1. We calculate the probabilities $P(\mu \to \nu)$ for transitions to all states which can be reached in one Monte Carlo step from the current state μ. We choose a new state ν with probability proportional to $P(\mu \to \nu)$ and change the state of the system to ν.
2. Using our values for the $P(\mu \to \nu)$ we also calculate the time interval Δt. Notice that we have to recalculate Δt at each step, since in general it will change from one step to the next.
3. We increment the time t by Δt, to mimic the effect of waiting Δt Monte Carlo steps. The variable t keeps a record of how long the simulation has gone on for in "equivalent Monte Carlo steps".

While this technique is in many respects a very elegant solution to the problem of simulating a system at low temperatures (or any other system which has a low acceptance ratio), it does suffer from one obvious drawback, which is that step (1) above involves calculating $P(\mu \to \nu)$ for every possible state ν which is accessible from μ. There may be very many such states (for some systems the number goes up exponentially with the size of the system), and so this step may take a very long time. However, in some cases, it turns out that the set of transition probabilities is very similar from one step of the algorithm to the next, only a few of them changing at each step, and hence it is possible to keep a table of probabilities and update only a few entries at each step to keep the table current. In cases such as these the continuous time method becomes very efficient and can save us a great deal of CPU time, despite being more complex than the accept/reject method discussed in the previous section. One example of a continuous time Monte Carlo algorithm is presented in Section 5.2.1 for the conserved-order-parameter Ising model.

In the next few chapters, we will examine a number of common models used for calculating the equilibrium properties of condensed matter systems, and show how the general ideas presented in this chapter can be used to find efficient numerical solutions to these physical problems.

Problems

2.1 Derive Equation (2.8) from Equation (1.1).

2.2 Consider a system which has just three energy states, with energies $E_0 < E_1 < E_2$. Suppose that the only allowed transitions are ones of the form $\mu \to \nu$, where $\nu = (\mu + 1) \bmod 3$. Such a system cannot satisfy detailed balance. Show nonetheless that it is possible to choose the transition probabilities $P(\mu \to \nu)$ so that the Boltzmann distribution is an equilibrium of the dynamics.

3

The Ising model and the Metropolis algorithm

In Section 1.2.2 we introduced the Ising model, which is one of the simplest and best studied of statistical mechanical models. In this chapter and the next we look in detail at the Monte Carlo methods that have been used to investigate the properties of this model. As well as demonstrating the application of the basic principles described in the last chapter, the study of the Ising model provides an excellent introduction to the most important Monte Carlo algorithms in use today. Along the way we will also look at some of the tricks used for implementing Monte Carlo algorithms in computer programs and at some of the standard techniques used to analyse the data those programs generate.

To recap briefly, the Ising model is a simple model of a magnet, in which dipoles or "spins" s_i are placed on the sites i of a lattice. Each spin can take either of two values: $+1$ and -1. If there are N sites on the lattice, then the system can be in 2^N states, and the energy of any particular state is given by the Ising Hamiltonian:

$$H = -J \sum_{\langle ij \rangle} s_i s_j - B \sum_i s_i, \tag{3.1}$$

where J is an interaction energy between nearest-neighbour spins $\langle ij \rangle$, and B is an external magnetic field. We are interested in simulating an Ising system of finite size using Monte Carlo methods, so that we can estimate the values of quantities such as the magnetization m (Equation (1.34)) or the specific heat c (Equation (1.37)) at any given temperature. Most of the interesting questions concerning the Ising model can be answered by performing simulations in zero magnetic field $B = 0$, so for the moment at least we will concentrate on this case.

3.1 The Metropolis algorithm

The very first Monte Carlo algorithm we introduce in this book is the most famous and widely used algorithm of them all, the **Metropolis algorithm**, which was introduced by Nicolas Metropolis and his co-workers in a 1953 paper on simulations of hard-sphere gases (Metropolis *et al.* 1953). We will use this algorithm to illustrate many of the general concepts involved in a real Monte Carlo calculation, including equilibration, measurement of expectation values, and the calculation of errors. First however, let us see how the algorithm is arrived at, and how one might go about implementing it on a computer.

The derivation of the Metropolis algorithm follows exactly the plan we outlined in Section 2.3. We choose a set of selection probabilities $g(\mu \to \nu)$, one for each possible transition from one state to another, $\mu \to \nu$, and then we choose a set of acceptance probabilities $A(\mu \to \nu)$ such that Equation (2.17) satisfies the condition of detailed balance, Equation (2.14). The algorithm works by repeatedly choosing a new state ν, and then accepting or rejecting it at random with our chosen acceptance probability. If the state is accepted, the computer changes the system to the new state ν. If not, it just leaves it as it is. And then the process is repeated again and again.

The selection probabilities $g(\mu \to \nu)$ should be chosen so that the condition of ergodicity—the requirement that every state be accessible from every other in a finite number of steps—is fulfilled (see Section 2.2.2). This still leaves us a good deal of latitude about how they are chosen; given an initial state μ we can generate any number of candidate states ν simply by flipping different subsets of the spins on the lattice. However, as we demonstrated in Section 1.2.1, the energies of systems in thermal equilibrium stay within a very narrow range—the energy fluctuations are small by comparison with the energy of the entire system. In other words, the real system spends most of its time in a subset of states with a narrow range of energies and rarely makes transitions that change the energy of the system dramatically. This tells us that we probably don't want to spend much time in our simulation considering transitions to states whose energy is very different from the energy of the present state. The simplest way of achieving this in the Ising model is to consider only those states which differ from the present one by the flip of a single spin. An algorithm which does this is said to have **single-spin-flip dynamics**. The algorithm we describe in this chapter has single-spin-flip dynamics, although this is not what makes it the Metropolis algorithm. (As discussed below, it is the particular choice of acceptance ratio that characterizes the Metropolis algorithm. Our algorithm would still be a Metropolis algorithm even if it flipped many spins at once.)

Using single-spin-flip dynamics guarantees that the new state ν will have an energy E_ν differing from the current energy E_μ by at most $2J$ for each bond between the spin we flip and its neighbours. For example, on a square lattice in two dimensions each spin has four neighbours, so the maximum difference in energy would be $8J$. The general expression is $2zJ$, where z is the **lattice coordination number**, i.e., the number of neighbours that each site on the lattice has.[1] Using single-spin-flip dynamics also ensures that our algorithm obeys ergodicity, since it is clear that we can get from any state to any other on a finite lattice by flipping one by one each of the spins by which the two states differ.

In the Metropolis algorithm the selection probabilities $g(\mu \to \nu)$ for each of the possible states ν are all chosen to be equal. The selection probabilities of all other states are set to zero. Suppose there are N spins in the system we are simulating. With single-spin-flip dynamics there are then N different spins that we could flip, and hence N possible states ν which we can reach from a given state μ. Thus there are N selection probabilities $g(\mu \to \nu)$ which are non-zero, and each of them takes the value

$$g(\mu \to \nu) = \frac{1}{N}. \tag{3.2}$$

With these selection probabilities, the condition of detailed balance, Equation (2.14), takes the form

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{g(\mu \to \nu)A(\mu \to \nu)}{g(\nu \to \mu)A(\nu \to \mu)} = \frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \mathrm{e}^{-\beta(E_\nu - E_\mu)}. \tag{3.3}$$

Now we have to choose the acceptance ratios $A(\mu \to \nu)$ to satisfy this equation. As we pointed out in Section 2.2.3, one possibility is to choose

$$A(\mu \to \nu) = A_0 \mathrm{e}^{-\frac{1}{2}\beta(E_\nu - E_\mu)}. \tag{3.4}$$

The constant of proportionality A_0 cancels out in Equation (3.3), so we can choose any value for it that we like, except that $A(\mu \to \nu)$, being a probability, should never be allowed to become greater than one. As we mentioned above, the largest difference in energy $E_\nu - E_\mu$ that we can have between our two states is $2zJ$, where z is the lattice coordination number. That means that the largest value of $\mathrm{e}^{-\frac{1}{2}\beta(E_\nu - E_\mu)}$ is $\mathrm{e}^{\beta zJ}$. Thus, in order to make sure $A(\mu \to \nu) \leq 1$ we want to choose

$$A_0 \leq \mathrm{e}^{-\beta zJ}. \tag{3.5}$$

To make the algorithm as efficient as possible, we want the acceptance probabilities to be as large as possible, so we make A_0 as large as it is allowed to

[1]This is not the same thing as the "spin coordination number" which we introduce in Chapter 5. The spin coordination number is the number of spins j neighbouring i which have the same value as spin i: $s_j = s_i$.

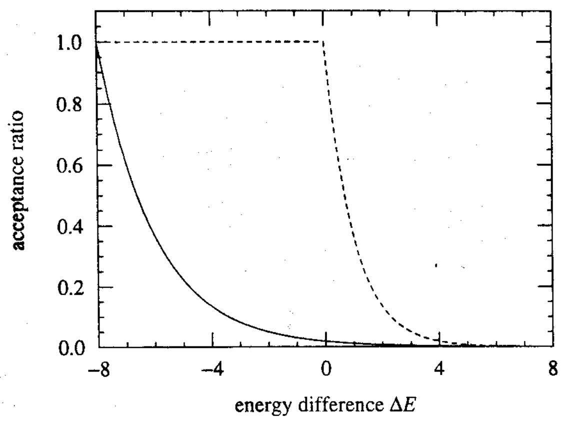

FIGURE 3.1 Plot of the acceptance ratio given in Equation (3.6) (solid line). This acceptance ratio gives rise to an algorithm which samples the Boltzmann distribution correctly, but is very inefficient, since it rejects the vast majority of the moves it selects for consideration. The Metropolis acceptance ratio (dashed line) is much more efficient.

be, which gives us

$$A(\mu \to \nu) = e^{-\frac{1}{2}\beta(E_\nu - E_\mu + 2zJ)}. \tag{3.6}$$

This is not the Metropolis algorithm (we are coming to that), but using this acceptance probability we can perform a Monte Carlo simulation of the Ising model, and it will correctly sample the Boltzmann distribution. However, the simulation will be very inefficient, because the acceptance ratio, Equation (3.6), is very small for almost all moves. Figure 3.1 shows the acceptance ratio (solid line) as a function of the energy difference $\Delta E = E_\nu - E_\mu$ over the allowed range of values for a simulation with $\beta = J = 1$ and a lattice coordination number $z = 4$, as on a square lattice for example. As we can see, although $A(\mu \to \nu)$ starts off at 1 for $\Delta E = -8$, it quickly falls to only about 0.13 at $\Delta E = -4$, and to only 0.02 when $\Delta E = 0$. The chances of making any move for which $\Delta E > 0$ are pitifully small, and in practice this means that an algorithm making use of this acceptance ratio would be tremendously slow, spending most of its time rejecting moves and not flipping any spins at all. The solution to this problem is as follows.

In Equation (3.4) we have assumed a particular functional form for the acceptance ratio, but the condition of detailed balance, Equation (3.3), doesn't actually require that it take this form. Equation (3.3) only specifies the ratio of pairs of acceptance probabilities, which still leaves us quite a lot of room to manœuvre. In fact, as we pointed out in Section 2.3, when given a con-

straint like (3.3) the way to maximize the acceptance ratios (and therefore produce the most efficient algorithm) is always to give the larger of the two ratios the largest value possible—namely 1—and then adjust the other to satisfy the constraint. To see how that works out in this case, suppose that of the two states μ and ν we are considering here, μ has the lower energy and ν the higher: $E_\mu < E_\nu$. Then the larger of the two acceptance ratios is $A(\nu \to \mu)$, so we set that equal to one. In order to satisfy Equation (3.3), $A(\mu \to \nu)$ must then take the value $e^{-\beta(E_\nu - E_\mu)}$. Thus the optimal algorithm is one in which

$$A(\mu \to \nu) = \begin{cases} e^{-\beta(E_\nu - E_\mu)} & \text{if } E_\nu - E_\mu > 0 \\ 1 & \text{otherwise.} \end{cases} \tag{3.7}$$

In other words, if we select a new state which has an energy lower than or equal to the present one, we should always accept the transition to that state. If it has a higher energy then we maybe accept it, with the probability given above. This is the Metropolis algorithm for the Ising model with single-spin-flip dynamics. It is Equation (3.7) which makes it the Metropolis algorithm. This is the part that was pioneered by Metropolis and co-workers in their paper on hard-sphere gases, and any algorithm, applied to any model, which chooses selection probabilities according to a rule like (3.7) can be said to be a Metropolis algorithm. At first, this rule may seem a little strange, especially the part about how we *always* accept a move that will lower the energy of the system. The first algorithm we suggested, Equation (3.6), seems much more natural in this respect, since it sometimes rejects moves to lower energy. However, as we have shown, the Metropolis algorithm satisfies detailed balance, and is by far the more efficient algorithm, so, natural or not, it has become the algorithm of choice in the overwhelming majority of Monte Carlo studies of simple statistical mechanical models in the last forty years. We have also plotted Equation (3.7) in Figure 3.1 (dashed line) for comparison between the two algorithms.

3.1.1 Implementing the Metropolis algorithm

Let us look now at how we would actually go about writing a computer program to perform a simulation of the Ising model using the Metropolis algorithm. For simplicity we will continue to focus on the case of zero magnetic field $B = 0$, although the generalization to the case $B \neq 0$ is not hard (see Problem 3.1). In fact almost all the past studies of the Ising model, including Onsager's exact solution in two dimensions, have looked only at the zero-field case.

First, we need an actual lattice of spins to work with, so we would define a set of N variables—an array—which can take the values ± 1. Probably we would use integer variables, so it would be an integer array. Normally, we apply **periodic boundary conditions** to the array. That is, we specify that the spins on one edge of the lattice are neighbours of the corresponding spins on the other edge. This ensures that all spins have the same number of neighbours and local geometry, and that there are no special edge spins which have different properties from the others; all the spins are equivalent and the system is completely translationally invariant. In practice this considerably improves the quality of the results from our simulation.

A variation on the idea of periodic boundary conditions is to use "helical boundary conditions" which are only very slightly different from periodic ones and possess all the same benefits but are usually considerably simpler to implement and can make our simulation significantly faster. The various types of boundary conditions and their implementation are described in detail in Section 13.1, along with methods for representing most common lattice geometries using arrays.

Next we need to decide at what temperature, or alternatively at what value of β we want to perform our simulation, and we need to choose some starting value for each of the spins—the initial state of the system. In a lot of cases, the initial state we choose is not particularly important, though sometimes a judicious choice can reduce the time taken to come to equilibrium (see Section 3.2). The two most commonly used initial states are the zero-temperature state and the infinite temperature state. At $T = 0$ the Ising model will be in its ground state. When the interaction energy J is greater than zero and the external field B is zero (as is the case in the simulations we will present in this chapter) there are actually two ground states. These are the states in which the spins are all up or all down. It is easy to see that these must be ground states, since in these states each pair of spins in the first term of Equation (3.1) contributes the lowest possible energy $-J$ to the Hamiltonian. In any other state there will be pairs of spins which contribute $+J$ to the Hamiltonian, so that its overall value will be higher. (If $B \neq 0$ then there will only be one ground state—the field ensures that one of the two is favoured over the other.) The other commonly used initial state is the $T = \infty$ state. When $T = \infty$ the thermal energy kT available to flip the spins is infinitely larger than the energy due to the spin–spin interaction J, so the spins are just oriented randomly up or down in an uncorrelated fashion.

These two choices of initial state are popular because they each correspond to a known, well defined temperature, and they are easy to generate. There is, however, one other initial state which can sometimes be very useful, which we should mention. Often we don't just perform one simulation at a single temperature, but rather a set of simulations one after another at a range of different values of T, to probe the behaviour of the model with varying temperature. In this case it is often advantageous to us to choose as the initial state of our system the *final* state of the system for a simulation

at a nearby temperature. For example, suppose we are interested in probing a range of temperatures between $T = 1.0$ and $T = 2.0$ in steps of 0.1. (Here and throughout much of the rest of this book, we measure temperature in energy units, so that $k = 1$. Thus when we say $T = 2.0$ we mean that $\beta^{-1} = 2.0$.) Then we might start off by performing a simulation at $T = 1.0$ using the zero-temperature state with all spins aligned as our initial state. At the end of the simulation, the system will be in equilibrium at $T = 1.0$, and we can use the final state of that simulation as the initial state for the simulation at $T = 1.1$, and so on. The justification for doing this is clear: we hope that the equilibrium state at $T = 1.0$ will be more similar to that at $T = 1.1$ than will the zero-temperature state. In most cases this is a correct assumption and our system will come to equilibrium quicker with this initial state than with either a $T = 0$ or a $T = \infty$ one.

Now we start our simulation. The first step is to generate a new state—the one we called ν in the discussion above. The new state should differ from the present one by the flip of just one spin, and every such state should be exactly as likely as every other to be generated. This is an easy task to perform. We just pick a single spin k at random from the lattice to be flipped. Next, we need to calculate the difference in energy $E_\nu - E_\mu$ between the new state and the old one, in order to apply Equation (3.7). The most straightforward way to do this would be to calculate E_μ directly by substituting the values s_i^μ of the spins in state μ into the Hamiltonian (3.1), then flip spin k and calculate E_ν, and take the difference. This, however, is not a very efficient way to do it. Even in zero magnetic field $B = 0$ we still have to perform the sum in the first term of (3.1), which has as many terms as there are bonds on the lattice, which is $\frac{1}{2}Nz$. But most of these terms don't change when we flip our single spin. The only ones that change are those that involve the flipped spin. The others stay the same and so cancel out when we take the difference $E_\nu - E_\mu$. The *change* in energy between the two states is thus

$$\begin{aligned} E_\nu - E_\mu &= -J \sum_{\langle ij \rangle} s_i^\nu s_j^\nu + J \sum_{\langle ij \rangle} s_i^\mu s_j^\mu \\ &= -J \sum_{i \text{ n.n. to } k} s_i^\mu (s_k^\nu - s_k^\mu). \end{aligned} \tag{3.8}$$

In the second line the sum is over only those spins i which are nearest neighbours of the flipped spin k and we have made use of the fact that all of these spins do not themselves flip, so that $s_i^\nu = s_i^\mu$. Now if $s_k^\mu = +1$, then after spin k has been flipped we must have $s_k^\nu = -1$, so that $s_k^\nu - s_k^\mu = -2$. On the other hand, if $s_k^\mu = -1$ then $s_k^\nu - s_k^\mu = +2$. Thus we can write

$$s_k^\nu - s_k^\mu = -2s_k^\mu, \tag{3.9}$$

and so

$$E_\nu - E_\mu = 2J \sum_{i \text{ n.n. to } k} s_i^\mu s_k^\mu$$
$$= 2Js_k^\mu \sum_{i \text{ n.n. to } k} s_i^\mu. \qquad (3.10)$$

This expression only involves summing over z terms, rather than $\frac{1}{2}Nz$, and it doesn't require us to perform any multiplications for the terms in the sum, so it is much more efficient than evaluating the change in energy directly. What's more, it involves only the values of the spins in state μ, so we can evaluate it *before* we actually flip the spin k.

The algorithm thus involves calculating $E_\nu - E_\mu$ from Equation (3.10) and then following the rule given in Equation (3.7): if $E_\nu - E_\mu \leq 0$ we definitely accept the move and flip the spin $s_k \to -s_k$. If $E_\nu - E_\mu > 0$ we still may want to flip the spin. The Metropolis algorithm tells us to flip it with probability $A(\mu \to \nu) = e^{-\beta(E_\nu - E_\mu)}$. We can do this as follows. We evaluate the acceptance ratio $A(\mu \to \nu)$ using our value of $E_\nu - E_\mu$ from Equation (3.10), and then we choose a random number r between zero and one. (Strictly the number can be equal to zero, but it must be less than one: $0 \leq r < 1$.) If that number is *less* than our acceptance ratio, $r < A(\mu \to \nu)$, then we flip the spin. If it isn't, we leave the spin alone.

And that is our complete algorithm. Now we just keep on repeating the same calculations over and over again, choosing a spin, calculating the energy change we would get if we flipped it, and then deciding whether to flip it according to Equation (3.7). Actually, there is one other trick that we can pull that makes our algorithm a bit faster still. (In fact, on most computers it will make it a lot faster.) One of the slowest parts of the algorithm as we have described it is the calculation of the exponential, which we have to perform if the energy of the new state we choose is greater than that of the current state. Calculating exponentials on a computer is usually done using a polynomial approximation which involves performing a number of floating-point multiplications and additions, and can take a considerable amount of time. We can save ourselves this effort, and thereby speed up our simulation, if we notice that the quantity, Equation (3.10), which we are calculating the exponential of, can only take a rather small number of values. Each of the terms in the sum can only take the values +1 and −1. So the entire sum, which has z terms, can only take the values $-z, -z+2, -z+4\ldots$ and so on up to $+z$—a total of $z+1$ possible values. And we only actually need to calculate the exponential when the sum is negative (see Equation (3.7) again), so in fact there are only $\frac{1}{2}z$ values of $E_\mu - E_\nu$ for which we ever need to calculate exponentials. Thus, it makes good sense to calculate the values of these $\frac{1}{2}z$ exponentials before we start the calculation proper, and store them in the computer's memory (usually in an array), where we can simply look them up when we need them during the simulation. We pay the one-time cost of evaluating them at the beginning, and save a great deal more by never having to evaluate any exponentials again during the rest of the simulation. Not only does this save us the effort of evaluating all those exponentials, it also means that we hardly have to perform any floating-point arithmetic during the simulation. The only floating-point calculations will be in the generation of the random number r. (We discuss techniques for doing this in Chapter 16.) All the other calculations involve only integers, which on most computers are much quicker to deal with than real numbers.

3.2 Equilibration

So what do we do with our Monte Carlo program for the Ising model, once we have written it? Well, we probably want to know the answer to some questions like "What is the magnetization at such-and-such a temperature?", or "How does the internal energy behave with temperature over such-and-such a range?" To answer these questions we have to do two things. First we have to run our simulation for a suitably long period of time until it has come to equilibrium at the temperature we are interested in—this period is called the **equilibration time** τ_{eq}—and then we have to measure the quantity we are interested in over another suitably long period of time and average it, to evaluate the estimator of that quantity (see Equation (2.4)). This leads us to several other questions. What exactly do we mean by "allowing the system to come to equilibrium"? And how long is a "suitably long" time for it to happen? How do we go about measuring our quantity of interest, and how long do we have to average over to get a result of a desired degree of accuracy? These are very general questions which we need to consider every time we do a Monte Carlo calculation. Although we will be discussing them here using our Ising model simulation as an example, the conclusions we will draw in this and the following sections are applicable to all equilibrium Monte Carlo calculations. These sections are some of the most important in this book.

As we discussed in Section 1.2, "equilibrium" means that the average probability of finding our system in any particular state μ is proportional to the Boltzmann weight $e^{-\beta E_\mu}$ of that state. If we start our system off in a state such as the $T = 0$ or $T = \infty$ states described in the last section and we want to perform a simulation at some finite non-zero temperature, it will take a little time before we reach equilibrium. To see this, recall that, as we demonstrated in Section 1.2.1, a system at equilibrium spends the overwhelming majority of its time in a small subset of states in which its internal energy and other properties take a narrow range of values. In order to get a good estimate of the equilibrium value of any property of the system therefore, we need to wait until it has found its way to one of the states that

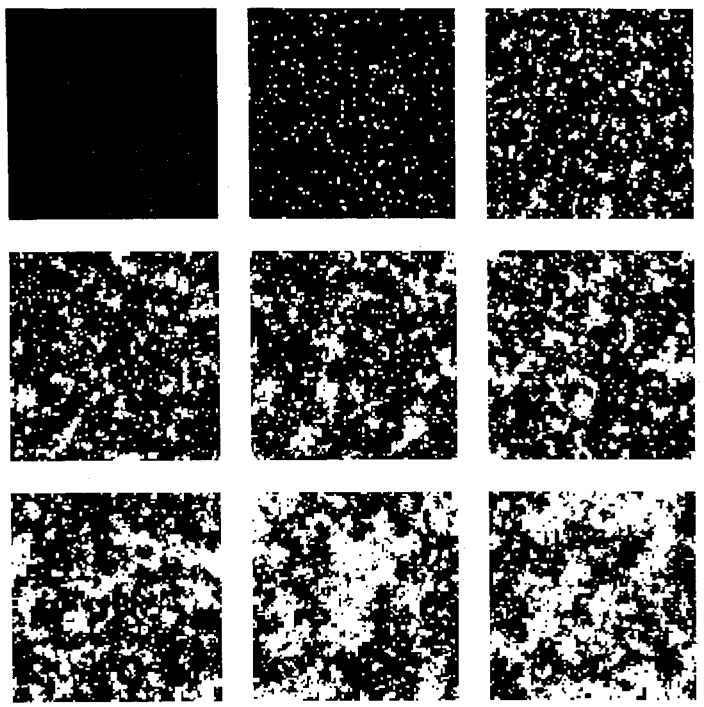

FIGURE 3.2 Nine snapshots of a 100×100 Ising model on a square lattice with $J = 1$ coming to equilibrium at a temperature $T = 2.4$ using the Metropolis algorithm. In these pictures the up-spins ($s_i = +1$) are represented by black squares and the down-spins ($s_i = -1$) by white ones. The starting configuration is one in which all the spins are pointing up. The progression of the figures is horizontally across the top row, then the middle row, then the bottom one. They show the lattice after 0, 1, 2, 4, 6, 10, 20, 40 and 100 times 100 000 steps of the simulation. In the last frame the system has reached equilibrium according to the criteria given in this section.

fall in this narrow range. Then, we assume, the Monte Carlo algorithm we have designed will ensure that it stays roughly within that range for the rest of the simulation—it should do since we designed the algorithm specifically to simulate the behaviour of the system at equilibrium. But it may take some time to find a state that lies within the correct range. In the version of the Metropolis algorithm which we have described here, we can only flip one spin at a time, and since we are choosing the spins we flip at random, it could take quite a while before we hit on the correct sequence of spins to

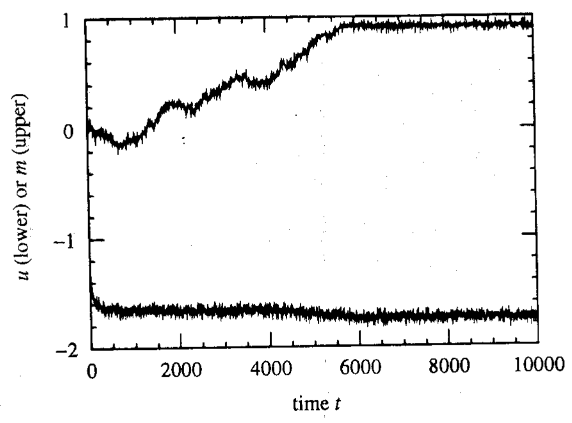

FIGURE 3.3 The magnetization (upper curve) and internal energy (lower curve) per site of a two-dimensional Ising model on a square lattice of 100×100 sites with $J = 1$ simulated using the Metropolis algorithm of Section 3.1. The simulation was started at $T = \infty$ (i.e., the initial states of the spins were chosen completely at random) and "cooled" to equilibrium at $T = 2.0$. Time is measured in Monte Carlo steps per lattice site, and equilibrium is reached after about 6000 steps per site (in other words, 6×10^7 steps altogether).

flip in order to get us to one of the states we want to be in. At the very least we can expect it to take about N Monte Carlo steps to reach a state in the appropriate energy range, where N is the number of spins on the lattice, since we need to allow every spin the chance to flip at least once. In Figure 3.2 we show a succession of states of a two-dimension Ising model on a square lattice of 100×100 spins with $J = 1$, as it is "warmed" up to a temperature $T = 2.4$ from an initial $T = 0$ state in which all the spins are aligned. In these pictures the $+1$ and -1 spins are depicted as black and white squares. By the time we reach the last frame out of nine, the system has equilibrated. The whole process takes on the order of 10^7 steps in this case.

However looking at pictures of the lattice is not a reliable way of gauging when the system has come to equilibrium. A better way, which takes very little extra effort, is to plot a graph of some quantity of interest, like the magnetization per spin m of the system or the energy of the system E, as a function of time from the start of the simulation. We have done this in Figure 3.3. (We will discuss the best ways of measuring these quantities in the next section, but for the moment let's just assume that we calculate them directly. For example, the energy of a given state can be calculated by feeding all the values of the spins s_i into the Hamiltonian, Equation (3.1).) It is not hard to guess simply by looking at this graph that the system came to equilibrium at around time $t = 6000$. Up until this point the energy and the magnetization are changing, but after this point they just fluctuate around a steady average value.

The horizontal axis in Figure 3.3 measures time in Monte Carlo steps *per lattice site*, which is the normal practice for simulations of this kind. The reason is that if time is measured in this way, then the average frequency with which any particular spin is selected for flipping is independent of the total number of spins N on the lattice. This average frequency is called the "attempt frequency" for the spin. In the simulation we are considering here the attempt frequency has the value 1. It is natural that we should arrange for the attempt frequency to be independent of the lattice size; in an experimental system, the rate at which spins or atoms or molecules change from one state to another does not depend on how many there are in the whole system. An atom in a tiny sample will change state as often as one in a sample the size of a house. Attempt frequencies are discussed further in Section 11.1.1.

When we perform N Monte Carlo steps—one for each spin in the system, on average—we say we have completed one **sweep** of the lattice. We could therefore also say that the time axis of Figure 3.3 was calibrated in sweeps.

Judging the equilibration of a system by eye from a plot such as Figure 3.3 is a reasonable method, provided we know that the system will come to equilibrium in a smooth and predictable fashion as it does in this case. The trouble is that we usually know no such thing. In many cases it is possible for the system to get stuck in some metastable region of its state space for a while, giving roughly constant values for all the quantities we are observing and so appearing to have reached equilibrium. In statistical mechanical terms, there can be a **local energy minimum** in which the system can remain temporarily, and we may mistake this for the **global energy minimum**, which is the region of state space that the equilibrium system is most likely to inhabit. (These ideas are discussed in more detail in the first few sections of Chapter 6.) To avoid this potential pitfall, we commonly adopt a different strategy for determining the equilibration time, in which we perform two different simulations of the same system, starting them in different initial states. In the case of the Ising model we might, for example, start one in the $T = 0$ state with all spins aligned, and one in the $T = \infty$ state with random spins. Or we could choose two different $T = \infty$ random-spin states. We should also run the two simulations with different "seeds" for the random number generator (see Section 16.1.2), to ensure that they take different paths to equilibrium. Then we watch the value of the magnetization or energy or other quantity in the two systems and when we see them reach the same approximately constant value, we deduce that both systems have reached equilibrium. We have done this for two 100×100 Ising systems in Figure 3.4. Again, we clearly see that it takes about 6000 Monte Carlo steps for the two systems to reach a consensus about the value of the magnetization. This technique avoids the problem mentioned above, since if one of the systems finds itself in some metastable region, and the other reaches equilibrium or gets stuck in another metastable region, this will be apparent from the graph, because the magnetization (or other quantity) will take different values for the two systems. Only in the unlikely event that the two systems coincidentally become trapped in the same metastable region (for example, if we choose two initial states that are too similar to one another) will we be misled into thinking they have reached equilibrium when they haven't. If we are worried about this possibility, we can run *three* different simulations from different starting points, or four, or five. Usually, however, two is sufficient.

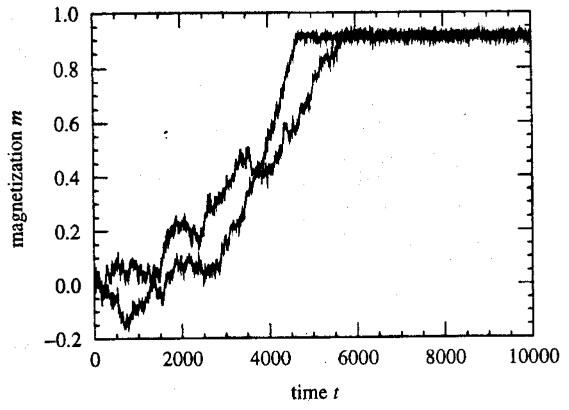

FIGURE 3.4 The magnetization of our 100×100 Ising model as a function of time (measured in Monte Carlo steps per lattice site) for two different simulations using the Metropolis algorithm. The two simulations were started off in two different $T = \infty$ (random-spin) states. By about time $t = 6000$ the two simulations have converged to the same value of the mean magnetization, within the statistical errors due to fluctuations, and so we conclude that both have equilibrated.

3.3 Measurement

Once we are sure the system has reached equilibrium, we need to measure whatever quantity it is that we are interested in. The most likely candidates for the Ising model are the energy and the magnetization of the system. As we pointed out above, the energy E_μ of the current state μ of the system can be evaluated directly from the Hamiltonian by substituting in the values of the spins s_i from our array of integers. However, this is not an especially efficient way of doing it, and there is a much better way. As part of our implementation of the Metropolis algorithm, you will recall we calculated the energy difference $\Delta E = E_\nu - E_\mu$ in going from state μ to state ν (see Equation (3.10)). So, if we know the energy of the current state μ, we can calculate the new energy when we flip a spin, using only a single addition:

$$E_\nu = E_\mu + \Delta E. \tag{3.11}$$

So the clever thing to do is to calculate the energy of the system from the Hamiltonian at the very start of the simulation, and then every time we flip a spin calculate the new energy from Equation (3.11) using the value of ΔE, which we have to calculate anyway.

Calculating the magnetization is even easier. The total magnetization M_μ of the whole system in state μ (as opposed to the magnetization *per spin*—we'll calculate that in a moment), is given by the sum

$$M_\mu = \sum_i s_i^\mu. \tag{3.12}$$

As with the energy, it is not a shrewd idea to evaluate the magnetization directly from this sum every time we want to know it. It is much better to notice that only one spin k flips at a time in the Metropolis algorithm, so the change of magnetization from state μ to state ν is

$$\Delta M = M_\nu - M_\mu = \sum_i s_i^\nu - \sum_i s_i^\mu = s_k^\nu - s_k^\mu = 2s_k^\nu, \tag{3.13}$$

where the last equality follows from Equation (3.9). Thus, the clever way to evaluate the magnetization is to calculate its value at the beginning of the simulation, and then make use of the formula

$$M_\nu = M_\mu + \Delta M = M_\mu + 2s_k^\nu \tag{3.14}$$

every time we flip a spin.[2]

[2]However, to be absolutely fair, we should point out that doing this involves performing at least one addition operation every time we flip a spin, or one addition every $\bar{A}^{-1}$ steps, where $\bar{A}$ is the mean acceptance ratio. Direct evaluation of Equation (3.12) on the other hand involves N additions every time we want to know the magnetization. Thus, if we want to make measurements less often than once every $N/\bar{A}$ steps, it may pay to use the direct method rather than employing Equation (3.14). Similar considerations apply to the measurement of the energy also.

Given the energy and the magnetization of our Ising system at a selection of times during the simulation, we can average them to find the estimators of the internal energy and average magnetization. Then dividing these figures by the number of sites N gives us the internal energy and average magnetization per site.

We can also average the squares of the energy and magnetization to find quantities like the specific heat and the magnetic susceptibility:

$$c = \frac{\beta^2}{N}(\langle E^2\rangle - \langle E\rangle^2), \tag{3.15}$$

$$\chi = \beta N(\langle m^2\rangle - \langle m\rangle^2). \tag{3.16}$$

(See Equations (1.36) and (1.37). Note that we have set $k = 1$ again.)

In order to average quantities like E and M, we need to know how long a run we have to average them over to get a good estimate of their expectation values. One simple solution would again be just to look at a graph like Figure 3.3 and guess how long we need to wait. However (as you might imagine) this is not a very satisfactory solution. What we really need is a measure of the **correlation time** τ of the simulation. The correlation time is a measure of how long it takes the system to get from one state to another one which is significantly different from the first, i.e., a state in which the number of spins which are the same as in the initial state is no more than what you would expect to find just by chance. (We will give a more rigorous definition in a moment.) There are a number of ways to estimate the correlation time. One that is sometimes used is just to assume that it is equal to the equilibration time. This is usually a fairly safe assumption:[3] usually the equilibration time is considerably longer than the correlation time, $\tau_{\text{eq}} > \tau$, because two states close to equilibrium are qualitatively more similar than a state far from equilibrium (like the $T = 0$ or $T = \infty$ states we suggested for starting this simulation with) and one close to equilibrium. However, this is again a rather unrigorous supposition, and there are more watertight ways to estimate τ. The most direct of these is to calculate the "time-displaced autocorrelation function" of some property of the model.

3.3.1 Autocorrelation functions

Let us take the example of the magnetization m of our Ising model. The **time-displaced autocorrelation** $\chi(t)$ of the magnetization is given by

$$\begin{aligned}\chi(t) &= \int dt' \,[m(t') - \langle m\rangle][m(t'+t) - \langle m\rangle] \\ &= \int dt' \,[m(t')m(t'+t) - \langle m\rangle^2].\end{aligned} \tag{3.17}$$

[3]In particular, it works fine for the Metropolis simulation of the Ising model which we are considering here.

where $m(t)$ is the instantaneous value of the magnetization at time t and $\langle m\rangle$ is the average value. This is rather similar to the connected correlation function which we defined in Equation (1.26). That measured the correlation between the values of a quantity (such as the magnetization) on two different sites, i and j, on the lattice. The autocorrelation gives us a similar measure of the correlation at two different times, one an interval t later than the other. If we measure the difference between the magnetization $m(t')$ at time t' and its mean value, and then we do the same thing at time $t'+t$, and we multiply them together, we will get a positive value if they were fluctuating in the same direction at those two times, and a negative one if they were fluctuating in opposite directions. If we then integrate over time as in Equation (3.17), then $\chi(t)$ will take a non-zero value if on average the fluctuations are correlated, or it will be zero if they are not. For our Metropolis simulation of the Ising model it is clear that if we measure the magnetization at two times just a single Monte Carlo step apart, the values we get will be very similar, so we will have a large positive autocorrelation. On the other hand, for two times a long way apart the magnetizations will probably be totally unrelated, and their autocorrelation will be close to zero. Ideally, we should calculate $\chi(t)$ by integrating over an infinite time, but this is obviously impractical in a simulation, so we do the best we can and just sum over all the measurements of m that we have, from beginning to end of our run. Figure 3.5 shows the magnetization autocorrelation of our 100×100 Ising model at temperature $T = 2.4$ and interaction energy $J = 1$, calculated in exactly this manner using results from our Metropolis Monte Carlo simulation. As we can see, the autocorrelation does indeed drop from a significant non-zero value at short times t towards zero at very long times. In this case, we have divided $\chi(t)$ by its value $\chi(0)$ at $t = 0$, so that its maximum value is one. The typical time-scale (if there is one) on which it falls off is a measure of the correlation time τ of the simulation. In fact, this is the *definition* of the correlation time. It is the typical time-scale on which the autocorrelation drops off; the autocorrelation is expected to fall off exponentially at long times thus:

$$\chi(t) \sim e^{-t/\tau}. \tag{3.18}$$

(In the next section we show why it should take this form.) With this definition, we see that in fact there is still a significant correlation between two samples taken a correlation time apart: at time $t = \tau$ the autocorrelation function, which is a measure of the similarity of the two states, is only a factor of $1/e$ down from its maximum value at $t = 0$. If we want truly independent samples then, we may want to draw them at intervals of greater than one correlation time. In fact, the most natural definition of statistical independence turns out to be samples drawn at intervals of 2τ. We discuss this point further in Section 3.4.1.

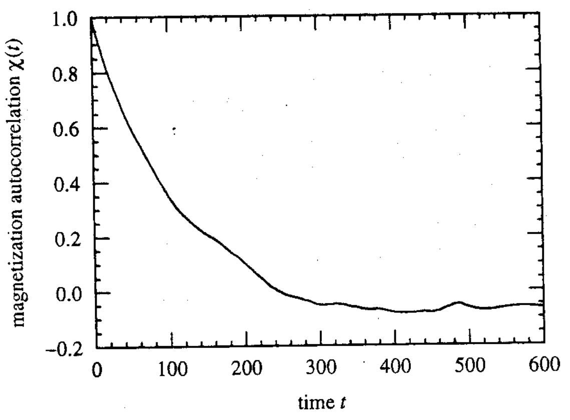

FIGURE 3.5 The magnetization autocorrelation function $\chi(t)$ for a two-dimensional Ising model at temperature $T = 2.4$ on a square lattice of 100×100 sites with $J = 1$ simulated using the Metropolis algorithm of Section 3.1. Time is measured in Monte Carlo steps per site.

We can make a reasonable estimate of τ by eye from Figure 3.5. At a guess we'd probably say τ was about 100 in this case. This is an accurate enough figure for estimating how long a Monte Carlo run we need to do in order to get decent statistics. It tells us that we expect to get a new independent spin configuration about once every $2\tau = 200$ sweeps of the lattice in our simulation. So if we want, say, 10 independent measurements of the magnetization, we need to run our Metropolis algorithm for about 2000 sweeps after equilibration, or 2×10^7 Monte Carlo steps. If we want 100 measurements we need to do 2×10^8 steps. In general, if a run lasts a time t_{max}, then the number of independent measurements we can get out of the run, after waiting a time τ_{eq} for equilibration, is on the order of

$$n = \frac{t_{\text{max}}}{2\tau}. \tag{3.19}$$

It is normal practice in a Monte Carlo simulation to make measurements at intervals of less than the correlation time. For example, in the case of the Metropolis algorithm we might make one measurement every sweep of the lattice. Thus the total number of measurements we make of magnetization or energy (or whatever) during the run is usually greater than the number of independent measurements. There are a number of reasons why we do it this way. First of all, we usually don't know what the correlation time is until after the simulation has finished, or at least until after it has run

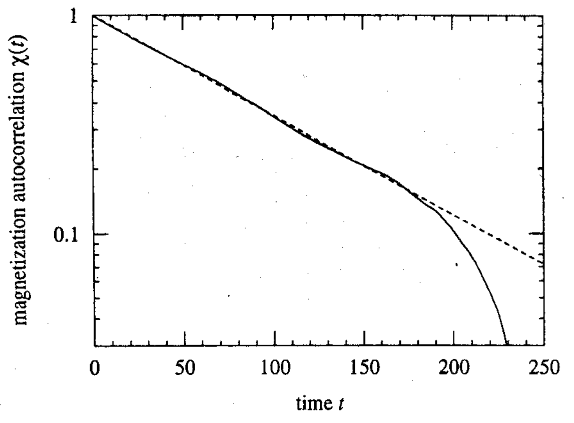

FIGURE 3.6 The autocorrelation function of Figure 3.5 replotted on semi-logarithmic axes. The dashed line is a straight line fit which yields a figure of $\tau = 95 \pm 5$ in this case. Note that the horizontal scale is not the same as in Figure 3.5.

for a certain amount of time, and we want to be sure of having at least one measurement every two correlation times. Another reason is that we want to be able to calculate the autocorrelation function for times less than a correlation time, so that we can use it to make an accurate estimate of the correlation time. If we only had one measurement every 2τ, we wouldn't be able to calculate τ with any accuracy at all.

If we want a more reliable figure for τ, we can replot our autocorrelation function on semi-logarithmic axes as we have done in Figure 3.6, so that the slope of the line gives us the correlation time. Then we can estimate τ by fitting the straight-line portion of the plot using a least-squares method. The dotted line in the figure is just such a fit and its slope gives us a figure of $\tau = 95 \pm 5$ for the correlation time in this case.

An alternative is to calculate the **integrated correlation time**.[4] If we assume that Equation (3.18) is accurate for all times t then

$$\int_0^\infty \frac{\chi(t)}{\chi(0)}\, dt = \int_0^\infty e^{-t/\tau}\, dt = \tau. \tag{3.20}$$

This form has a number of advantages. First, it is often easier to apply Equation (3.20) than it is to perform the exponential fit to the autocorrelation

[4]This is a rather poor name for this quantity, since it is not the correlation time that is integrated but the autocorrelation function. However, it is the name in common use so we use it here too.

function. Second, the method for estimating τ illustrated in Figure 3.6 is rather sensitive to the range over which we perform the fit. In particular, the very long time behaviour of the autocorrelation function is often noisy, and it is important to exclude this portion from the fitted data. However, the exact point at which we truncate the fit can have quite a large effect on the resulting value for τ. The integrated correlation time is much less sensitive to the way in which the data are truncated, although it is by no means perfect either, since, as we will demonstrate in Section 3.3.2, Equation (3.18) is only strictly correct for long times t, and we introduce an uncontrolled error by assuming it to be true for all times. On the other hand, the direct fitting method also suffers from this problem, unless we only perform our fit over the exact range of times for which true exponential behaviour is present. Normally, we don't know what this range is, so the fitting method is no more accurate than calculating the integrated correlation time. Usually then, Equation (3.20) is the method of choice for calculating τ. Applying it to the data from Figure 3.6 gives a figure of $\tau = 86 \pm 5$, which is in moderately good agreement with our previous figure.

The autocorrelation function in Figure 3.5 was calculated directly from a discrete form of Equation (3.17). If we have a set of samples of the magnetization $m(t)$ measured at evenly-spaced times up to some maximum time t_{max}, then the correct formula for the autocorrelation function is[5]

$$\begin{aligned} \chi(t) = {} & \frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t} m(t')\, m(t'+t) \\ & - \frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t} m(t') \times \frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t} m(t'+t). \end{aligned} \tag{3.21}$$

Notice how we have evaluated the mean magnetization m in the second term using the same subsets of the data that we used in the first term. This is not strictly speaking necessary, but it makes $\chi(t)$ a little better behaved. In Figure 3.5 we have also normalized $\chi(t)$ by dividing throughout by $\chi(0)$, but this is optional. We've just done it for neatness.

Note that one should be careful about using Equation (3.21) to evaluate $\chi(t)$ at long times. When t gets close to $t_{\max}$, the upper limit of the sums becomes small and we end up integrating over a rather small time interval to get our answer. This means that the statistical errors in $\chi(t)$ due to the random nature of the fluctuations in $m(t)$ may become large. A really satisfactory simulation would always run for many correlation times, in which case we will probably not be interested in the very tails of $\chi(t)$, since the correlations will have died away by then, by definition. However, it is not

[5]In fact, this formula differs from (3.17) by a multiplicative constant, but this makes no difference as far as the calculation of the correlation time is concerned.

always possible, because we only have limited computer resources, to perform simulations as long as we would like, and one should always be aware that errors of this type can crop up with shorter data sets.

Calculating the autocorrelation function from Equation (3.21) takes time of order n^2, where n is the number of samples. For most applications, this is not a problem. Even for simulations where several thousand samples are taken, the time needed to evaluate the autocorrelation is only a few seconds on a modern computer, and the simplicity of the formulae makes their programming very straightforward, which is a big advantage—people-time is usually more expensive than computer-time. However, sometimes it is desirable to calculate the autocorrelation function more quickly. This is particularly the case when one needs to do it very often during the course of a calculation, for some reason. If you need to calculate an autocorrelation a thousand times, and each time takes a few seconds on the computer, then the seconds start to add up. In this case, at the expense of rather greater programming effort, we can often speed up the process by the following trick. Instead of calculating $\chi(t)$ directly, we calculate the Fourier transform $\widetilde{\chi}(\omega)$ of the autocorrelation and then we invert the Fourier transform to get $\chi(t)$. The Fourier transform is related to the magnetization as follows:

$$\begin{aligned}\widetilde{\chi}(\omega) &= \int \mathrm{d}t\, \mathrm{e}^{\mathrm{i}\omega t} \int \mathrm{d}t'\, [m(t') - \langle m\rangle][m(t'+t) - \langle m\rangle] \\ &= \int \mathrm{d}t \int \mathrm{d}t'\, \mathrm{e}^{-\mathrm{i}\omega t'} [m(t') - \langle m\rangle]\, \mathrm{e}^{\mathrm{i}\omega(t'+t)} [m(t'+t) - \langle m\rangle] \\ &= \widetilde{m}'(\omega)\, \widetilde{m}'(-\omega) = \left|\widetilde{m}'(\omega)\right|^2, \end{aligned} \qquad (3.22)$$

where $\widetilde{m}'(\omega)$ is the Fourier transform of $m'(t) \equiv m(t) - \langle m\rangle$.[6]

So all we need to do is calculate the Fourier transform of $m'(t)$ and feed it into this formula to get $\widetilde{\chi}(\omega)$. The advantage in doing this is that the Fourier transform can be evaluated using the so-called **fast Fourier transform** or FFT algorithm, which was given by Cooley and Tukey in 1965. This is a standard algorithm which can evaluate the Fourier transform in a time which goes like $n \log n$, where n is the number of measurements of the magnetization.[7] Furthermore, there exist a large number of ready-made computer software packages that perform FFTs. These packages have

[6]Since $m(t)$ and $m'(t)$ differ only by a constant, $\widetilde{m}(\omega)$ and $\widetilde{m}'(\omega)$ differ only in their $\omega = 0$ component (which is zero in the latter case, but may be non-zero in the former). For this reason, it is often simplest to calculate $\widetilde{m}'(\omega)$ by first calculating $\widetilde{m}(\omega)$ and then just setting the $\omega = 0$ component to zero.

[7]On a technical note, if we simply apply the FFT algorithm directly to our magnetization data, the result produced is the Fourier transform of an infinite periodic repetition of the data set, which is not quite what we want in Equation (3.22). A simple way of getting around this problem is to add n zeros to the end of the data set before we perform the transform. It can be shown that this then gives a good estimate of the autocorrelation function (Futrelle and McGinty 1971).

been written very carefully by people who understand the exact workings of the computer and are designed to be as fast as possible at performing this particular calculation, so it usually saves us much time and effort to make use of the programs in these packages. Having calculated $\widetilde{\chi}(\omega)$, we can then invert the Fourier transform, again using a streamlined inverse FFT routine which also runs in time proportional to $n \log n$, and so recover the function $\chi(t)$.

We might imagine that, if we wanted to calculate the integrated correlation time, Equation (3.20), we could avoid inverting the Fourier transform, since

$$\widetilde{\chi}(0) = \int_0^\infty \chi(t)\, \mathrm{d}t, \qquad (3.23)$$

and thus

$$\tau = \frac{\widetilde{\chi}(0)}{\chi(0)}, \qquad (3.24)$$

where $\chi(0)$ is simply the magnetization fluctuation

$$\chi(0) = \langle m^2\rangle - \langle m\rangle^2. \qquad (3.25)$$

However, you should avoid calculating τ this way because, as footnote 6 on the previous page makes clear, $\widetilde{\chi}(0)$ is zero when calculated directly for finite datasets. Equation (3.24) is only applicable if $\langle m\rangle$ is calculated over a much longer run than $\widetilde{\chi}(\omega)$.

3.3.2 Correlation times and Markov matrices

The techniques outlined in the previous section are in most cases quite sufficient for estimating correlation times in Monte Carlo simulations. However, we have simplified the discussion somewhat by supposing there to be only one correlation time in the system. In real life there are as many correlation times as there are states of the system, and the interplay between these different times can sometimes cause the methods of the last section to give inaccurate results. In this section we look at these issues in more detail. The reader who is interested only in how to calculate a rough measure of τ could reasonably skip this section.

In Section 2.2.3 we showed that the probabilities $w_\mu(t)$ and $w_\mu(t+1)$ of being in a particular state μ at consecutive Monte Carlo steps are related by the Markov matrix $\mathbf{P}$ for the algorithm. In matrix notation we wrote

$$\mathbf{w}(t+1) = \mathbf{P} \cdot \mathbf{w}(t), \qquad (3.26)$$

where $\mathbf{w}$ is the vector whose elements are the probabilities w_μ (see Equation (2.9)). By iterating this equation from time $t = 0$ we can then show that

$$\mathbf{w}(t) = \mathbf{P}^t \cdot \mathbf{w}(0). \qquad (3.27)$$

Now $\mathbf{w}(0)$ can be expressed as a linear combination of the right eigenvectors $\mathbf{v}_i$ of $\mathbf{P}$ thus:[8]

$$\mathbf{w}(0) = \sum_i a_i \mathbf{v}_i, \tag{3.28}$$

where the quantities a_i are coefficients whose values depend on the configuration of the system at $t = 0$. Then

$$\mathbf{w}(t) = \mathbf{P}^t \cdot \sum_i a_i \mathbf{v}_i = \sum_i a_i \lambda_i^t \mathbf{v}_i, \tag{3.29}$$

where λ_i is the eigenvalue of $\mathbf{P}$ corresponding to the eigenvector $\mathbf{v}_i$. As $t \to \infty$, the right-hand side of this equation will be dominated by the term involving the largest eigenvalue λ_0 of the Markov matrix. This means that in the limit of long times, the probability distribution $\mathbf{w}(t)$ becomes proportional to $\mathbf{v}_0$, the eigenvector corresponding to the largest eigenvalue. We made use of this result in Section 2.2.3 to demonstrate that $\mathbf{w}(t)$ tends to the Boltzmann distribution at long times.

Now suppose that we are interested in knowing the value of some observable quantity Q, such as the magnetization. The expectation value of this quantity at time t can be calculated from the formula

$$Q(t) = \sum_\mu w_\mu(t)\, Q_\mu \tag{3.30}$$

or

$$Q(t) = \mathbf{q} \cdot \mathbf{w}(t), \tag{3.31}$$

where $\mathbf{q}$ is the vector whose elements are the values Q_μ of the quantity in the various states of the system. Substituting Equation (3.29) into (3.31) we then get

$$Q(t) = \sum_i a_i \lambda_i^t \mathbf{q} \cdot \mathbf{v}_i = \sum_i a_i \lambda_i^t q_i. \tag{3.32}$$

Here $q_i \equiv \mathbf{q} \cdot \mathbf{v}_i$ is the expectation value of Q in the i^{th} eigenstate. The long time limit $Q(\infty)$ of this expression is also dominated by the largest eigenvalue λ_0 and is proportional to q_0. If we now define a set of quantities τ_i thus:

$$\tau_i = -\frac{1}{\log \lambda_i} \tag{3.33}$$

for all $i \neq 0$, then Equation (3.32) can be written

$$Q(t) = Q(\infty) + \sum_{i \neq 0} a_i q_i e^{-t/\tau_i}. \tag{3.34}$$

[8]$\mathbf{P}$ is in general not symmetric, so its right and left eigenvectors are not the same.

The quantities τ_i are the correlation times for the system, as we can demonstrate by showing that they also govern the decay of the autocorrelation function. Noting that the long-time limit $Q(\infty)$ of the expectation of Q is none other than the equilibrium expectation $\langle Q \rangle$, we can write the autocorrelation of Q as the correlation between the expectations at zero time and some later time t thus:

$$\chi(t) = [Q(0) - Q(\infty)][Q(t) - Q(\infty)] = \sum_{i \neq 0} b_i e^{-t/\tau_i}, \tag{3.35}$$

with

$$b_i = \sum_{j \neq 0} a_i a_j q_i q_j. \tag{3.36}$$

Equation (3.35) is the appropriate generalization of Equation (3.18) to all times t (not just long ones).

As we said, there are as many correlation times as there are states of the system since that is the rank of the matrix $\mathbf{P}$. (Well, strictly there are as many of them as the rank of the matrix *less one*, since there is no τ_0 corresponding to the highest eigenvalue. There are $2^N - 1$ correlation times in the case of the Ising model, for example. However, the rank of the matrix is usually very large, so let's not quibble over one correlation time.) The longest of these correlation times is τ_1, the one which corresponds to the second largest eigenvalue of the matrix. This is the correlation time we called τ in the last section. Clearly, for large enough times t, this will be the only correlation time we need to worry about, since all the other terms in Equation (3.35) will have decayed away to insignificance. (This is how Equation (3.18) is derived.) However, depending on how close together the higher eigenvalues of the Markov matrix are, and how long our simulation runs for, we may or may not be able to extract reliable results for τ_1 by simply ignoring all the other terms. In general the most accurate results are obtained by fitting our autocorrelation function to a sum of a small number of decaying exponentials, of the form of Equation (3.35), choosing values for the quantities b_i by a least-squares or similar method. In work on the three-dimensional Ising model, for example, Wansleben and Landau (1991) showed that including three terms was sufficient to get a good fit to the magnetization autocorrelation function, and thus get an accurate measure of the longest correlation time $\tau \equiv \tau_1$. In studies of the dynamical properties of statistical mechanical models, this is the most correct way to measure the correlation time. Strictly speaking it gives only a lower bound on τ since it is always possible that correlations exist beyond the longest times that one can measure. However, in practice it usually gives good results.

3.4 Calculation of errors

Normally, as well as measuring expectation values, we also want to calculate the errors on those values, so that we have an idea of how accurate they are. As with experiments, the errors on Monte Carlo results divide into two classes: **statistical errors** and **systematic errors**.[9] Statistical errors are errors which arise as a result of random changes in the simulated system from measurement to measurement—thermal fluctuations, for example—and they can be estimated simply by taking many measurements of the quantity we are interested in and calculating the spread of the values. Systematic errors on the other hand, are errors due to the procedure we have used to make the measurements, and they affect the whole simulation. An example is the error introduced by waiting only a finite amount of time for our system to equilibrate. (Ideally, we should allow an infinite amount of time for this, in order to be sure the system has completely equilibrated. However, this, of course, is not practical.)

3.4.1 Estimation of statistical errors

In a Monte Carlo calculation the principal source of statistical error in the measured value of a quantity is usually the fluctuation of that quantity from one time step to the next. This error is inherent in the Monte Carlo method. As the name "Monte Carlo" itself makes clear, there is an innate randomness and statistical nature to Monte Carlo calculations. (In Chapter 6 on glassy spin models, we will see another source of statistical error: "sample-to-sample" fluctuations in the actual system being simulated. However, for the simple Ising model we have been considering, thermal fluctuations are the only source of statistical error. All other errors fall into the category of systematic errors.) It is often straightforward to estimate the statistical error in a measured quantity, since the assumption that the error is statistical—i.e., that it arises through random deviations in the measured value of the quantity—implies that we can estimate the true value by taking the mean of several different measurements, and that the error on that estimate is simply the error on the mean. Thus, if we are performing the Ising model simulation described in the last section, and we make n measurements m_i of the magnetization of the system during a particular run, then our best estimate of the true thermal average of the magnetization is the mean $\overline{m}$ of those n measurements (which is just the estimator of m, as defined in Section 2.1),

[9]Monte Carlo simulations are in many ways rather similar to experiments. It often helps to regard them as "computer experiments", and analyse the results in the same way as we would analyse the results of a laboratory experiment.

and our best estimate of the standard deviation on the mean is given by[10]

$$\sigma = \sqrt{\frac{\frac{1}{n}\sum_{i=0}^{n}(m_i - \overline{m})^2}{n-1}} = \sqrt{\frac{1}{n-1}(\overline{m^2} - \overline{m}^2)}. \qquad (3.37)$$

This expression assumes that our samples m_i are statistically independent, which in general they won't be. As we pointed out in Section 3.3.1, it is normal to sample at intervals of less than a correlation time, which means that successive samples will in general be correlated. A simple and usually perfectly adequate solution to this problem is to use the value of n given by Equation (3.19), rather than the actual number of samples taken. In fact, it can be shown (Müller-Krumbhaar and Binder 1973) that the correct expression for σ in terms of the actual number of samples is

$$\sigma = \sqrt{\frac{1 + 2\tau/\Delta t}{n-1}(\overline{m^2} - \overline{m}^2)}, \qquad (3.38)$$

where τ is the correlation time and Δt is the time interval at which the samples were taken. Clearly this becomes equal to Equation (3.37) when $\Delta t \gg \tau$, but more often we have $\Delta t \ll \tau$. In this case, we can ignore the 1 in the numerator of Equation (3.38). Noting that for a run of length $t_{\max}$ (after equilibration) the interval Δt is related to the total number of samples by

$$n = \frac{t_{\max}}{\Delta t}, \qquad (3.39)$$

we then find that for large n

$$\sigma = \sqrt{\frac{2\tau}{t_{\max}}(\overline{m^2} - \overline{m}^2)}, \qquad (3.40)$$

which is the same result as we would get by simply using Equation (3.19) for n in Equation (3.37). This in fact was the basis for our assertion in Section 3.3.1 that the appropriate sampling interval for getting independent samples was twice the correlation time. Note that the value of σ in Equation (3.40) is independent of the value of Δt, which means we are free to choose Δt in whatever way is most convenient.

3.4.2 The blocking method

There are some cases where it is either not possible or not straightforward to estimate the error in a quantity using the direct method described in the last

[10]The origin of the $n - 1$ in this and following expressions for error estimates is a little obscure. The curious reader is referred to any good book on data analysis for an explanation, such as Bevington and Robinson (1992).

section. This happens when the result we want is not merely the average of some measurement repeated many times over the course of the simulation, as the magnetization is, but is instead derived in some more complex way from measurements we make during the run. An example is the specific heat c, Equation (3.15), which is inherently an average macroscopic quantity. Unlike the magnetization, the specific heat is not defined at a single time step in the simulation. It is only defined in terms of averages of many measurements of E and E^2 over a longer period of time. We might imagine that we could calculate the error on $\langle E \rangle$ and the error on $\langle E^2 \rangle$ using the techniques we employed for the magnetization, and then combine them in some fashion to give an estimate of the error in c. But this is not as straightforward as it seems at first, since the errors in these two quantities are correlated—when $\langle E \rangle$ goes up, so does $\langle E^2 \rangle$. It is possible to do the analysis necessary to calculate the error on c in this fashion. However, it is not particularly simple, and there are other more general methods of error estimation which lend themselves to this problem. As the quantities we want to measure become more complex, these methods—"blocking", the "bootstrap" and the "jackknife"—will save us a great deal of effort in estimating errors. We illustrate these methods here for the case of the specific heat, though it should be clear that they are applicable to almost any quantity that can be measured in a Monte Carlo simulation.

The simplest of our general-purpose error estimation methods is the **blocking method**. Applied to the specific heat, the idea is that we take the measurements of E that we made during the simulation and divide them into several groups, or **blocks**. We then calculate c separately for each block, and the spread of values from one block to another gives us an estimate of the error. To see how this works, suppose we make 200 measurements of the energy during our Ising model simulation, and then split those into 10 groups of 20 measurements. We can evaluate the specific heat from Equation (3.15) for each group and then find the mean of those 10 results exactly as we did for the magnetization above. The error on the mean is given again by Equation (3.37), except that n is now replaced by the number n_b of blocks, which would be 10 in our example. This method is intuitive, and will give a reasonable estimate of the order of magnitude of the error in a quantity such as c. However, the estimates it gives vary depending on the number of different blocks you divide your data up into, with the smallest being associated with large numbers of blocks, and the largest with small numbers of blocks, so it is clearly not a very rigorous method. A related but more reliable method, which can be used for error estimation in a wide variety of different circumstances, is the **bootstrap method**, which we now describe.

3.4.3 The bootstrap method

The bootstrap method is a **resampling method**. Applied to our problem of calculating the specific heat for the Ising model, it would work like this. We take our list of measurements of the energy of the model and from the n numbers in this list we pick out n at random. (Strictly, n should be the number of *independent* measurements. In practice the measurements made are usually not all independent, but luckily it transpires that the bootstrap method is not much affected by this difference, a point which is discussed further below.) We specifically allow ourselves to pick the same number twice from the list, so that we end up with n numbers each of which appears on the original list, and some of which may be duplicates of one another. (In fact, if you do your sums, you can show that about a fraction $1 - 1/e \simeq 63\%$ of the numbers will be duplicates.) We calculate the specific heat from these n numbers just as we would normally, and then we repeat the process, picking (or **resampling**) another n numbers at random from the original measurements. It can be shown (Efron 1979) that after we have repeated this calculation several times, the standard deviation of the distribution in the results for c is a measure of the error in the value of c. In other words if we make several of these "bootstrap" calculations of the specific heat, our estimate of the error σ is given by

$$\sigma = \sqrt{\overline{c^2} - \bar{c}^2}. \tag{3.41}$$

Notice that there is no extra factor of $1/(n-1)$ here as there was in Equation (3.37). (It is clear that the latter would not give a correct result, since it would imply that our estimate of the error could be reduced by simply resampling our data more times.)

As we mentioned, it is not necessary for the working of the bootstrap method that all the measurements made be independent in the sense of Section 3.3.1 (i.e., one every two correlation times or more). As we pointed out earlier, it is more common in a Monte Carlo simulation to make measurements at comfortably short intervals throughout the simulation so as to be sure of making at least one every correlation time or so and then calculate the number of independent measurements made using Equation (3.19). Thus the number of samples taken usually exceeds the number which actually constitute independent measurements. One of the nice things about the bootstrap method is that it is not necessary to compensate for this difference in applying the method. You get fundamentally the same estimate of the error if you simply resample your n measurements from the entire set of measurements that were made. In this case, still about 63% of the samples will be duplicates of one another, but many others will effectively be duplicates as well because they will be measurements taken at times less than a correlation time apart. Nonetheless, the resulting estimate of σ is the same.

The bootstrap method is a good general method for estimating errors in quantities measured by Monte Carlo simulation. Although the method initially met with some opposition from mathematicians who were not convinced of its statistical validity, it is now widely accepted as giving good estimates of errors (Efron 1979).

3.4.4 The jackknife method

A slightly different but related method of error estimation is the **jackknife**. For this method, unlike the bootstrap method, we really do need to choose n independent samples out of those that were made during the run, taking one approximately every two correlation times or more. Applying the jackknife method to the case of the specific heat, we would first use these samples to calculate a value c for the specific heat. Now however, we also calculate n other estimates c_i as follows. We take our set of n measurements, and we remove the first one, leaving $n-1$, and we calculate the specific heat c_1 from that subset. Then we put the first one back, but remove the second and calculate c_2 from that subset, and so forth. Each c_i is the specific heat calculated with the i^{th} measurement of the energy removed from the set, leaving $n-1$ measurements. It can then be shown that an estimate of the error in our value of c is

$$\sigma = \sqrt{\sum_{i=1}^{n} (c_i - c)^2}, \tag{3.42}$$

where c is our estimate of the specific heat using all the data.[11]

Both the jackknife and the bootstrap give good estimates of errors for large data sets, and as the size of the data set becomes infinite they give exact estimates. Which one we choose in a particular case usually depends on how much work is involved applying them. In order to get a decent error estimate from the bootstrap method we usually need to take at least 100 resampled sets of data, and 1000 would not be excessive. (100 would give the error to a bit better than 10% accuracy.) With the jackknife we have to recalculate the quantity we are interested in exactly n times to get the error estimate. So, if n is much larger than 100 or so, the bootstrap is probably the more efficient. Otherwise, we should use the jackknife.[12]

[11] In fact, it *is* possible to use the jackknife method with samples taken at intervals Δt less than 2τ. In this case we just reduce the sum inside the square root by a factor of $\Delta t/2\tau$ to get an estimate of σ which is independent of the sampling interval.

[12] For a more detailed discussion of these two methods, we refer the interested reader to the review article by Efron (1979).

3.4.5 Systematic errors

Just as in experiments, systematic errors are much harder to gauge than statistical ones; they don't show up in the fluctuations of the individual measurements of a quantity. (That's why they are systematic errors.) The main source of systematic error in the Ising model simulation described in this chapter is the fact that we wait only a finite amount of time for the system to equilibrate. There is no good general method for estimating systematic errors; each source of error has to be considered separately and a strategy for estimating it evolved. This is essentially what we were doing when we discussed ways of estimating the equilibration time for the Metropolis algorithm. Another possible source of systematic error would be not running the simulation for a long enough time after equilibration to make good independent measurements of the quantities of interest. When we discussed methods for estimating the correlation time τ, we were dealing with this problem. In the later sections of this chapter, and indeed throughout this book, we will discuss methods for estimating and controlling systematic errors as they crop up in various situations.

3.5 Measuring the entropy

We have described how we go about measuring the internal energy, specific heat, magnetization and magnetic susceptibility from our Metropolis Monte Carlo simulation of the Ising model. However, there are three quantities of interest which were mentioned in Chapter 1 which we have not yet discussed: the free energy F, the entropy S and the correlation function $G_c^{(2)}(i,j)$. The correlation function we will consider in the next section. The other two we consider now.

The free energy and the entropy are related by

$$F = U - TS \tag{3.43}$$

so that if we can calculate one, we can easily find the other using the known value of the total internal energy U. Normally, we calculate the entropy, which we do by integrating the specific heat over temperature as follows.

We can calculate the specific heat of our system from the fluctuations in the internal energy as described in Section 3.3. Moreover, we know that the specific heat C is equal to

$$C = T\frac{\mathrm{d}S}{\mathrm{d}T}. \tag{3.44}$$

Thus the entropy $S(T)$ at temperature T is

$$S(T) = S(T_0) + \int_{T_0}^{T} \frac{C}{T}\,\mathrm{d}T. \tag{3.45}$$

If we are only interested in how S varies with T, then it is not necessary to know the value of the integration constant $S(T_0)$ and we can give it any value we like. If we want to know the absolute value of S, then we have to fix $S(T_0)$ by choosing T_0 to be some temperature at which we know the value of the entropy. The conventional choice, known as the third law of thermodynamics, is to make the entropy zero[13] when $T_0 = 0$. In other words

$$S(T) = \int_0^T \frac{C}{T}\,\mathrm{d}T. \tag{3.46}$$

As with the other quantities we have discussed, we often prefer to calculate the entropy per spin $s(T)$ of our system, which is given in terms of the specific heat per spin c by

$$s(T) = \int_0^T \frac{c}{T}\,\mathrm{d}T. \tag{3.47}$$

Of course, evaluating either of these expressions involves calculating the specific heat over a range of temperatures up to the temperature we are interested in, at sufficiently small intervals that its variation with T is well approximated. Then we have to perform a numerical integral using, for example, the trapezium rule or any of a variety of more sophisticated integration techniques (see, for instance, Press *et al.* 1988). Calculating the entropy (or equivalently the free energy) of a system is therefore a more complex task than calculating the internal energy, and may use up considerably more computer time. On the other hand, if we are interested in probing the behaviour of our system over a range of temperatures anyway (as we often are), we may as well make use of the data to calculate the entropy; the integration is a small extra computational effort to make by comparison with the simulation itself.

The integration involved in the entropy calculation can give problems. In particular, if there is any very sharp behaviour in the curve of specific heat as a function of temperature, we may miss it in our simulation, which would give the integral the wrong value. This is a particular problem near "phase transitions", where the specific heat often diverges (see Section 3.7.1).

3.6 Measuring correlation functions

One other quantity which we frequently want to measure is the two-point connected correlation function $G_c^{(2)}(i,j)$. Let us see how we would go about

[13] Systems which have more than one ground state may violate the third law. This point is discussed in more detail in Section 7.1.2.

calculating this correlation function for the Ising model. The most straightforward way is to evaluate it directly from the definition, Equation (1.26), using the values of the spins s_i from our simulation:

$$G_c^{(2)}(i,j) = \langle s_i s_j\rangle - \langle s_i\rangle\langle s_j\rangle = \langle s_i s_j\rangle - m^2. \tag{3.48}$$

(Here m denotes the expectation value of the magnetization.) In fact, since our Ising system is translationally invariant, $G_c^{(2)}(i,j)$ is dependent only on the displacement $\mathbf{r}$ between the sites i and j, and not on exactly where they are. In other words, if $\mathbf{r}_i$ is the position vector of the i^{th} spin, then we should have

$$G_c^{(2)}(\mathbf{r}_i, \mathbf{r}_i + \mathbf{r}) = G_c^{(2)}(\mathbf{r}) \tag{3.49}$$

independent of the value of $\mathbf{r}_i$. This means that we can improve our estimate of the correlation function $G_c^{(2)}(\mathbf{r})$ by averaging its value over the whole lattice for all pairs of spins separated by a displacement $\mathbf{r}$:

$$G_c^{(2)}(\mathbf{r}) = \frac{1}{N} \sum_{\substack{i,j \text{ with} \\ \mathbf{r}_j - \mathbf{r}_i = \mathbf{r}}} \left[\langle s_i s_j\rangle - m^2\right]. \tag{3.50}$$

If, as with our Ising model simulation, the system we are simulating has periodic boundary conditions (see Section 3.1), then $G_c^{(2)}(\mathbf{r})$ will not die away for very large values of $\mathbf{r}$. Instead, it will be periodic, dying away for values of $\mathbf{r}$ up to half the width of the lattice, and then building up again to another maximum when we have gone all the way across the lattice and got back to the spin we started with.

In order to evaluate $G_c^{(2)}(\mathbf{r})$ using Equation (3.50) we have to record the value of every single spin on the lattice at intervals during the simulation. This is not usually a big problem given the generous amounts of storage space provided by modern computers. However, if we want to calculate $G_c^{(2)}(\mathbf{r})$ for every value of $\mathbf{r}$ on the lattice, this kind of direct calculation does take an amount of time which scales with the number N of spins on the lattice as N^2. As with the calculation of the autocorrelation function in Section 3.3.1, it actually turns out to be quicker to calculate the Fourier transform of the correlation function instead.

The spatial Fourier transform $\tilde{G}_c^{(2)}(\mathbf{k})$ is defined by

$$\begin{aligned}\tilde{G}_c^{(2)}(\mathbf{k}) &= \sum_{\mathbf{r}} \mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\mathbf{r}}\, G_c^{(2)}(\mathbf{r}) \\ &= \frac{1}{N} \sum_{\mathbf{r}} \sum_{\substack{i,j \text{ with} \\ \mathbf{r}_j - \mathbf{r}_i = \mathbf{r}}} \mathrm{e}^{\mathrm{i}\mathbf{k}\cdot(\mathbf{r}_j - \mathbf{r}_i)} \left[\langle s_i s_j\rangle - m^2\right] \\ &= \frac{1}{N} \Bigl\langle \sum_{\mathbf{r}_i} \mathrm{e}^{-\mathrm{i}\mathbf{k}\cdot\mathbf{r}_i}(s_i - m) \sum_{\mathbf{r}_j} \mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\mathbf{r}_j}(s_j - m) \Bigr\rangle\end{aligned}$$

$$= \frac{1}{N}\langle|\tilde{s}'(\mathbf{k})|^2\rangle, \qquad (3.51)$$

where $\tilde{s}'(\mathbf{k})$ is the Fourier transform of $s_i' \equiv s_i - m$. In other words, in order to calculate $\widetilde{G}_c^{(2)}(\mathbf{k})$, we just need to perform a Fourier transform of the spins at a succession of different times throughout the simulation and then feed the results into Equation (3.51). As in the case of the autocorrelation function of Section 3.2, this can be done using a standard FFT algorithm. To get the correlation function in real space we then have to use the algorithm again to Fourier transform back, but the whole process still only takes a time which scales as $N \log N$, and so for the large lattices of today's Monte Carlo studies it is usually faster than direct calculation from Equation (3.50).[14]

Occasionally we also need to calculate the disconnected correlation function defined in Section 1.2.1. The equivalent of (3.51) in this case is simply

$$\widetilde{G}^{(2)}(\mathbf{k}) = \frac{1}{N}\langle|\tilde{s}(\mathbf{k})|^2\rangle. \qquad (3.52)$$

Note that s_i and s_i' differ only by the average magnetization m, which is a constant. As a result, $\widetilde{G}^{(2)}(\mathbf{k})$ and $\widetilde{G}_c^{(2)}(\mathbf{k})$ are in fact identical, except at $\mathbf{k} = 0$. For this reason, it is often simpler to calculate $\widetilde{G}_c^{(2)}(\mathbf{k})$ by first calculating $\widetilde{G}^{(2)}(\mathbf{k})$ and then just setting the $\mathbf{k} = 0$ component to zero.

3.7 An actual calculation

In this section we go through the details of an actual Monte Carlo simulation and demonstrate how the calculation proceeds. The example that we take is that of the simulation of the two-dimensional Ising model on a square lattice using the Metropolis algorithm. This system has the advantage that its properties in the thermodynamic limit are known exactly, following the analytic solution given by Onsager (1944). Comparing the results from our simulation with the exact solution will give us a feel for the sort of accuracy one can expect to achieve using the Monte Carlo method. Some of the results have already been presented (see Figures 3.3 and 3.5 for example). Here we

[14]Again we should point out that this does not necessarily mean that one should always calculate the correlation function this way. As with the calculation of the autocorrelation function, using the Fourier transform is a more complicated method than direct calculation of the correlation function, and if your goal is to get an estimate quickly, and your lattice is not very large, you may be better advised to go the direct route. However, the Fourier transform method is more often of use in the present case of the two-point correlation function, since in order to perform the thermal average appearing in Equations (3.50) and (3.51) we need to repeat the calculation about once every two correlation times throughout the entire simulation, which might mean doing it a hundred or a thousand times in one run. Under these circumstances the FFT method may well be advantageous.

describe in detail how these and other results are arrived at, and discuss what conclusions we can draw from them.

The first step in performing any Monte Carlo calculation, once we have decided on the algorithm we are going to use, is to write the computer program to implement that algorithm. The code used for our Metropolis simulation of the Ising model is given in Appendix B. It is written in the computer language C.

As a test of whether the program is correct, we have first used it to simulate a small 5×5 Ising model for a variety of temperatures between $T = 0$ and $T = 5.0$ with J set equal to 1. For such a small system our program runs very fast, and the entire simulation only took about a second at each temperature. In Section 1.3 we performed an exact calculation of the magnetization and specific heat for the 5×5 Ising system by directly evaluating the partition function from a sum over all the states of the system. This gives us something to compare our Monte Carlo results with, so that we can tell if our program is doing the right thing. At this stage, we are not interested in doing a very rigorous calculation, only in performing a quick check of the program, so we have not made much effort to ensure the equilibration of the system or to measure the correlation time. Instead, we simply ran our program for 20 000 Monte Carlo steps per site (i.e., $20\,000 \times 25 = 500\,000$ steps in all), and averaged over the last 18 000 of these to measure the magnetization and the energy. Then we calculated m from Equation (3.12) and c from Equation (3.15). If the results do not agree with our exact calculation then it could mean either that there is a problem with the program, or that we have not waited long enough in either the equilibration or the measurement sections of the simulation. However, as shown in Figure 3.7, the numerical results agree rather well with the exact ones. Even though we have not calculated the statistical errors on our data in order to determine the degree of agreement, these results still give us enough confidence in our program to proceed with a more thorough calculation on a larger system.

For our large-scale simulation, we have chosen to examine a system of 100×100 spins on a square lattice. We started the program with randomly chosen values of all the spins—the $T = \infty$ state of Section 3.1.1—and ran the simulations at a variety of temperatures from $T = 0.2$ to $T = 5.0$ in steps of 0.2, for a total of 25 simulations in all.[15] Again we ran our simulations

[15]Note that it is not possible to perform a simulation at $T = 0$ because the acceptance ratio, Equation (3.7), for spin flips which increase the energy of the system becomes zero in this limit. This means that it is not possible to guarantee that the system will come to equilibrium, because the requirement of ergodicity is violated; there are some states which it is not possible to get to in a finite number of moves. It is true in general of thermal Monte Carlo methods that they break down at $T = 0$, and often they become very slow close to $T = 0$. The continuous time Monte Carlo method of Section 2.4 can sometimes by used to overcome this problem in cases where we are particularly interested

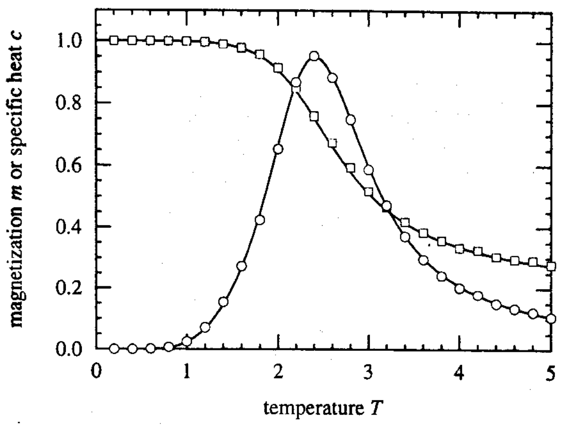

FIGURE 3.7 The magnetization (squares) and specific heat (circles) per spin of an Ising model in two dimensions on a 5×5 square lattice. The points are the results of the Metropolis Monte Carlo calculation described in the text. The lines are the exact calculations performed in Section 1.3, in which we evaluated the partition function by summing over all the states of the system.

for 20 000 Monte Carlo steps per lattice site. This is a fairly generous first run, and is only possible because we are looking at quite a small system still. In the case of larger or more complex models, one might well first perform a shorter run to get a rough measure of the equilibration and correlation times for the system, before deciding how long a simulation to perform. A still more sophisticated approach is to perform a short run and then store the configuration of the spins on the lattice before the program ends. Then, after deciding how long the entire calculation should last on the basis of the measurements during that short run, we can pick up exactly where we left off using the stored configuration, thus saving ourselves the effort of equilibrating the system twice.

Taking the data from our 25 simulations at different temperatures, we first estimate the equilibration times τ_{eq} at each temperature using the methods described in Section 3.2. In this case we found that all the equilibration times were less than about 1000 Monte Carlo steps per site, except for the simulations performed at $T = 2.0$ and $T = 2.2$, which both had equilibration times on the order of 6000 steps per site. (The reason for this anomaly is explained in the next section.) Allowing ourselves a margin for error in these estimates, we therefore took the data from time 2000 onwards as our equilibrium measurements for all the temperatures except the two slower ones, for which we took the data for times 10 000 onwards.

in the behaviour of a model close to $T = 0$.

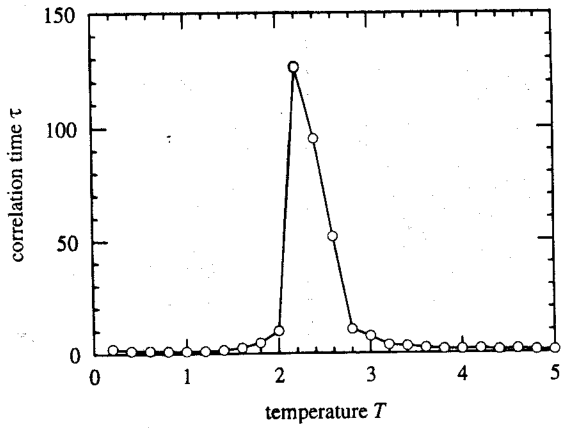

FIGURE 3.8 The correlation time for the 100×100 Ising model simulated using the Metropolis algorithm. The correlation time is measured in Monte Carlo steps per lattice site (i.e., in multiples of 10 000 Monte Carlo steps in this case). The straight lines joining the points are just to guide the eye.

Next we need to estimate how many independent measurements these data constitute, which means estimating the correlation time. To do this, we calculate the magnetization autocorrelation function at each temperature from Equation (3.21), for times t up to 1000. (We must be careful only to use our equilibrium data for this calculation since the autocorrelation function is an equilibrium quantity. That is, we should not use the data from the early part of the simulation during which the system was coming to equilibrium.) Performing a fit to these functions as in Figure 3.6, we make an estimate of the correlation time τ at each temperature. The results are shown in Figure 3.8. Note the peak in the correlation time around $T = 2.2$. This effect is called "critical slowing down", and we will discuss it in more detail in Section 3.7.2. Given the length of the simulation $t_{\max}$ and our estimates of τ_{eq} and τ for each temperature, we can calculate the number n of independent measurements to which our simulations correspond using Equation (3.19).

Using these figures we can now calculate the equilibrium properties of the 100×100 Ising model in two dimensions. As an example, we have calculated the magnetization and the specific heat again. Our estimate of the magnetization is calculated by averaging over the magnetization measurements from the simulation, again excluding the data from the early portion

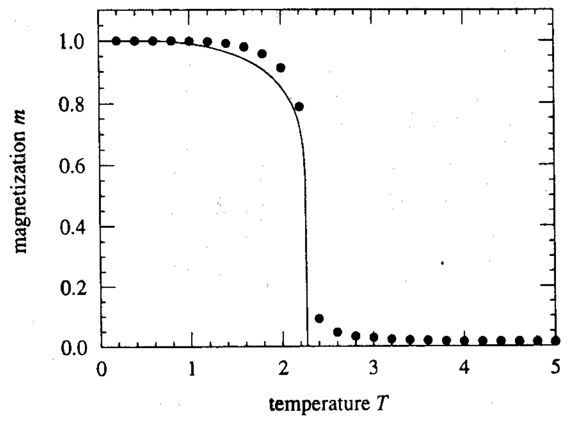

FIGURE 3.9 The magnetization per spin of the two-dimensional Ising model. The points are the results from our Monte Carlo simulation using the Metropolis algorithm. The errors are actually smaller than the points in this figure because the calculation is so accurate. The solid line is the known exact solution for the Ising model on an infinite two-dimensional square lattice.

of the run where the system was not equilibrated. The results are shown in Figure 3.9, along with the known exact solution for the infinite system. Calculating the errors on the magnetization from Equation (3.37), we find that the errors are so small that the error bars would be completely covered by the points themselves, so we have not bothered to put them in the figure. The agreement between the numerical calculation and the exact one for the infinite lattice is much better than it was for the smaller system in Figure 1.1, although there is still some discrepancy between the two. This discrepancy arises because the quantity plotted in the figure is in fact the average $\langle|m|\rangle$ of the *magnitude* of the magnetization, and not the average magnetization itself; we discuss our reasons for doing this in Section 3.7.1 when we examine the spontaneous magnetization of the Ising model.

The figure clearly shows the benefits of the Monte Carlo method. The calculation on the 5×5 system which we performed in Section 1.3 was exact, whereas the Monte Carlo calculation on the 100×100 system is not. However, the Monte Carlo calculation still gives a better estimate of the magnetization of the infinite system. The errors due to statistical fluctuations in our measurements of the magnetization are much smaller than the inaccuracy of working with a tiny 5×5 system.

Using the energy measurements from our simulation, we have also calculated the specific heat for our Ising model from Equation (3.15). To calculate the errors on the resulting numbers we could use the blocking method of Section 3.4.2 to get a rough estimate. Here we are interested in doing a more accurate calculation, and for that we should use either the bootstrap or the jackknife method (see Sections 3.4.3 and 3.4.4). The number of independent samples n is for most temperatures considerably greater than 100, so by the criterion given in Section 3.4.4, the bootstrap method is the more efficient one to use. In Figure 3.10 we show our results for the specific heat with error bars calculated from 200 bootstrap resamplings of the data (giving errors accurate to about 5%). The agreement here with the known exact result for the specific heat is excellent—better than for the magnetization—though the errors are larger, especially in the region close to the peak. If we were particularly interested to know the value of c in this region it would make sense for us to go back and do a longer run of our program in this region to get more accurate data. For example, if we wanted to calculate the entropy difference from one side of the peak to the other using Equation (3.45), then large error bars in this region would make the integral inaccurate and we might well benefit from expending a little more effort in this region to get more accurate results.

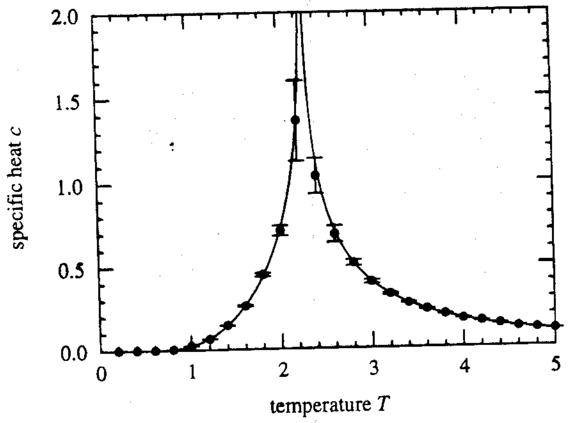

FIGURE 3.10 The specific heat per spin of the two-dimensional Ising model calculated by Monte Carlo simulation (points with error bars) and the exact solution for the same quantity (solid line). Note how the error bars get bigger close to the peak in the specific heat. This phenomenon is discussed in detail in the next section.

3.7.1 The phase transition

It is not really the intention of this book to discuss the properties of the Ising model in detail, or the properties of any other model. However, there are a few things about the Ising model which we need to look into in a little more detail in order to understand the strengths and weaknesses of the Metropolis algorithm, and to explain why other algorithms may be better for certain calculations.

If we look at Figures 3.9 and 3.10, the most obvious feature that strikes us is the behaviour of the model around temperature $T = 2.2$ or so. The results shown in Figure 3.9 indicate that above this temperature the mean magnetization per site m is quite small, whereas below it the magnetization is definitely non-zero and for the most part quite close to its maximum possible value of 1. This seems like a sensible way for the model to behave, since we know (see Section 3.1.1) that the $T = \infty$ state is one in which all the spins are randomly oriented up or down so that the net magnetization will be zero on average, and we know that the $T = 0$ state is one in which all the spins line up with one another, either all up or all down, so that the magnetization per site is either $+1$ or -1. If we only had results for a small Ising system to go by, like the ones depicted in Figure 3.7, we might imagine that the true behaviour of the system was simply that the magnetization rose smoothly from zero in the $T \to \infty$ limit to 1 as the temperature tends to zero. However, our results for the larger 100×100 system indicate that the transition from small m to large m becomes sharper as we go to larger systems, and in fact we know in the case of this particular model, because we have an exact solution, that the change from one regime to the other is actually infinitely sharp in the thermodynamic limit. This kind of change is called a **phase transition**. The Ising model is known to have a phase transition in two or more dimensions. The two regimes we observe are called the **phases** of the model. The phase transition between them takes place at a temperature T_c, which we call the **critical temperature**, whose value in the particular case of the two-dimensional Ising model is known to be

$$T_c = \frac{2J}{\log(1+\sqrt{2})} \simeq 2.269J. \tag{3.53}$$

Above this temperature the system is in the **paramagnetic phase**, in which the average magnetization is zero. Below it, the system is in the **ferromagnetic phase** and develops a **spontaneous magnetization** (i.e., most of the spins align one way or the other and the magnetization becomes non-zero all of its own accord without the application of a magnetic field to the model). This spontaneous magnetization rises from zero at the phase transition to unity at absolute zero. The magnetization is referred to as the **order parameter** of the Ising model because of this behaviour. In general, an order parameter is any quantity which is zero on one side of a phase transition and

non-zero on the other. A phase transition in which the order parameter is continuous at T_c, as it is here, is called a **continuous phase transition**.

In fact, to be strictly correct, the *mean* magnetization of the Ising model below the critical temperature is still zero, since the system is equally happy to have most of its spins pointing either down or up. Thus if we average over a long period of time we will find that the magnetization is close to $+1$ half the time and close to -1 for the other half, with occasional transitions between the two, so that the average is still close to zero. However, the average of the magnitude $|m|$ of the magnetization will be close to $+1$, whereas it will be close to zero above the phase transition. In Figure 3.9 we therefore actually plotted the average of $|m|$, and not m. This explains why the magnetization above the transition temperature is still slightly greater than zero. The average magnetization in this phase is definitely zero (give or take the statistical error) but the average of the *magnitude* of m is always greater than zero, since we are taking the average of a number which is never negative. Still, as we go to the thermodynamic limit we expect this quantity to tend to zero, so that the numerical result and the exact solution should agree.[16]

We can look in detail at what happens to the spins in our Ising system as we pass through the phase transition from high to low temperatures by examining pictures such as those in Figure 3.2. At high temperatures the spins are random and uncorrelated, but as the temperature is lowered the interactions between them encourage nearby spins to point in the same direction, giving rise to correlations in the system. Groups of adjacent spins which are correlated in this fashion and tend to point in the same direction are called **clusters**.[17] As we approach T_c, the typical size ξ of these clusters—also called the **correlation length**—diverges, so that when we are precisely at the transition, we may encounter arbitrarily large areas in which the spins are pointing mostly up or mostly down. Then, as we pass below the transition temperature, the system spontaneously chooses to have the majority of its spins in either the up or the down direction, and develops a non-zero magnetization in that direction. Which direction it chooses depends solely

[16]This also provides an explanation of why the agreement between the analytic solution and the Monte Carlo calculation was better for the specific heat, Figure 3.10, than it was for the magnetization. The process of taking the mean of the magnitude $|m|$ of the magnetization means that we consistently overestimate the magnetization above the critical temperature, and in fact this problem extends to temperatures a little below T_c as well (see Figure 3.9). No such adjustments are necessary when calculating the specific heat, and as a result our simulation agrees much better with the known values for c, even though the error bars are larger in this case.

[17]A number of different mathematical definitions of a cluster are possible. Some of them require that all spins in the cluster point in the same direction whilst others are less strict. We discuss these definitions in detail in the next chapter, particularly in Sections 4.2 and 4.4.2.

on the random details of the thermal fluctuations it was going through as we passed the critical temperature, and so is itself completely random. As the temperature drops further towards $T = 0$, more and more of the spins line up in the same direction, and eventually as $T \to 0$ we get $|m| = 1$.

The study of phase transitions is an entire subject in itself and we refer the interested reader to other sources for more details of this interesting field. For our purposes the brief summary given above will be enough.

3.7.2 Critical fluctuations and critical slowing down

We are interested in the behaviour of the Ising model in the region close to T_c. This region is called the **critical region**, and the processes typical of the critical region are called **critical phenomena**. As we mentioned, the system tends to form into large clusters of predominantly up- or down-pointing spins as we approach the critical temperature from above. These clusters contribute significantly to both the magnetization and the energy of the system, so that, as they flip from one orientation to another, they produce large fluctuations in m and E, often called **critical fluctuations**. As the typical size ξ of the clusters diverges as $T \to T_c$, the size of the fluctuations does too. And since fluctuations in m and E are related to the magnetic susceptibility and the specific heat through Equations (3.15) and (3.16), we expect to get divergences in these quantities at T_c also. This is what we see in Figure 3.10. These divergences are some of the most interesting of critical phenomena, and a lot of effort, particularly using Monte Carlo methods, has been devoted to investigating their exact nature. Many Monte Carlo studies of many different models have focused exclusively on the critical region to the exclusion of all else. Unfortunately, it is in precisely this region that our Metropolis algorithm is least accurate.

There are two reasons for this. The first has to do with the critical fluctuations. The statistical errors in the measured values of quantities like the magnetization and the internal energy are proportional to the size of these critical fluctuations (see Section 3.4) and so grow as we approach T_c. In a finite-sized system like the ones we study in our simulations, the size of the fluctuations never actually diverges—that can only happen in the thermodynamic limit—but they can become very large, and this makes for large statistical errors in the measured quantities.

What can we do about this? Well, recall that the error on, for example, the magnetization m indeed increases with the size of the magnetization fluctuations, but it also decreases with the number of independent measurements of m that we make during our simulation (see Equation (3.37)). Thus, in order to reduce the error bars on measurements close to T_c, we need to run our program for longer, so that we get a larger number of measurements. This however, is where the other problem with the Metropolis algorithm

comes in. As we saw in Figure 3.8, the correlation time τ of the simulation is also large in the region around T_c. In fact, like the susceptibility and the specific heat, the correlation time actually diverges at T_c in the thermodynamic limit. For the finite-sized systems of our Monte Carlo simulations τ does not diverge, but it can still become very large in the critical region, and a large correlation time means that the number of independent measurements we can extract from a simulation of a certain length is small (see Equation (3.19)). This effect on its own would increase the size of the errors on measurements from our simulation, even without the large critical fluctuations. The combination of both effects is particularly unfortunate, because it means that in order to increase the number of independent measurements we make during our simulation, we have to perform a much longer run of the program; the computer time necessary to reduce the error bars to a size comparable with those away from T_c increases very rapidly as we approach the phase transition.

The critical fluctuations which increase the size of our error bars are an innate physical feature of the Ising model. Any Monte Carlo algorithm which correctly samples the Boltzmann distribution will also give critical fluctuations. There is nothing we can do to change our algorithm which will reduce this source of error. However, the same is not true of the increase in correlation time. This effect, known as **critical slowing down**, is a property of the Monte Carlo algorithm we have used to perform the simulation, but not of the Ising model in general. Different algorithms can have different values of the correlation time at any given temperature, and the degree to which the correlation time grows as we approach T_c, if it grows at all, depends on the precise details of the algorithm. Therefore, if we are particularly interested in the behaviour of a model in the critical region, it may be possible to construct an algorithm which suffers less from critical slowing down than does the Metropolis algorithm, or even eliminates it completely, allowing us to achieve much greater accuracy for our measurements. In the next chapter we look at a number of other algorithms which do just this and which allow us to study the critical region of the Ising model more accurately.

Problems

3.1 Derive the appropriate generalization of Equation (3.10) for a simulation of an Ising model with non-zero external magnetic field B.

3.2 Suppose we have a set of n measurements $x_1 \ldots x_n$ of a real quantity x. Find an approximate expression for the error on our best estimate of the mean of their squares. Take the numbers below and estimate this error.

20.27	19.61	20.06	20.73
20.09	20.68	19.37	20.40
19.95	20.55	19.64	19.94

Now estimate the same quantity for the same set of numbers using the jackknife method of Section 3.4.4.

3.3 In this chapter we described methods for calculating a variety of quantities from Monte Carlo data, including internal energies, specific heats and entropies. Suggest a way in which we might measure the partition function using data from a Monte Carlo simulation.

3.4 Write a computer program to carry out a Metropolis Monte Carlo simulation of the one-dimensional Ising model in zero field. Use it to calculate the internal energy of the model for a variety of temperatures and check the results against the analytic solution from Problem 1.4.

4 Other algorithms for the Ising model

In the last chapter we saw that the Metropolis algorithm with single-spin-flip dynamics is an excellent Monte Carlo algorithm for simulating the Ising model when we are interested in temperatures well away from the critical temperature T_c. However, as we approach the critical temperature, the combination of large critical fluctuations and long correlation time makes the errors on measured quantities grow enormously. As we pointed out, there is little to be done about the critical fluctuations, since these are an intrinsic property of the model near its phase transition (and are, what's more, precisely the kind of interesting physical effect that we *want* to study with our simulations). On the other hand, the increase in the correlation time close to the phase transition is a function of the particular algorithm we are using—the Metropolis algorithm in this case—and it turns out that by using different algorithms we can greatly reduce this undesirable effect. In the first part of this chapter we will study one of the most widely used and successful such algorithms, the Wolff algorithm. Before introducing the algorithm however, we need to define a few terms.

4.1 Critical exponents and their measurement

As discussed in Section 3.7.1, the spins of an Ising model in equilibrium group themselves into clusters of a typical size ξ, called the correlation length, and this correlation length diverges as the temperature approaches T_c. Let us define a dimensionless parameter t, called the **reduced temperature**, which measures how far away we are from T_c:

$$t = \frac{T - T_c}{T_c}. \tag{4.1}$$