ and define a parameter K.

(3) For the configuration obtained at the current time t, compute and store the magnetisation as

$$M_k = \frac{1}{N} \sum_i \sigma_i(t),$$

(4) Continue updating for an additional rescaled time n_I (that is, perform $n_I \times N$ updates). Increase the counter k by 1.

(5) Repeat steps (3) and (4) until $k = K$. At the end of this procedure you will have stored a list of K values of the magnetisation, which are $M_1, M_2, \ldots, M_K$. It is suggested that you store these in an array of size K.

(6) Compute the average of your magnetisation values, and also a scaled measure of their variance, as

$$\overline{M} = \frac{1}{K} \sum_{k=1}^{K} M_k,$$
$$\hat{\chi} = \frac{N}{K} \sum_{k=1}^{K} (M_k - \overline{M})^2$$

The algorithm is designed in such a way that if K and n_w are large enough, we expect $\overline{M}$ and $\hat{\chi}$ to be close to the equilibrium magnetisation and susceptibility defined in (1,2).

2.2 Numerical results

As a starting point, some recommended values for parameters are

$$N = 50, \qquad n_w = 10^4, \qquad n_I = 5, \qquad K = 2048\ .$$

Instead of specifying the parameter T, it may be useful to work instead with $\beta = 1/T$. A typical simulation with these parameters should not take more than a few minutes. In some cases it may be useful to store your lists of magnetisation values in a file, to avoid recreating many similar lists.

Question 1 Fix $J = 1$ and $h = 0.1$. Choose some values of β between 0 and 2.5; for each value of β, run a simulation and compute $\overline{M}$ and $\hat{\chi}$. Plot these values on a graph. Compare your result for $\overline{M}$ with the theoretical prediction of (1).

Question 2 The simulations use random numbers, so if you do several different computations with the same parameters, you should get different answers for $\overline{M}$. Check that you do get different answers (if not, you may need to read about choosing different "seeds" for your random number generator). For each point in the graph in question 1, do several simulations, compute the standard deviation of the values of $\overline{M}$ and $\hat{\chi}$ between the runs, and use these to add suitable error bars to your plot. (If the error bars are too small to see then this is not a problem, but you should state this fact.)

Question 3 Repeat the analysis of questions 1 and 2, but now for $J = 1$ and $h = 0$. (That is, make plots of $\overline{M}$ and $\hat{\chi}$ against β, with error bars.) In this case you can also compare your result for $\hat{\chi}$ with the theoretical prediction of (2). Make sure to analyse the errors in $\overline{M}$: even if the theoretical prediction for this quantity is trivial, your numerical estimate $\overline{M}$ is not a trivial quantity.
When interpreting these results, you should recall that this numerical method is accurate only if n_w and K are large enough. The dependence of the results on these parameters will be discussed later, in question 6.

2.3 Validity of the method, and error estimates

It was claimed above that as the number of updates goes to infinity, the final configuration generated by the method will be distributed as $p(\boldsymbol{\sigma}|T)$. This can be proven, but we don't do it here. However, the main step in the proof would be to verify the *detailed balance* property, which concerns the probability that an update generates configuration $\boldsymbol{\sigma}$ at time $t+1$, given that the configuration at time t was $\boldsymbol{\sigma}'$. If this probability is $P(\boldsymbol{\sigma}' \to \boldsymbol{\sigma})$ then the required condition is that for all pairs $\boldsymbol{\sigma}, \boldsymbol{\sigma}'$ then

$$p(\boldsymbol{\sigma}'|T)P(\boldsymbol{\sigma}' \to \boldsymbol{\sigma}) = p(\boldsymbol{\sigma}|T)P(\boldsymbol{\sigma} \to \boldsymbol{\sigma}')$$

Roughly speaking, this says that if the configurations are already distributed as $p(\boldsymbol{\sigma}|T)$, then the probability to go from $\boldsymbol{\sigma}'$ to $\boldsymbol{\sigma}$ is equal to the probability of making the opposite transition (from $\boldsymbol{\sigma}$ to $\boldsymbol{\sigma}'$).

Question 4 Show that the detailed balance property holds for the simulation algorithm that you are using.

We now make a short theoretical analysis of the sequence of magnetisation values $M_1, M_2, \ldots, M_K$ that is generated by your program. These are sequences of random variables, in the sense that repeating the same computation with the same parameters will give a different sequence. If n_w is large enough, each of these numbers should have the same distribution, but that they are correlated random variables (they are not independent, because values that are adjacent in the sequence are likely to be similar to each other). This is particularly true if n_I is small.

Programming Task: To investigate this, generate a long sequence of magnetisation values (at least $K = 16384$), at the state point $h = 0$, $J = 1$, $\beta = 0.7$. You need to choose sensible values of n_w and n_I so that the $\overline{M}$ obtained from this sequence is a good estimate of $M(T)$.

Split this sequence into blocks (subsequences) of length ℓ with (for example) $\ell = 2, 4, 8, \ldots$. There will be K/ℓ such blocks and you will compute the average magnetisation of the r-th block as

$$m(\ell, r) = \frac{1}{\ell} \sum_{k=r\ell+1}^{(r+1)\ell} M_k$$

with $r = 0, 1, 2, \ldots, (K/\ell) - 1$. Note, $m(K, 0) = \overline{M}$.

If the M_k were independent, standard statistical methods could be used to show that the variance of $m(\ell, r)$ is proportional to $1/\ell$. If the M_k are not independent (as in your case), the variance of $m(\ell, r)$ depends on whether typical data points within a block are strongly- or

weakly-correlated. Since you have a long time series, you can estimate the variance of $m(\ell, r)$ as

$$S(\ell)^2 = \frac{\ell}{K} \sum_{r=0}^{(K/\ell)-1} [m(\ell, r) - \overline{M}]^2$$

and it is useful to define a rescaled variance

$$s(\ell) = \ell \cdot S(\ell)^2$$

If the correlations of the samples within a block are weak, one expects $s(\ell)$ to depend weakly on ℓ. This typically happens for longer blocks. On the other hand, if the correlations are strong, one expects $s(\ell)$ to depend strongly on ℓ (this typically happens for small blocks, in which all the values are strongly correlated with each other).

Question 5 Using your long sequence of magnetisation results, compute $s(\ell)$ for $\ell = 2, 4, 8, 16, 32$ and present the results in a graph. Explain the behaviour that you observe. What happens at different state points, for example $\beta = 1.7$? Why?

Question 6 Consider the graph (with error bars) for $\overline{M}$ from question 3. Investigate how the mean value $\overline{M}$ and its error bar depend on n_w and K. Explain the behaviour that you observe. It may be useful to refer to your answer to question 5.

Question 7 Discuss the physical behaviour that is occurring at high and low temperatures in the 1d Ising model. You will find it useful to explain how typical configurations depend on the system parameters. What happens if you increase (or reduce) the system size N?
Explain how the behaviour of the system affects the computational time required to compute accurate estimates of $M(T)$ using this simulation method.

