are allowed to move back and forward between the boxes under the influence of thermal excitations from a reservoir at temperature T. Find the partition function for this system and then use this result to calculate the internal energy.

1.4 Solve the Ising model, whose Hamiltonian is given in Equation (1.30), in one dimension for the case where $B = 0$ as follows. Define a new set of variables σ_i which take values 0 and 1 according to $\sigma_i = \frac{1}{2}(1 - s_i s_{i+1})$ and rewrite the Hamiltonian in terms of these variables for a system of N spins with periodic boundary conditions. Show that the resulting system is equivalent to the one studied in Problem 1.3 in the limit of large N and hence calculate the internal energy as a function of temperature.

2 The principles of equilibrium thermal Monte Carlo simulation

In Section 1.3.1 we looked briefly at the general ideas behind equilibrium thermal Monte Carlo simulations. In this chapter we discuss these ideas in more detail in preparation for the discussion in the following chapters of a variety of specific algorithms for use with specific problems. The three crucial ideas that we introduce in this chapter are "importance sampling", "detailed balance" and "acceptance ratios". If you know what these phrases mean, you can understand most of the thermal Monte Carlo simulations that have been performed in the last thirty years.

2.1 The estimator

The usual goal in the Monte Carlo simulation of a thermal system is the calculation of the expectation value $\langle Q\rangle$ of some observable quantity Q, such as the internal energy in a model of a gas, or the magnetization in a magnetic model. As we showed in Section 1.3, the ideal route to calculating such an expectation, that of averaging the quantity of interest over all states μ of the system, weighting each with its own Boltzmann probability

$$\langle Q\rangle = \frac{\sum_\mu Q_\mu \,\mathrm{e}^{-\beta E_\mu}}{\sum_\mu \mathrm{e}^{-\beta E_\mu}} \tag{2.1}$$

is only tractable in the very smallest of systems. In larger systems, the best we can do is average over some subset of the states, though this necessarily introduces some inaccuracy into the calculation. Monte Carlo techniques work by choosing a subset of states at random from some probability distribution p_μ which we specify. Suppose we choose M such states $\{\mu_1 \dots \mu_M\}$.

Our best estimate of the quantity Q will then be given by

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} p_{\mu_i}^{-1} \mathrm{e}^{-\beta E_{\mu_i}}}{\sum_{j=1}^{M} p_{\mu_j}^{-1} \mathrm{e}^{-\beta E_{\mu_j}}}. \qquad (2.2)$$

Q_M is called the **estimator** of Q. It has the property that, as the number M of states sampled increases, it becomes a more and more accurate estimate of $\langle Q\rangle$, and when $M \to \infty$ we have $Q_M = \langle Q\rangle$.

The question we would like to answer now is how should we choose our M states in order that Q_M be an accurate estimate of $\langle Q\rangle$? In other words, how should we choose the probability distribution p_μ? The simplest choice is to pick all states with equal probability; in other words make all p_μ equal. Substituting this choice into Equation (2.2), we get

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} \mathrm{e}^{-\beta E_{\mu_i}}}{\sum_{j=1}^{M} \mathrm{e}^{-\beta E_{\mu_j}}}. \qquad (2.3)$$

It turns out however, that this is usually a rather poor choice to make. In most numerical calculations it is only possible to sample a very small fraction of the total number of states. Consider, for example, the Ising model of Section 1.2.2 again. A small three-dimensional cubic system of $10 \times 10 \times 10$ Ising spins would have $2^{1000} \simeq 10^{300}$ states, and a typical numerical calculation could only hope to sample up to about 10^8 of those in a few hours on a good computer, which would mean we were only sampling one in every 10^{292} states of the system, a very small fraction indeed. The estimator given above is normally a poor guide to the value of $\langle Q\rangle$ under these circumstances. The reason is that one or both of the sums appearing in Equation (2.1) may be dominated by a small number of states, with all the other states, the vast majority, contributing a negligible amount even when we add them all together. This effect is often especially obvious at low temperatures, where these sums may be dominated by a hundred states, or ten states, or even one state, because at low temperatures there is not enough thermal energy to lift the system into the higher excited states, and so it spends almost all of its time sitting in the ground state, or one of the lowest of the excited states. In the example described above, the chances of one of the 10^8 random states we sample in our simulation being the ground state are one in 10^{292}, which means there is essentially no chance of our picking it, which makes Q_M a very inaccurate estimate of $\langle Q\rangle$ if the sums are dominated by the contribution from this state.

On the other hand, if we had some way of knowing which states made the important contributions to the sums in Equation (2.1) and if we could pick our sample of M states from just those states and ignore all the others, we could get a very good estimate of $\langle Q\rangle$ with only a small number of terms. This is the essence of the idea behind thermal Monte Carlo methods. The technique for picking out the important states from amongst the very large number of possibilities is called **importance sampling**.

2.2 Importance sampling

As discussed in Section 1.1, we can regard an expectation value as a time average over the states that a system passes through during the course of a measurement. We do not assume that the system passes through every state during the measurement, even though every state appears in the sums of Equation (2.1). When you count how many states a typical system has you realize that this would never be possible. For instance, consider again the example we took in the last chapter of a litre container of gas at room temperature and atmospheric pressure. Such a system contains on the order of 10^{22} molecules. Typical speeds for these molecules are in the region of $100\ \mathrm{m\,s^{-1}}$, giving them a de Broglie wavelength of around 10^{-10} m. Each molecule will then have about 10^{27} different quantum states within the one litre box, and the complete gas will have around $(10^{27})^{10^{22}}$ states, which is a spectacularly large number.[1] The molecules will change from one state to another when they undergo collisions with one another or with the walls of the container, which they do at a rate of about 10^9 collisions per second, or 10^{31} changes of state per second for the whole gas. At this rate, it will take about $10^{10^{23}}$ times the lifetime of the universe for our litre of gas to move through every possible state. Clearly then, our laboratory systems are only sampling the tiniest portion of their state spaces during the time that we conduct our experiments on them. In effect, real systems are carrying out a sort of Monte Carlo calculation of their own properties; they are "analogue computers" which evaluate expectations by taking a small but representative sample of their own states and averaging over that sample.[2] So it should not come as a great surprise to learn that we can also perform a reasonable calculation of the properties of a system using a simulation which only samples a small fraction of its states.

In fact, our calculations are often significantly better than this simple argument suggests. In Section 1.2.1 we showed that the range of energies of the states sampled by a typical system is very small compared with the total

[1]Actually, this is probably an overestimate, since it counts states which are classically distinguishable but quantum mechanically identical. For the purpose of the present rough estimation however, it will do fine.

[2]There are some systems which, because they have certain conservation laws, will not in fact sample their state spaces representatively, and this can lead to discrepancies between theory and experiment. Special Monte Carlo techniques exist for simulating these "conservative" systems, and we will touch on one or two of them in the coming chapters. For the moment, however, we will make the assumption that our system takes a representative sample of its own states.

energy of the system—the ratio was about 10^{-20} in the case of our litre of gas, for instance. Similar arguments can be used to show that systems sample very narrow ranges of other quantities as well. The reason for this, as we saw, is that the system is not sampling all states with equal probability, but instead sampling them according to the Boltzmann probability distribution, Equation (1.5). If we can mimic this effect in our simulations, we can exploit these narrow ranges of energy and other quantities to make our estimates of such quantities very accurate. For this reason, we normally try to take a sample of the states of the system in which the likelihood of any particular one appearing is proportional to its Boltzmann weight. This is the most common form of importance sampling, and most of the algorithms in this book make use of this idea in one form or another.

Our strategy then is this: instead of picking our M states in such a way that every state of the system is as likely to get chosen as every other, we pick them so that the probability that a particular state μ gets chosen is $p_\mu = Z^{-1}e^{-\beta E_\mu}$. Then our estimator for $\langle Q \rangle$, Equation (2.2), becomes just

$$Q_M = \frac{1}{M}\sum_{i=1}^{M} Q_{\mu_i}. \qquad (2.4)$$

Notice that the Boltzmann factors have now cancelled out of the estimator, top and bottom, leaving a particularly simple expression. This definition of Q_M works much better than (2.3), especially when the system is spending the majority of its time in a small number of states (such as, for example, the lowest-lying ones when we are at low temperatures), since these will be precisely the states that we pick most often, and the relative frequency with which we pick them will exactly correspond to the amount of time the real system would spend in those states.

The only remaining question is *how* exactly we pick our states so that each one appears with its correct Boltzmann probability. This is by no means a simple task. In the remainder of this chapter we describe the standard solution to the problem, which makes use of a "Markov process".

2.2.1 Markov processes

The tricky part of performing a Monte Carlo simulation is the generation of an appropriate random set of states according to the Boltzmann probability distribution. For a start, one cannot simply choose states at random and accept or reject them with a probability proportional to $e^{-\beta E_\mu}$. That would be no better than our original scheme of sampling states at random; we would end up rejecting virtually all states, since the probabilities for their acceptance would be exponentially small. Instead, almost all Monte Carlo schemes rely on **Markov processes** as the generating engine for the set of states used.

For our purposes, a Markov process is a mechanism which, given a system in one state μ, generates a new state of that system ν. It does so in a random fashion; it will not generate the same new state every time it is given the initial state μ. The probability of generating the state ν given μ is called the **transition probability** $P(\mu \to \nu)$ for the transition from μ to ν, and for a true Markov process all the transition probabilities should satisfy two conditions: (1) they should not vary over time, and (2) they should depend only on the properties of the current states μ and ν, and not on any other states the system has passed through. These conditions mean that the probability of the Markov process generating the state ν on being fed the state μ is the same every time it is fed the state μ, irrespective of anything else that has happened. The transition probabilities $P(\mu \to \nu)$ must also satisfy the constraint

$$\sum_\nu P(\mu \to \nu) = 1, \qquad (2.5)$$

since the Markov process must generate *some* state ν when handed a system in the state μ. Note however, that the transition probability $P(\mu \to \mu)$, which is the probability that the new state generated will be the same as the old one, need not be zero. This amounts to saying there may be a finite probability that the Markov process will just stay in state μ.

In a Monte Carlo simulation we use a Markov process repeatedly to generate a **Markov chain** of states. Starting with a state μ, we use the process to generate a new one ν, and then we feed *that* state into the process to generate another λ, and so on. The Markov process is chosen specially so that when it is run for long enough starting from any state of the system it will eventually produce a succession of states which appear with probabilities given by the Boltzmann distribution. (We call the process of reaching the Boltzmann distribution "coming to equilibrium", since it is exactly the process that a real system goes through with its "analogue computer" as it reaches equilibrium at the ambient temperature.) In order to achieve this, we place two further conditions on our Markov process, in addition to the ones specified above, the conditions of "ergodicity" and "detailed balance"

2.2.2 Ergodicity

The **condition of ergodicity** is the requirement that it should be possible for our Markov process to reach any state of the system from any other state, if we run it for long enough. This is necessary to achieve our stated goal of generating states with their correct Boltzmann probabilities. Every state ν appears with some non-zero probability p_ν in the Boltzmann distribution, and if that state were inaccessible from another state μ no matter how long we continue our process for, then our goal is thwarted if we start in state μ: the probability of finding ν in our Markov chain of states will be zero, and not p_ν as we require it to be.

The condition of ergodicity tells us that we are allowed to make some of the transition probabilities of our Markov process zero, but that there must be at least one path of non-zero transition probabilities between any two states that we pick. In practice, most Monte Carlo algorithms set almost all of the transition probabilities to zero, and we must be careful that in so doing we do not create an algorithm which violates ergodicity. For most of the algorithms we describe in this book we will explicitly prove that ergodicity is satisfied before making use of the algorithm.

2.2.3 Detailed balance

The other condition we place on our Markov process is the **condition of detailed balance**. This condition is the one which ensures that it is the Boltzmann probability distribution which we generate after our system has come to equilibrium, rather than any other distribution. Its derivation is quite subtle. Consider first what it means to say that the system is in equilibrium. The crucial defining condition is that the rate at which the system makes transitions into and out of any state μ must be equal. Mathematically we can express this as[3]

$$\sum_\nu p_\mu P(\mu \to \nu) = \sum_\nu p_\nu P(\nu \to \mu). \tag{2.6}$$

Making use of the sum rule, Equation (2.5), we can simplify this to

$$p_\mu = \sum_\nu p_\nu P(\nu \to \mu). \tag{2.7}$$

For any set of transition probabilities satisfying this equation, the probability distribution p_μ will be an equilibrium of the dynamics of the Markov process. Unfortunately, however, simply satisfying this equation is not sufficient to guarantee that the probability distribution will tend to p_μ from any state of the system if we run the process for long enough. We can demonstrate this as follows.

The transition probabilities $P(\mu \to \nu)$ can be thought of as the elements of a matrix $\mathbf{P}$. This matrix is called the **Markov matrix** or the **stochastic matrix** for the Markov process. Let us return to the notation of Section 1.1, in which we denoted by $w_\mu(t)$, the probability that our system is in a state μ at time t. If we measure time in steps along our Markov chain, then the

[3]This equation is essentially just a discrete-time version of the one we would get if we were to set the derivative in the master equation, Equation (1.1), to zero.

probability $w_\nu(t+1)$ of being in state ν at time $t+1$ is given by[4]

$$w_\nu(t+1) = \sum_\mu P(\mu \to \nu)\, w_\mu(t). \tag{2.8}$$

In matrix notation, this becomes

$$\mathbf{w}(t+1) = \mathbf{P} \cdot \mathbf{w}(t), \tag{2.9}$$

where $\mathbf{w}(t)$ is the vector whose elements are the weights $w_\mu(t)$. If the Markov process reaches a simple equilibrium state $\mathbf{w}(\infty)$ as $t \to \infty$, then that state satisfies

$$\mathbf{w}(\infty) = \mathbf{P} \cdot \mathbf{w}(\infty). \tag{2.10}$$

However, it is also possible for the process to reach a **dynamic equilibrium** in which the probability distribution $\mathbf{w}$ rotates around a number of different values. Such a rotation is called a **limit cycle**. In this case $\mathbf{w}(\infty)$ would satisfy

$$\mathbf{w}(\infty) = \mathbf{P}^n \cdot \mathbf{w}(\infty), \tag{2.11}$$

where n is the length of the limit cycle. If we choose our transition probabilities (or equivalently our Markov matrix) to satisfy Equation (2.7) we guarantee that the Markov chain will have a simple equilibrium probability distribution p_μ, but it may also have any number of limit cycles of the form (2.11). This means that there is no guarantee that the actual states generated will have anything like the desired probability distribution.

We get around this problem by applying an additional condition to our transition probabilities thus:

$$p_\mu P(\mu \to \nu) = p_\nu P(\nu \to \mu). \tag{2.12}$$

This is the condition of detailed balance. It is clear that any set of transition probabilities which satisfy this condition also satisfy Equation (2.6). (To prove it, simply sum both sides of Equation (2.12) over ν.) We can also show that this condition eliminates limit cycles. To see this, look first at the left-hand side of the equation, which is the probability of being in a state μ multiplied by the probability of making a transition from that state to another state ν. In other words, it is the overall rate at which transitions from μ to ν happen in our system. The right-hand side is the overall rate for the reverse transition. The condition of detailed balance tells us that on average the system should go from μ to ν just as often as it goes from ν to μ. In a limit cycle, in which the probability of occupation of some or all of the states changes in a cyclic fashion, there must be states for which this

[4]This equation is also closely related to Equation (1.1). The reader may like to work out how the one can be transformed into the other.

condition is violated on any particular step of the Markov chain; in order for the probability of occupation of a particular state to increase, for instance, there must be more transitions into that state than out of it, on average. The condition of detailed balance forbids dynamics of this kind and hence forbids limit cycles.

Once we remove the limit cycles in this way, it is straightforward to show that the system will always tend to the probability distribution p_μ as $t \to \infty$. As $t \to \infty$, $\mathbf{w}(t)$ will tend exponentially towards the eigenvector corresponding to the largest eigenvalue of $\mathbf{P}$. This may be obvious to you if you are familiar with stochastic matrices. If not, we prove it in Section 3.3.2. For the moment, let us take it as given. Looking at Equation (2.10) we see that the largest eigenvalue of the Markov matrix must in fact be one.[5] If limit cycles of the form (2.11) were present, then we could also have eigenvalues which are complex roots of one, but the condition of detailed balance prevents this from happening. Now look back at Equation (2.7) again. We can express this equation in matrix notation as

$$\mathbf{p} = \mathbf{P} \cdot \mathbf{p}. \tag{2.13}$$

In other words, if Equation (2.7) (or equivalently the condition of detailed balance) holds for our Markov process, then the vector $\mathbf{p}$ whose elements are the probabilities p_μ is precisely the one correctly normalized eigenvector of the Markov matrix which has eigenvalue one. Putting this together with Equation (2.10) we see that the equilibrium probability distribution over states $\mathbf{w}(\infty)$ is none other than $\mathbf{p}$, and hence $\mathbf{w}(t)$ must tend exponentially to $\mathbf{p}$ as $t \to \infty$.

There is another reason why detailed balance makes sense for Monte Carlo simulations: the "analogue computers" which constitute the real physical systems we are trying to mimic almost always obey the condition of detailed balance. The reason is that they are based on standard quantum or classical mechanics, which is time-reversal symmetric. If they did not obey detailed balance, then in equilibrium they could have one or more limit cycles around which the system passes in one particular direction. If we take such a system and reverse it in time, the motion around this cycle is also reversed, and it becomes clear that the dynamics of the system in equilibrium is not the same forward as it is in reverse. Such a violation of time-reversal symmetry is forbidden for most systems, implying that they must satisfy detailed balance. Although this does not mean that we are necessarily obliged

[5]All Markov matrices have at least one eigenvector with corresponding eigenvalue one, a fact which is easily proven since Equation (2.5) implies that the vector $(1, 1, 1, \ldots)$ is a left eigenvector of $\mathbf{P}$ with eigenvalue one. It is possible to have more than one eigenvector with eigenvalue one if the states of the system divide into two or more mutually inaccessible subsets. However, if the condition of ergodicity is satisfied then such subsets are forbidden and hence there is only one such eigenvector.

to enforce detailed balance in our simulations as well, it is helpful if we do, because it makes the behaviour of our model system more similar to that of the real one we are trying to understand.

So, we have shown that we can arrange for the probability distribution of states generated by our Markov process to tend to any distribution p_μ we please by choosing a set of transition probabilities which satisfy Equation (2.12). Given that we wish the equilibrium distribution to be the Boltzmann distribution,[6] clearly we want to choose the values of p_μ to be the Boltzmann probabilities, Equation (1.5). The detailed balance equation then tells us that the transition probabilities should satisfy

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{p_\nu}{p_\mu} = \mathrm{e}^{-\beta(E_\nu - E_\mu)}. \tag{2.14}$$

This equation and Equation (2.5) are the constraints on our choice of transition probabilities $P(\mu \to \nu)$. If we satisfy these, as well as the condition of ergodicity, then the equilibrium distribution of states in our Markov process will be the Boltzmann distribution. Given a suitable set of transition probabilities, our plan is then to write a computer program which implements the Markov process corresponding to these transition probabilities so as to generate a chain of states. After waiting a suitable length of time[7] to allow the probability distribution of states $w_\mu(t)$ to get sufficiently close to the Boltzmann distribution, we average the observable Q that we are interested in over M states and we have calculated the estimator Q_M defined in Equation (2.4). A number of refinements on this outline are possible and we will discuss some of those in the remainder of this chapter and in later chapters of the book, but this is the basic principle on which virtually all modern equilibrium Monte Carlo calculations are based.

Our constraints still leave us a good deal of freedom over how we choose the transition probabilities. There are many ways in which to satisfy them. One simple choice for example is

$$P(\mu \to \nu) \propto \mathrm{e}^{-\frac{1}{2}\beta(E_\mu - E_\nu)}, \tag{2.15}$$

although as we will show in Section 3.1 this choice is not a very good one. There are some other choices which are known to work well in many cases, such as the "Metropolis algorithm" proposed by Metropolis and co-workers in 1953, and we will discuss the most important of these in the coming chapters. However, it must be stressed—and this is one of the most important

[6]Occasionally, in fact, we want to generate equilibrium distributions other than the Boltzmann distribution. An example is the entropic sampling algorithm of Section 6.3. In this case the arguments here still apply. We simply feed our required distribution into the condition of detailed balance.

[7]Exactly how long we have to wait can be a difficult thing to decide. A number of possible criteria are discussed in Section 3.2.

things this book has to say—that the standard algorithms are *very rarely* the best ones for solving new problems with. In most cases they will work, and in some cases they will even give quite good answers, but you can almost always do a better job by giving a little extra thought to choosing the best set of transition probabilities to construct an algorithm that will answer the particular questions that you are interested in. A purpose-built algorithm can often give a much faster simulation than an equivalent standard algorithm, and the improvement in efficiency can easily make the difference between finding an answer to a problem and not finding one.

2.3 Acceptance ratios

Our little summary above makes rather light work of the problems of constructing a Monte Carlo algorithm. Given a desired set of transition probabilities $P(\mu \to \nu)$ satisfying the conditions (2.5) and (2.14), we say, we simply concoct some Markov process that generates states with exactly those transition probabilities, and *presto!* we produce a string of states of our system with exactly their correct Boltzmann probabilities. However, it is often very far from obvious what the appropriate Markov process is that has the required transition probabilities, and finding one can be a haphazard, trial-and-error process. For some problems we can use known algorithms such as the Metropolis method (see Section 3.1), but for many problems the standard methods are far from ideal, and we will do much better if we can tailor a new algorithm to our specific needs. But though we may be able to suggest many candidate Markov processes—different ways of creating a new state ν from an old one μ—still we may not find one which gives exactly the right set of transition probabilities. The good news however is that we don't have to. In fact it turns out that we can choose any algorithm we like for generating the new states, and still have our desired set of transition probabilities, by introducing something called an **acceptance ratio**. The idea behind the trick is this.

We mentioned in Section 2.2.1 that we are allowed to make the "stay-at-home" transition probability $P(\mu \to \mu)$ non-zero if we want. If we set $\nu = \mu$ in Equation (2.14), we get the simple tautology $1 = 1$, which means that the condition of detailed balance is always satisfied for $P(\mu \to \mu)$, no matter what value we choose for it. This gives us some flexibility about how we choose the other transition probabilities with $\mu \neq \nu$. For a start, it means that we can adjust the value of any $P(\mu \to \nu)$ and keep the sum rule (2.5) satisfied, by simply compensating for that adjustment with an equal but opposite adjustment of $P(\mu \to \mu)$. The only thing we need to watch is that $P(\mu \to \mu)$ never passes out of its allowed range between zero and one. If we make an adjustment like this in $P(\mu \to \nu)$, we can also arrange for Equation (2.14) to remain satisfied, by simultaneously making a change in

$P(\nu \to \mu)$, so that the ratio of the two is preserved.

It turns out that these considerations actually give us enough freedom that we can make the transition probabilities take any set of values we like by tweaking the values of the probabilities $P(\mu \to \mu)$. To see this, we break the transition probability down into two parts:

$$P(\mu \to \nu) = g(\mu \to \nu)\, A(\mu \to \nu). \tag{2.16}$$

The quantity $g(\mu \to \nu)$ is the **selection probability**, which is the probability, given an initial state μ, that our algorithm will generate a new target state ν, and $A(\mu \to \nu)$ is the acceptance ratio (sometimes also called the "acceptance probability"). The acceptance ratio says that if we start off in a state μ and our algorithm generates a new state ν from it, we should accept that state and change our system to the new state ν a fraction of the time $A(\mu \to \nu)$. The rest of the time we should just stay in the state μ. We are free to choose the acceptance ratio to be any number we like between zero and one; choosing it to be zero for all transitions is equivalent to choosing $P(\mu \to \mu) = 1$, which is the largest value it can take, and means that we will never leave the state μ. (Not a very desirable situation. We would never choose an acceptance ratio of zero for an actual calculation.)

This gives us complete freedom about how we choose the selection probabilities $g(\mu \to \nu)$, since the constraint (2.14) only fixes the ratio

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{g(\mu \to \nu)A(\mu \to \nu)}{g(\nu \to \mu)A(\nu \to \mu)}. \tag{2.17}$$

The ratio $A(\mu \to \nu)/A(\nu \to \mu)$ can take any value we choose between zero and infinity, which means that both $g(\mu \to \nu)$ and $g(\nu \to \mu)$ can take any values we like.

Our other constraint, the sum rule of Equation (2.5), is still satisfied, since the system must end up in *some* state after each step in the Markov chain, even if that state is just the state we started in.

So, in order to create our Monte Carlo algorithm what we actually do is think up an algorithm which generates random new states ν given old ones μ, with some set of probabilities $g(\mu \to \nu)$, and then we accept or reject those states with acceptance ratios $A(\mu \to \nu)$ which we choose to satisfy Equation (2.17). This will then satisfy all the requirements for the transition probabilities, and so produce a string of states which, when the algorithm reaches equilibrium, will each appear with their correct Boltzmann probability.

This all seems delightful, but there is a catch which we must always bear in mind, and which is one of the most important considerations in the design of Monte Carlo algorithms. If the acceptance ratios for our moves are low, then the algorithm will on most time steps simply stay in the state that it is in, and not go anywhere. The step on which it actually accepts a change to a new state will be rare, and this is wasteful of time. We want an algorithm that moves nimbly about state space and samples a wide selection of different states. We don't want to take a million time steps and find that our algorithm has only sampled a dozen states. The solution to this problem is to make the acceptance ratio as close to unity as possible. One way to do this is to note that Equation (2.17) fixes only the ratio $A(\mu \to \nu)/A(\nu \to \mu)$ of the acceptance ratios for the transitions in either direction between any two states. Thus we are free to multiply both $A(\mu \to \nu)$ and $A(\nu \to \mu)$ by the same factor, and the equation will still be obeyed. The only constraint is that both acceptance ratios should remain between zero and one. In practice then, what we do is to set the larger of the two acceptance ratios to one, and have the other one take whatever value is necessary for the ratio of the two to satisfy (2.17). This ensures that the acceptance ratios will be as large as they can be while still satisfying the relevant conditions, and indeed that the ratio in one direction will be unity, which means that in that direction at least, moves will always be accepted.

However, the best thing we can do to keep the acceptance ratios large is to try to embody in the selection probabilities $g(\mu \to \nu)$ as much as we can of the dependence of $P(\mu \to \nu)$ on the characteristics of the states μ and ν, and put as little as we can in the acceptance ratio. The ideal algorithm is one in which the new states are selected with exactly the correct transition probabilities all the time, and the acceptance ratio is always one. A good algorithm is one in which the acceptance probability is usually close to one. Much of the effort invested in the algorithms described in this book is directed at making the acceptance ratios large.

2.4 Continuous time Monte Carlo

There is a another twist we can add to our Markov process to allow ourselves further freedom about the way in which we choose states, without letting the acceptance ratios get too low. It is called **continuous time Monte Carlo**, or sometimes the **BKL algorithm**, after Bortz, Kalos and Lebowitz (1975), who invented it. Continuous time Monte Carlo is not nearly as widely used as it ought to be; it is an important and powerful technique and many calculations can be helped enormously by making use of it.

Consider a system at low temperature. Such systems are always a problem where Monte Carlo methods are concerned—cool systems move from state to state very slowly in real life and the problem is no less apparent in simulations. A low-temperature system is a good example of the sort of problem system that was described in the last section. Once it reaches equilibrium at its low temperature it will spend a lot of its time in the ground state. Maybe it will spend a hundred consecutive time-steps of the simulation in the ground state, then move up to the first excited state for one time-step and then relax back to the ground state again. Such behaviour is not unreasonable for a cold system but we waste a lot of computer time simulating it. Time-step after time-step our algorithm selects a possible move to some excited state, but the acceptance ratio is very low and virtually all of these possible moves are rejected, and the system just ends up spending most of its time in the ground state.

Well, what if we were to accept that this is the case, and take a look at the acceptance ratio for a move from the ground state to the first excited state, and say to ourselves, "Judging by this acceptance ratio, this system is going to spend a hundred time-steps in the ground state before it accepts a move to the first excited state". Then we could jump the gun by *assuming* that the system will do this, miss out the calculations involved in the intervening useless one hundred time-steps, and progress straight to the one time-step in which something interesting happens. This is the essence of the idea behind the continuous time method. In this technique, we have a time-step which corresponds to a varying length of time, depending on how long we expect the system to remain in its present state before moving to a new one. Then when we come to take the average of our observable Q over many states, we weight the states in which the system spends longest the most heavily—the calculation of the estimator of Q is no more than a time average, so each value Q_μ for Q in state μ should be weighted by how long the system spends in that state.

How can we adapt our previous ideas concerning the transition probabilities for our Markov process to take this new idea into account? Well, assuming that the system is in some state μ, we can calculate how long a time Δt (measured in steps of the simulation) it will stay there for before a move to another state is accepted by considering the "stay-at-home" probability $P(\mu \to \mu)$. The probability that it is still in this same state μ after t time-steps is just

$$[P(\mu \to \mu)]^t = \mathrm{e}^{t \log P(\mu \to \mu)}, \tag{2.18}$$

and so the time-scale Δt is

$$\begin{aligned}\Delta t &= -\frac{1}{\log P(\mu \to \mu)} = -\frac{1}{\log[1 - \sum_{\nu \neq \mu} P(\mu \to \nu)]} \\ &\simeq \frac{1}{\sum_{\nu \neq \mu} P(\mu \to \nu)}.\end{aligned} \tag{2.19}$$

So, if we can calculate this quantity Δt, then rather than wait this many time-steps for a Monte Carlo move to get accepted, we can simply pretend that we have done the waiting and go right ahead and change the state of the system to a new state $\nu \neq \mu$. Which state should we choose for ν? We should choose one at random, but in proportion to $P(\mu \to \nu)$. Thus our continuous time Monte Carlo algorithm consists of the following steps: