Chapter 3

Monte Carlo Simulations

In the present chapter, we describe the basic principles of the Monte Carlo method. In particular, we focus on simulations of systems of a fixed number of particles (N) in a given volume (V) at a temperature (T).

3.1 The Monte Carlo Method

In the previous chapter, we introduced some of the basic concepts of (classical) statistical mechanics. Our next aim is to indicate where the Monte Carlo method comes in. We start from the classical expression for the partition function Q, equation (2.2.5):

$$Q = c \int d\mathbf{p}^N d\mathbf{r}^N \exp[-\mathcal{H}(\mathbf{r}^N\ \mathbf{p}^N)/k_B T], \tag{3.1.1}$$

where $\mathbf{r}^N$ stands for the coordinates of all N particles, and $\mathbf{p}^N$ for the corresponding momenta. The function $\mathcal{H}(\mathbf{q}^N, \mathbf{p}^N)$ is the Hamiltonian of the system. It expresses the total energy of an isolated system as a function of the coordinates and momenta of the constituent particles: $\mathcal{H} = \mathcal{K} + \mathcal{U}$, where $\mathcal{K}$ is the kinetic energy of the system and $\mathcal{U}$ is the potential energy. Finally, c is a constant of proportionality, chosen such that the sum over quantum states in equation (2.1.15) approaches the classical partition function in the limit $\hbar \to 0$. For instance, for a system of N identical atoms, $c = 1/(h^{3N} N!)$. The classical equation corresponding to equation (2.2.1) is

$$\langle A \rangle = \frac{\int d\mathbf{p}^N d\mathbf{r}^N\ A(\mathbf{p}^N, \mathbf{r}^N) \exp[-\beta \mathcal{H}(\mathbf{p}^N, \mathbf{r}^N)]}{\int d\mathbf{p}^N d\mathbf{r}^N\ \exp[-\beta \mathcal{H}(\mathbf{p}^N, \mathbf{r}^N)]}, \tag{3.1.2}$$

where $\beta = 1/k_B T$. In this equation, the observable A has been expressed as a function of coordinates and momenta. As $\mathcal{K}$ is a quadratic function of

the momenta the integration over momenta can be carried out analytically. Hence, averages of functions that depend on momenta only are usually easy to evaluate.[1] The difficult problem is the computation of averages of functions $A(\mathbf{r}^N)$. Only in a few exceptional cases can the multidimensional integral over particle coordinates be computed analytically; in all other cases numerical techniques must be used.

Having thus defined the nature of the numerical problem that we must solve, let us next look at possible solutions. It might appear that the most straightforward approach would be to evaluate $\langle A \rangle$ in equation (3.1.2) by numerical quadrature, for instance using Simpson's rule. It is easy to see, however, that such a method is completely useless even if the number of independent coordinates DN (D is the dimensionality of the system) is still very small $\mathcal{O}(100)$. Suppose that we plan to carry out the quadrature by evaluating the integrand on a mesh of points in the DN-dimensional configuration space. Let us assume that we take m equidistant points along each coordinate axis. The total number of points at which the integrand must be evaluated is then equal to m^{DN}. For all but the smallest systems this number becomes astronomically large, even for small values of m. For instance, if we take 100 particles in three dimensions, and $m = 5$, then we would have to evaluate the integrand at 10^{210} points! Computations of such magnitude cannot be performed in the known universe. And this is fortunate, because the answer that would be obtained would have been subject to a large statistical error. After all, numerical quadratures work best on functions that are smooth over distances corresponding to the mesh size. But for most intermolecular potentials, the Boltzmann factor in equation (3.1.2) is a rapidly varying function of the particle coordinates. Hence an accurate quadrature requires a small mesh spacing (i.e., a large value of m). Moreover, when evaluating the integrand for a dense liquid (say), we would find that for the overwhelming majority of points this Boltzmann factor is vanishingly small. For instance, for a fluid of 100 hard spheres at the freezing point, the Boltzmann factor would be nonzero for 1 out of every 10^{260} configurations!

The preceding example clearly demonstrates that better numerical techniques are needed to compute thermal averages. One such a technique is the Monte Carlo method or, more precisely, the Monte Carlo importance-sampling algorithm introduced in 1953 by Metropolis *et al.* [6]. The application of this method to the numerical simulation of dense molecular systems is the subject of the present chapter.

3.1.1 Importance Sampling

Before discussing importance sampling, let us first look at the simplest Monte Carlo technique, that is, random sampling. Suppose we wish to evaluate

[1] This is not the case when hard constraints are used, see section 11.2.1.

numerically a one-dimensional integral I:

$$I = \int_a^b dx\, f(x). \tag{3.1.3}$$

Instead of using a conventional quadrature where the integrand is evaluated at predetermined values of the abscissa, we could do something else. Note that equation (3.1.3) can be rewritten as

$$I = (b - a)\, \langle f(x) \rangle\,, \tag{3.1.4}$$

where $\langle f(x) \rangle$ denotes the unweighted average of $f(x)$ over the interval $[a, b]$. In brute force Monte Carlo, this average is determined by evaluating $f(x)$ at a large number (say, L) of x values randomly distributed over the interval $[a, b]$. It is clear that, as $L \to \infty$, this procedure should yield the correct value for I. However, as with the conventional quadrature procedure, this method is of little use to evaluate averages such as in equation (3.1.2) because most of the computing is spent on points where the Boltzmann factor is negligible. Clearly, it would be much preferable to sample many points in the region where the Boltzmann factor is large and few elsewhere. This is the basic idea behind importance sampling.

How should we distribute our sampling through configuration space? To see this, let us first consider a simple, one-dimensional example. Suppose we wish to compute the definite integral in equation (3.1.3) by Monte Carlo sampling, but with the sampling points distributed nonuniformly over the interval $[a, b]$ (for convenience we assume $a = 0$ and $b = 1$), according to some nonnegative probability density $w(x)$. Clearly, we can rewrite equation (3.1.3) as

$$I = \int_0^1 dx\, w(x) \frac{f(x)}{w(x)}. \tag{3.1.5}$$

Let us assume that we know that $w(x)$ is the derivative of another (nonnegative, nondecreasing) function $u(x)$, with $u(0) = 0$ and $u(1) = 1$ (these boundary conditions imply that $w(x)$ is normalized). Then I can be written as

$$I = \int_0^1 du\, \frac{f[x(u)]}{w[x(u)]}. \tag{3.1.6}$$

In equation (3.1.6) we have written $x(u)$ to indicate that, if we consider u as the integration variable, then x must be expressed as a function of u. The next step is to generate L random values of u uniformly distributed in the interval $[0, 1]$. We then obtain the following estimate for I:

$$I \approx \frac{1}{L} \sum_{i=1}^{L} \frac{f[x(u_i)]}{w[x(u_i)]}. \tag{3.1.7}$$

What have we gained by rewriting I in this way? The answer depends crucially on our choice for $w(x)$. To see this, let us estimate σ_I^2, the variance in I_L, where I_L denotes the estimate for I obtained from equation (3.1.7) with L random sample points:

$$\sigma_I^2 = \frac{1}{L^2}\sum_{i=1}^{L}\sum_{j=1}^{L}\left\langle\left(\frac{f[x(u_i)]}{w[x(u_i)]} - \langle f/w\rangle\right)\left(\frac{f[x(u_j)]}{w[x(u_j)]} - \langle f/w\rangle\right)\right\rangle, \qquad (3.1.8)$$

where the angular brackets denote the true average, that is, the one that would be obtained in the limit $l \to \infty$. As different samples i and j are assumed to be totally independent, all cross terms in equation (3.1.8) vanish, and we are left with

$$\begin{aligned}\sigma_I^2 &= \frac{1}{L^2}\sum_{i=1}^{L}\left\langle\left(\frac{f[x(u_i)]}{w[x(u_i)]} - \langle f/w\rangle\right)^2\right\rangle \\ &= \frac{1}{L}\left[\langle (f/w)^2\rangle - \langle f/w\rangle^2\right]. \qquad (3.1.9)\end{aligned}$$

Equation (3.1.9) shows that the variance in I still goes as $1/L$, but the magnitude of this variance can be reduced greatly by choosing $w(x)$ such that $f(x)/w(x)$ is a smooth function of x. Ideally, we should have $f(x)/w(x)$ constant, in which case the variance would vanish altogether. In contrast, if $w(x)$ is constant, as is the case for the brute force Monte Carlo sampling, then the relative error in I can become very large. For instance, if we are sampling in a (multidimensional) configuration space of volume Ω, of which only a small fraction f is accessible (for instance, $f = 10^{-260}$, see previous section), then the relative error that results in a brute force MC sampling will be of order $1/(Lf)$. As the integrand in equation (3.1.2) is nonzero only for those configurations where the Boltzmann factor is nonzero, it would clearly be advisable to carry out a nonuniform Monte Carlo sampling of configuration space, such that the weight function w is approximately proportional to the Boltzmann factor. Unfortunately, the simple importance sampling scheme described previously cannot be used to sample multidimensional integrals over configuration space, such as equation (3.1.2). The reason is simply that we do not know how to construct a transformation such as the one from equation (3.1.5) to equation (3.1.6) that would enable us to generate points in configuration space with a probability density proportional to the Boltzmann factor. In fact, a necessary (but not nearly sufficient) condition for the solution to the latter problem is that we must be able to compute analytically the partition function of the system under study. If we could do that for the systems of interest to us, there would be hardly any need for computer simulation.

3.1.2 The Metropolis Method

The closing lines of the previous section suggest that it is in general not possible to evaluate an integral, such as $\int d\mathbf{r}^N \exp[-\beta\mathcal{U}(\mathbf{r}^N)]$, by direct Monte Carlo sampling. However, in many cases, we are not interested in the configurational part of the partition function itself but in averages of the type

$$\langle A \rangle = \frac{\int d\mathbf{r}^N \exp[-\beta\mathcal{U}(\mathbf{r}^N)]A(\mathbf{r}^N)}{\int d\mathbf{r}^N \exp[-\beta\mathcal{U}(\mathbf{r}^N)]}. \tag{3.1.10}$$

Hence, we wish to know the *ratio* of two integrals. What Metropolis *et al.* [6] showed is that it is possible to devise an efficient Monte Carlo scheme to sample such a ratio.[2] To understand the Metropolis method, let us first look more closely at the structure of equation (3.1.10). In what follows we denote the configurational part of the partition function by Z:

$$Z \equiv \int d\mathbf{r}^N \exp[-\beta\mathcal{U}(\mathbf{r}^N)]. \tag{3.1.11}$$

Note that the ratio $\exp(-\beta\mathcal{U})/Z$ in equation (3.1.10) is the probability density of finding the system in a configuration around $\mathbf{r}^N$. Let us denote this probability density by

$$\mathcal{N}(\mathbf{r}^N) \equiv \frac{\exp[-\beta\mathcal{U}(\mathbf{r}^N)]}{Z}.$$

Clearly, $\mathcal{N}(\mathbf{r}^N)$ is nonnegative.

Suppose now that we are somehow able to randomly generate points in configuration space according to this probability distribution $\mathcal{N}(\mathbf{r}^N)$. This means that, on average, the number of points n_i generated per unit volume around a point $\mathbf{r}^N$ is equal to $L\mathcal{N}(\mathbf{r}^N)$, where L is the total number of points that we have generated. In other words,

$$\langle A \rangle \approx \frac{1}{L}\sum_{i=1}^{L} n_i\, A(\mathbf{r}_i^N). \tag{3.1.12}$$

By now the reader is almost certainly confused about the difference, if any, between equation (3.1.12) and equation (3.1.7) of section 3.1.1. The difference is that in the case of equation (3.1.7) we know *a priori* the probability of sampling a point in a (hyper)volume $d\mathbf{r}^N$ around $\mathbf{r}^N$. In other words we know both $\exp[-\beta\mathcal{U}(\mathbf{r}^N)]$ *and* Z. In contrast, in equation (3.1.12) we know only $\exp[-\beta\mathcal{U}(\mathbf{r}^N)]$; that is, we know only the relative but not the absolute probability of visiting different points in configuration space. This may sound

[2] An interesting account of the early history of the Metropolis method may be found in refs. [1,46].

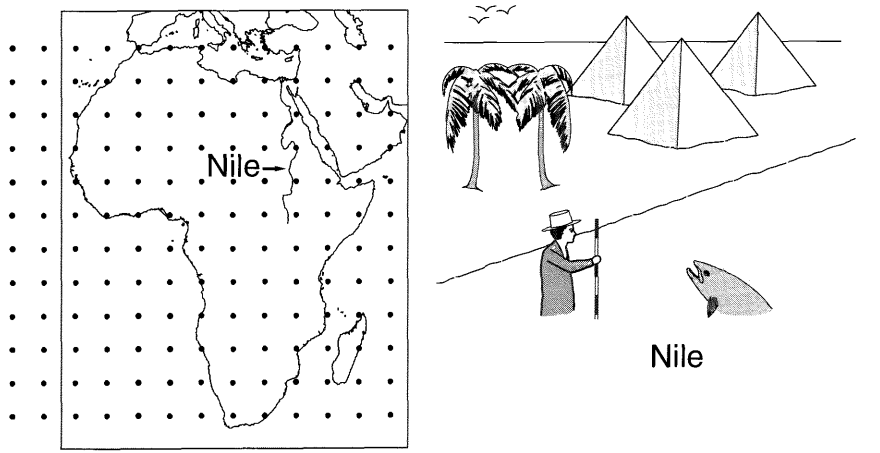

Figure 3.1: Measuring the depth of the Nile: a comparison of conventional quadrature (left), with the Metropolis scheme (right).

rather abstract: let us therefore try to clarify the difference with the help of a simple example (see Figure 3.1). In this figure, we compare two ways of measuring the depth of the river Nile, by conventional quadrature (left) and by Metropolis sampling; that is, the construction of an importance-weighted random walk (right). In the conventional quadrature scheme, the value of the integrand is measured at a predetermined set of points. As the choice of these points does not depend on the value of the integrand, many points may be located in regions where the integrand vanishes. In contrast, in the Metropolis scheme, a random walk is constructed through that region of space where the integrand is nonnegligible (i.e., through the Nile itself). In this random walk, a trial move is rejected if it takes you out of the water and is accepted otherwise. After *every* trial move (accepted or not), the depth of the water is measured. The (unweighted) average of all these measurements yields an estimate of the average depth of the Nile. This, then, is the essence of the Metropolis method. In principle, the conventional quadrature scheme would also give results for the *total* area of the Nile. In the importance sampling scheme, however, information on the total area cannot be obtained directly, since this quantity is similar to Z.

Let us next consider how to generate points in configuration space with a relative probability proportional to the Boltzmann factor. The general approach is first to prepare the system in a configuration $\mathbf{r}^N$, which we denote by o (old), that has a nonvanishing Boltzmann factor $\exp[-\beta\mathcal{U}(o)]$. This configuration, for example, may correspond to a regular crystalline lattice with no hard-core overlaps. Next, we generate a new trial configuration $\mathbf{r}'^N$,

which we denote by n (new), by adding a small random displacement Δ to o. The Boltzmann factor of this trial configuration is $\exp[-\beta\mathcal{U}(n)]$. We must now decide whether we will accept or reject the trial configuration. Many rules for making this decision satisfy the constraint that on average the probability of finding the system in a configuration n is proportional to $\mathcal{N}(n)$. Here we discuss only the Metropolis scheme, because it is simple and generally applicable.

Let us now "derive" the Metropolis scheme to determine the transition probability $\pi(o \to n)$ to go from configuration o to n. It is convenient to start with a thought experiment (actually a thought simulation). We carry out a very large number (say M) Monte Carlo simulations in parallel, where M is much larger than the total number of accessible configurations. We denote the number of points in any configuration o by $m(o)$. We wish that, on average, $m(o)$ is proportional to $\mathcal{N}(o)$. The matrix elements $\pi(o \to n)$ must satisfy one obvious condition: they do not destroy such an equilibrium distribution once it is reached. This means that, in equilibrium, the average number of accepted trial moves that result in the system leaving state o must be exactly equal to the number of accepted trial moves from all other states n to state o. It is convenient to impose a much stronger condition; namely, that in equilibrium the average number of accepted moves from o to any other state n is exactly canceled by the number of reverse moves. This detailed balance condition implies the following:

$$\mathcal{N}(o)\pi(o \to n) = \mathcal{N}(n)\pi(n \to o). \tag{3.1.13}$$

Many possible forms of the transition matrix $\pi(o \to n)$ satisfy equation (3.1.13). Let us look how $\pi(o \to n)$ is constructed in practice. We recall that a Monte Carlo move consists of two stages. First, we perform a trial move from state o to state n. We denote the transition matrix that determines the probability of performing a trial move from o to n by $\alpha(o \to n)$, where α is usually referred to as the underlying matrix of the Markov chain [47]. The next stage is the decision to either accept or reject this trial move. Let us denote the probability of accepting a trial move from o to n by $\mathrm{acc}(o \to n)$. Clearly,

$$\pi(o \to n) = \alpha(o \to n) \times \mathrm{acc}(o \to n). \tag{3.1.14}$$

In the original Metropolis scheme, α is chosen to be a symmetric matrix ($\alpha(o \to n) = \alpha(n \to o)$). However, in later sections we shall see several examples where α is *not* symmetric. If α is symmetric, we can rewrite equation (3.1.13) in terms of the $\mathrm{acc}(o \to n)$:

$$\mathcal{N}(o) \times \mathrm{acc}(o \to n) = \mathcal{N}(n) \times \mathrm{acc}(n \to o). \tag{3.1.15}$$

From equation (3.1.15) follows

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{\mathcal{N}(n)}{\mathcal{N}(o)} = \exp\{-\beta[\mathcal{U}(n) - \mathcal{U}(o)]\}. \tag{3.1.16}$$

Again, many choices for $\mathrm{acc}(o \to n)$ satisfy this condition (and the obvious condition that the probability $\mathrm{acc}(o \to n)$ cannot exceed 1). The choice of Metropolis *et al.* is

$$\begin{aligned} \mathrm{acc}(o \to n) &= \mathcal{N}(n)/\mathcal{N}(o) \quad \text{if } \mathcal{N}(n) < \mathcal{N}(o) \\ &= 1 \qquad\qquad\quad \text{if } \mathcal{N}(n) \geq \mathcal{N}(o). \end{aligned} \tag{3.1.17}$$

Other choices for $\mathrm{acc}(o \to n)$ are possible (for a discussion, see for instance [19]), but the original choice of Metropolis *et al.* appears to result in a more efficient sampling of configuration space than most other strategies that have been proposed.

In summary, then, in the Metropolis scheme, the transition probability for going from state o to state n is given by

$$\begin{aligned} \pi(o \to n) &= \alpha(o \to n) & \mathcal{N}(n) \geq \mathcal{N}(o) \\ &= \alpha(o \to n)[\mathcal{N}(n)/\mathcal{N}(o)] & \mathcal{N}(n) < \mathcal{N}(o) \\ \pi(o \to o) &= 1 - \textstyle\sum_{n \neq o} \pi(o \to n). \end{aligned} \tag{3.1.18}$$

Note that we still have not specified the matrix α, except for the fact that it must be symmetric. This reflects considerable freedom in the choice of our trial moves. We will come back to this point in subsequent sections.

One thing that we have not yet explained is how to decide whether a trial move is to be accepted or rejected. The usual procedure is as follows. Suppose that we have generated a trial move from state o to state n, with $\mathcal{U}(n) > \mathcal{U}(o)$. According to equation (3.1.16) this trial move should be accepted with a probability

$$\mathrm{acc}(o \to n) = \exp\{-\beta[\mathcal{U}(n) - \mathcal{U}(o)]\} < 1.$$

In order to decide whether to accept or reject the trial move, we generate a random number, denoted by Ranf, from a uniform distribution in the interval $[0, 1]$. Clearly, the probability that Ranf is less than $\mathrm{acc}(o \to n)$ is equal to $\mathrm{acc}(o \to n)$. We now accept the trial move if $\mathrm{Ranf} < \mathrm{acc}(o \to n)$ and reject it otherwise. This rule guarantees that the probability to accept a trial move from o to n is indeed equal to $\mathrm{acc}(o \to n)$. Obviously, it is very important that our random-number generator does indeed generate numbers uniformly in the interval $[0, 1]$. Otherwise the Monte Carlo sampling will be biased. The quality of random-number generators should never be taken for granted. A good discussion of random-number generators can be found in *Numerical Recipes* [33] and in *Monte Carlo Methods* by Kalos and Whitlock [32].

Thus far, we have not mentioned another condition that $\pi(o \to n)$ should satisfy, namely that it is *ergodic* (i.e., every accessible point in configuration space can be reached in a finite number of Monte Carlo steps from any other point). Although some simple MC schemes are guaranteed to be ergodic,

these are often not the most efficient schemes. Conversely, many efficient Monte Carlo schemes have either not been proven to be ergodic or, worse, been proven to be nonergodic. The solution is usually to mix the efficient, nonergodic scheme with an occasional trial move of the less-efficient but ergodic scheme. The method as a whole will then be ergodic (at least, in principle).

At this point, we should stress that, in the present book, we focus on Monte Carlo methods to model phenomena that do not depend on time. In the literature one can also find dynamic Monte Carlo schemes. In such dynamic algorithms, Monte Carlo methods are used to generate a numerical solution of the master equation that is supposed to describe the time evolution of the system under study. These dynamic techniques fall outside the scope of this book. The reader interested in dynamic MC schemes is referred to the relevant literature, for example Ref. [48] and references therein.

3.2 A Basic Monte Carlo Algorithm

It is difficult to talk about Monte Carlo or Molecular Dynamics programs in abstract terms. The best way to explain how such programs work is to write them down. This will be done in the present section.

Most Monte Carlo or Molecular Dynamics programs are only a few hundred to several thousand lines long. This is very short compared to, for instance, a typical quantum-chemistry code. For this reason, it is not uncommon that a simulator will write many different programs that are tailor-made for specific applications. The result is that there is no such thing as a standard Monte Carlo or Molecular Dynamics program. However, the cores of most MD/MC programs are, if not identical, at least very similar. Next, we shall construct such a core. It will be very rudimentary, and efficiency has been traded for clarity. But it should demonstrate how the Monte Carlo method works.

3.2.1 The Algorithm

The prime purpose of the kind of Monte Carlo or Molecular Dynamics program that we shall be discussing is to compute equilibrium properties of classical many-body systems. From now on, we shall refer to such programs simply as MC or MD programs, although it should be remembered that there exist many other applications of the Monte Carlo method (and, to a lesser extent, of the Molecular Dynamics method). Let us now look at a simple Monte Carlo program.

In the previous section, the Metropolis method was introduced as a Markov process in which a random walk is constructed in such a way that the

probability of visiting a particular point $\mathbf{r}^N$ is proportional to the Boltzmann factor $\exp[-\beta\mathcal{U}(\mathbf{r}^N)]$. There are many ways to construct such a random walk. In the approach introduced by Metropolis *et al.* [6], the following scheme is proposed:

1. Select a particle at random, and calculate its energy $\mathcal{U}(\mathbf{r}^N)$.
2. Give the particle a random displacement, $r' = r + \Delta$, and calculate its new energy $\mathcal{U}(\mathbf{r'}^N)$.
3. Accept the move from $\mathbf{r}^N$ to $\mathbf{r'}^N$ with probability

$$\mathrm{acc}(o \to n) = \min\left(1, \exp\{-\beta[\mathcal{U}(\mathbf{r'}^N) - \mathcal{U}(\mathbf{r}^N)]\}\right). \tag{3.2.1}$$

An implementation of this basic Metropolis scheme is shown in Algorithms 1 and 2.

3.2.2 Technical Details

In this section, we discuss a number of computational tricks that are of great practical importance for the design of an efficient simulation program. It should be stressed that most of these tricks, although undoubtedly very useful, are not unique and have no deep physical significance. But this does not imply that the use of such computational tools is free of risks or subtleties. Ideally, schemes to save computer time should not affect the results of a simulation in a systematic way. Yet, in some cases, time-saving tricks do have a measurable effect on the outcome of a simulation. This is particularly true for the different procedures used to avoid explicit calculation of intermolecular interactions between particles that are far apart. Fortunately, once this is recognized, it is usually possible to estimate the undesirable side effect of the time-saving scheme and correct for it.

Boundary Conditions

Monte Carlo and Molecular Dynamics simulations of atomic or molecular systems aim to provide information about the properties of a macroscopic sample. Yet, the number of degrees of freedom that can be conveniently handled in present-day computers ranges from a few hundred to a few million. Most simulations probe the structural and thermodynamical properties of a system of a few hundred to a few thousand particles. Clearly, this number is still far removed from the thermodynamic limit. To be more precise, for such small systems it cannot be safely assumed that the choice of the boundary conditions (e.g., free or hard or periodic) has a negligible effect on the properties of the system. In fact, in a three-dimensional N-particle system

Algorithm 1 (Basic Metropolis Algorithm)

```
  PROGRAM mc                              basic Metropolis algorithm

  do icycl=1,ncycl                        perform ncycl MC cycles
    call mcmove                           displace a particle
    if (mod(icycl,nsamp).eq.0)
 +    call sample                        sample averages
  enddo
  end
```

Comments to this algorithm:

1. *Subroutine* `mcmove` *attempts to displace a randomly selected particle (see Algorithm 2).*
2. *Subroutine* `sample` *samples quantities every* `nsampth` *cycle.*

Algorithm 2 (Attempt to Displace a Particle)

```
  SUBROUTINE mcmove                       attempts to displace a particle

  o=int(ranf()*npart)+1                   select a particle at random
  call ener(x(o),eno)                     energy old configuration
  xn=x(o)+(ranf()-0.5)*delx               give particle random displacement
  call ener(xn,enn)                       energy new configuration
  if (ranf().lt.exp(-beta                 acceptance rule (3.2.1)
 +   *(enn-eno)) x(o)=xn                  accepted: replace x(o) by xn
  return
  end
```

Comments to this algorithm:

1. *Subroutine* `ener` *calculates the energy of a particle at the given position.*
2. *Note that, if a configuration is rejected, the old configuration is retained.*
3. *The* `ranf()` *is a random number uniform in* $[0, 1]$.

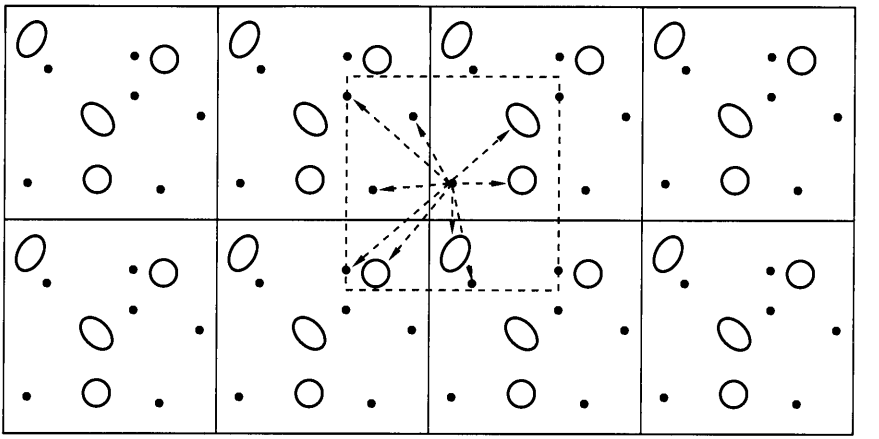

Figure 3.2: Schematic representation of periodic boundary conditions.

with free boundaries, the fraction of all molecules that is at the surface is proportional to $N^{-1/3}$. For instance, in a simple cubic crystal of 1000 atoms, some 49% of all atoms are at the surface, and for 10^6 atoms this fraction has decreased to only 6%.

In order to simulate bulk phases it is essential to choose boundary conditions that mimic the presence of an infinite bulk surrounding our N-particle model system. This is usually achieved by employing periodic boundary conditions. The volume containing the N particles is treated as the primitive cell of an infinite periodic lattice of identical cells (see Figure 3.2). A given particle (i, say) now interacts with all other particles in this infinite periodic system, that is, all other particles in the same periodic cell and all particles (including its own periodic image) in all other cells. For instance, if we assume that all intermolecular interactions are pairwise additive, then the total potential energy of the N particles in any one periodic box is

$$\mathcal{U}_{\text{tot}} = \frac{1}{2} {\sum_{i,j,\mathbf{n}}}' u(|\mathbf{r}_{ij} + \mathbf{n}L|),$$

where L is the diameter of the periodic box (assumed cubic, for convenience) and $\mathbf{n}$ is an arbitrary vector of three integer numbers, while the prime over the sum indicates that the term with $i = j$ is to be excluded when $\mathbf{n} = \mathbf{0}$. In this very general form, periodic boundary conditions are not particularly useful, because to simulate bulk behavior, we had to rewrite the potential energy as an infinite sum rather than a finite one.[3] In practice, however, we

[3]In fact, in the first MC simulation of three-dimensional Lennard-Jones particles, Wood and Parker [49] discuss the use of such infinite sums in relation to the now conventional approach discussed here.

are often dealing with short-range interactions. In that case it is usually permissible to truncate all intermolecular interactions beyond a certain cutoff distance r_c. How this is done in practice is discussed next.

Although the use of periodic boundary conditions proves to be a surprisingly effective method for simulating homogeneous bulk systems, one should always be aware that the use of such boundary conditions may lead to spurious correlations not present in a truly macroscopic bulk system. In particular, one consequence of the periodicity of the model system is that only those fluctuations are allowed that have a wavelength compatible with the periodic lattice: the longest wavelength that still fits in the periodic box is the one for which $\lambda = L$. If long wavelength fluctuations are expected to be important (as, for instance, in the vicinity of a continuous phase transition), then one should expect problems with the use of periodic boundary conditions. Another unphysical effect that is a manifestation of the use of periodic boundary conditions is that the radial distribution function $g(r)$ of a dense atomic fluid is found to be not exactly isotropic [50].

Finally, it is useful to point out one common misconception about periodic boundary conditions, namely, the idea that the boundary of the periodic box itself has any special significance. It has none. The origin of the periodic lattice of primitive cells may be chosen anywhere, and this choice will not affect any property of the model system under study. In contrast, what *is* fixed is the shape of the periodic cell and its orientation.

Truncation of Interactions

Let us now consider the case that we perform a simulation of a system with *short-range* interactions. In this context, short-ranged means that the total potential energy of a given particle i is dominated by interactions with neighboring particles that are closer than some cutoff distance r_c. The error that results when we ignore interactions with particles at larger distances can be made arbitrarily small by choosing r_c sufficiently large. If we use periodic boundary conditions, the case that r_c is less than $L/2$ (half the diameter of the periodic box) is of special interest because in that case we need to consider the interaction of a given particle i only with the nearest periodic image of any other particles j (see the dotted box in Figure 3.2). If the intermolecular potential is not rigorously zero for $r \geq r_c$, truncation of the intermolecular interactions at r_c will result in a systematic error in $\mathcal{U}^{tot}$. If the intermolecular interactions decay rapidly, one may correct for the systematic error by adding a tail contribution to $\mathcal{U}^{tot}$:

$$\mathcal{U}^{tot} = \sum_{i<j} u_c(r_{ij}) + \frac{N\rho}{2} \int_{r_c}^{\infty} dr\, u(r) 4\pi r^2, \tag{3.2.2}$$

where u_c stands for the truncated potential energy function and ρ is the average number density. In writing down this expression, it is implicitly assumed that the radial distribution function $g(r) = 1$ for $r > r_c$. Clearly, the nearest periodic image convention can be applied only if the tail correction is small. From equation (3.2.2) it can be seen that the tail correction to the potential energy is infinite unless the potential energy function $u(r)$ decays more rapidly than r^{-3} (in three dimensions). This condition is satisfied if the long-range interaction between molecules is dominated by dispersion forces. However, for the very important case of Coulomb and dipolar interactions, the tail correction diverges and hence the nearest-image convention cannot be used for such systems. In such cases, the interactions with all periodic images should be taken into account explicitly. Ways to do this are described in Chapter 12.1.

Several factors make truncation of the potential a tricky business. First of all, it should be realized that, although the absolute value of the potential energy function decreases with interparticle separation r, for sufficiently large r, the number of neighbors is a rapidly increasing function of r. In fact, the number of particles at a distance r of a given atom increases asymptotically as r^{d-1}, where d is the dimensionality of the system. As an example, let us compute the effect of truncating the pair potential for a simple example — the three-dimensional Lennard-Jones fluid. The pair potential for this rather popular model system is given by

$$u^{lj}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]. \tag{3.2.3}$$

The average potential energy (in three dimensions) of any given atom i is given by

$$u_i = (1/2) \int_0^\infty dr\, 4\pi r^2 \rho(r) u(r),$$

where $\rho(r)$ denotes the average number density at a distance r from a given atom i. The factor $(1/2)$ has been included to correct for double counting of intermolecular interactions. If we truncate the potential at a distance r_c, we ignore the tail contribution u^{tail}:

$$u^{tail} \equiv (1/2) \int_{r_c}^\infty dr\, 4\pi r^2 \rho(r) u(r), \tag{3.2.4}$$

where we have dropped the subscript i, because all atoms in the system are identical. To simplify the calculation of u^{tail}, we assume that for $r \geq r_c$, the density $\rho(r)$ is equal to the average number density ρ. If $u(r)$ is the Lennard-

Jones potential, we find for u^{tail}

$$\begin{aligned} u^{tail} &= \frac{1}{2} 4\pi\rho \int_{r_c}^{\infty} dr\, r^2 u(r) \\ &= \frac{1}{2} 16\pi\rho\epsilon \int_{r_c}^{\infty} dr\, r^2 \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \\ &= \frac{8}{3}\pi\rho\epsilon\sigma^3 \left[\frac{1}{3} \left(\frac{\sigma}{r_c}\right)^{9} - \left(\frac{\sigma}{r_c}\right)^{3} \right]. \end{aligned} \tag{3.2.5}$$

For a cutoff distance $r_c = 2.5\ \sigma$ the potential has decayed to a value that is about 1/60th of the well depth. This seems to be a very small value, but in fact the tail correction is usually nonnegligible. For instance, at a density $\rho\sigma^3 = 1$, we find $u^{tail} = -0.535\epsilon$. This number is certainly not negligible compared to the total potential energy per atom (almost 10% at a typical liquid density); hence although we can truncate the potential at 2.5 σ, we cannot ignore the effect of this truncation.

There are several ways to truncate potentials in a simulation. Although the methods are designed to yield similar results, it should be realized that they yield results that may differ significantly, in particular in the vicinity of critical points [51–53] (see Figure 3.3). Often used methods to truncate the potential are

1. Simple truncation
2. Truncation and shift
3. Minimum image convention.

Simple Truncation The simplest method to truncate potentials is to ignore all interaction beyond r_c, the potential that is simulated is

$$u^{trunc}(r) = \begin{cases} u^{lj}(r) & r \le r_c \\ 0 & r > r_c \end{cases}. \tag{3.2.6}$$

As already explained, this may result in an appreciable error in our estimate of the potential energy of the true Lennard-Jones potential (3.2.3). Moreover, as the potential changes discontinuously at r_c, a truncated potential is not particularly suitable for a Molecular Dynamics simulation. It can, however, be used in Monte Carlo simulations. In that case, one should be aware that there is an "impulsive" contribution to the pressure due to the discontinuous change of the potential at r_c. That contribution can by no means be ignored.

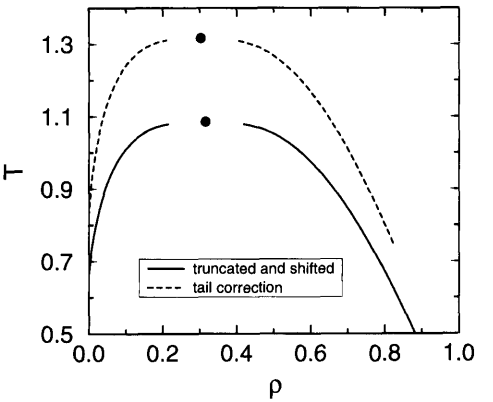

Figure 3.3: Vapor-liquid coexistence curves of various three-dimensional Lennard-Jones fluids: effect of the truncation of the potential on the location of the critical point (large black dots). The upper curve gives the phase envelope for the full Lennard-Jones potential (i.e., a truncated potential with tail correction); the lower curve gives the envelope for the potential that is used in most Molecular Dynamics simulations (truncated and shifted potential with $r_c = 2.5\sigma$), data from [53].

For instance, for the three-dimensional Lennard-Jones system,

$$\begin{aligned} \Delta P^{\text{imp}} &= \frac{8\pi}{3}\rho^2 g(r_c)\epsilon\sigma^3\left[\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3\right] \\ &\approx \frac{8\pi}{3}\rho^2\epsilon\sigma^3\left[\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3\right]. \end{aligned} \tag{3.2.7}$$

It is rare, however, to see this impulsive correction to the pressure applied in simulations of systems with truncated potentials. Usually, it is simply assumed that we can correct for the truncation of the intermolecular potential by adding the correction given by equation (3.2.5) to the potential energy. The corresponding correction to the pressure is

$$\begin{aligned} \Delta P^{\text{tail}} &= (1/2)4\pi\rho^2\int_{r_c}^{\infty} dr\, r^2 \mathbf{r}\cdot\mathbf{f}(r) \\ &= \frac{16}{3}\pi\rho^2\epsilon\sigma^3\left[\frac{2}{3}\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3\right]. \end{aligned} \tag{3.2.8}$$

But, as is immediately obvious from a comparison of equations (3.2.7) and (3.2.8), the impulsive correction to the pressure is not equivalent to the tail

correction. Rather, the impulsive pressure is the contribution that must be included if one wishes to compute the true pressure of a system with a truncated potential, whereas the tail correction should be included to obtain an estimate of the pressure in a system with untruncated interactions.

Truncated and Shifted In Molecular Dynamics simulations, it is common to use another procedure: the potential is truncated and shifted, such that the potential vanishes at the cutoff radius:

$$u^{tr-sh}(r) = \begin{cases} u^{lj}(r) - u^{lj}(r_c) & r \leq r_c \\ 0 & r > r_c \end{cases} . \tag{3.2.9}$$

In this case, there are no discontinuities in the intermolecular potential and hence no impulsive corrections to the pressure. The advantage of using such a truncated and shifted potential is that the intermolecular forces are always finite.[4] This is important because impulsive forces cannot be handled in those Molecular Dynamics algorithms to integrate the equations of motion that are based on a Taylor expansion of the particle positions. Of course, the potential energy and pressure of a system with a truncated and shifted potential differ from the corresponding properties of both the models with untruncated and with truncated but unshifted pair potentials. But, as before, we can approximately correct for the effect of the modification of the intermolecular potential on both the potential energy and the pressure. For the pressure, the tail correction is the same as in equation (3.2.8). For the potential energy, we must add to the long-range correction (3.2.5) a contribution equal to the average number of particles that are within a distance r_c from a given particle, multiplied by half the value of the (untruncated) pair potential at r_c. The factor one-half is included to correct for overcounting of the intermolecular interactions. One should be extremely careful when applying truncated and shifted potentials in models with anisotropic interactions. In that case, truncation should not be carried out at a fixed value of the distance between the molecular centers of mass but at a point where the pair potential has a fixed value, because otherwise the potential cannot be shifted to 0 at the point where it is truncated. For Monte Carlo simulations, this is not serious, but for Molecular Dynamics simulations this would be quite disastrous, as the system would no longer conserve energy, unless the impulsive forces due to the truncating and shifting are taken into account explicitly.

Minimum Image Convention Sometimes the minimum image convention is used. The truncation is in this case not at a spherical cutoff; instead the

[4] The first derivative of the force is discontinuous at the cutoff radius; some authors remove this discontinuity as well (for more details, see [19]).

interaction with the (nearest image) of all the particles in the simulation box is calculated. As a consequence, the potential is not a constant on the surface of a cube around a given particle. Hence, for the same reasons as mentioned in the previous paragraph, the simple minimum image convention should never be used in Molecular Dynamics simulations.

In the preceding, we described some details on how the energy should be calculated. The implementation of a simple, order-N^2, algorithm to compute the energy will be discussed in section 4.2.2 in the context of a Molecular Dynamics simulation (see Algorithm 5). More advanced schemes to simulate large systems efficiently are described in Appendix F.

Initialization

To start the simulation, we should assign initial positions to all particles in the system. As the equilibrium properties of the system do not (or, at least, should not) depend on the choice of initial conditions, all reasonable initial conditions are in principle acceptable. If we wish to simulate the solid state of a particular model system, it is logical to prepare the system in the crystal structure of interest. In contrast, if we are interested in the fluid phase, we simply prepare the system in any convenient crystal structure. This crystal subsequently melts, because at the temperature and density of a typical liquid-state point, the solid state is not thermodynamically stable. Actually, one should be careful here, because the crystal structure may be metastable, even if it is not absolutely stable. For this reason, it is unwise to use a crystal structure as the starting configuration of a liquid close to the freezing curve. In such cases, it is better to use the final (liquid) configuration of a system at a higher temperature or lower density, where the solid is unstable and has melted spontaneously. In any event, it is usually preferable to use the final (well-equilibrated) configuration of an earlier simulation at a nearby state point as the starting configuration for a new run and adjust the temperature and density to the desired values.

The equilibrium properties of a system should not depend on the initial conditions. If such a dependence nevertheless is observed in a simulation, there are two possibilities. The first is that our results reflect the fact that the system that we simulate really behaves nonergodically. This is the case, for instance, in glassy materials or low-temperature, substitutionally disordered alloys. The second (and much more likely) explanation is the system we simulate is ergodic but our sampling of configuration space is inadequate; in other words, we have not yet reached equilibrium.

Reduced Units

In simulations it is often convenient to express quantities such as temperature, density, pressure, and the like in reduced units. This means that we choose a convenient unit of energy, length and mass and then express all

other quantities in terms of these basic units. In the example of a Lennard-Jones system, we use a pair potential that is of the form $u(r) = \epsilon f(r/\sigma)$ (see equation (3.2.3)). A natural (though not unique) choice for our basic units is the following:

- Unit of length, σ
- Unit of energy, ϵ
- Unit of mass, m (the mass of the atoms in the system)

and from these basic units, all other units follow. For instance, our unit of time is

$$\sigma\sqrt{m/\epsilon}$$

and the unit of temperature is

$$\epsilon/k_B.$$

In terms of these reduced units, denoted with superscript *, the reduced pair potential $u^* \equiv u/\epsilon$ is a dimensionless function of the reduced distance $r^* \equiv r/\sigma$. For instance, the reduced form for the Lennard-Jones potential is

$$u^{*lj}(r^*) = 4\left[\left(\frac{1}{r^*}\right)^{12} - \left(\frac{1}{r^*}\right)^{6}\right]. \tag{3.2.10}$$

With these conventions we can define the following reduced units: the potential energy $U^* = U\epsilon^{-1}$, the pressure $P^* = P\sigma^3\epsilon^{-1}$, the density $\rho^* = \rho\sigma^3$, and the temperature $T^* = k_B T\epsilon^{-1}$.

One may wonder why it is convenient to introduce reduced units. The most important reason is that (infinitely) many combinations of ρ, T, ϵ, and σ all correspond to the same state in reduced units. This is the law of corresponding states: the same simulation of a Lennard-Jones model can be used to study Ar at 60 K and a density of 840 kg/m^3 and Xe at 112 K and a density of 1617 kg/m^3. In reduced units, both simulations correspond to the state point $\rho^* = 0.5$, $T^* = 0.5$. If we had not used reduced units, we might have easily missed the equivalence of these two simulations. Another, practical, reason for using reduced units is the following: when we work with real (SI) units, we find that the absolute numerical values of the quantities that we are computing (e.g., the average energy of a particle or its acceleration) are either much less or much larger than 1. If we multiply several such quantities using standard floating-point multiplication, we face a distinct risk that, at some stage, we will obtain a result that creates an overflow or underflow. Conversely, in reduced units, almost all quantities of interest are of order 1 (say, between 10^{-3} and 10^3). Hence, if we suddenly find a very large (or very small) number in our simulations (say, 10^{42}), then there is a good chance that we have made an error somewhere. In other words, reduced

Quantity	Reduced units		Real units
temperature	$T^* = 1$	$\leftrightarrow$	$T = 119.8$ K
density	$\rho^* = 1.0$	$\leftrightarrow$	$\rho = 1680$ kg/m^3
time	$\Delta t^* = 0.005$	$\leftrightarrow$	$\Delta t = 1.09 \times 10^{-14}$ s
pressure	$P^* = 1$	$\leftrightarrow$	$P = 41.9$ MPa

Table 3.1: Translation of reduced units to real units for Lennard-Jones argon ($\epsilon/k_B = 119.8$ K, $\sigma = 3.405 \times 10^{-10}$m, $M = 0.03994$ kg/mol)

units make it easier to spot errors. Simulation results that are obtained in reduced units can always be translated back into real units. For instance, if we wish to compare the results of a simulation on a Lennard-Jones model at $T^* = 1$ and $P^* = 1$ with experimental data for argon ($\epsilon/k_B = 119.8$ K, $\sigma = 3.405 \times 10^{-10}$ m, $M = 0.03994$ kg/mol), then we can use the translation given in Table 3.1 to convert our simulation parameters to real SI units.[5]

3.2.3 Detailed Balance versus Balance

Throughout this book we use the condition of detailed balance as a test of the validity of a Monte Carlo scheme. However, as stated before, the detailed-balance condition is sufficient, but not necessary. Manousiouthakis and Deem [54] have shown that the weaker "balance condition" is a necessary and sufficient requirement. A consequence of this proof is that one has more freedom in developing Monte Carlo moves. For example, in the simple Monte Carlo scheme shown in Algorithm 2 we select a particle at random and give it a random displacement. During the next trial move, the *a priori* probability to select the *same* particle is the same. Thus the reverse trial move has the same *a priori* probability and detailed balance is satisfied. An alternative scheme is to attempt moving all particles sequentially, i.e., first an attempt to move particle one, followed by an attempt to move particle two, etc. In this sequential scheme, the probability that a single-particle move is followed by its reverse is zero. Hence, this scheme clearly violates detailed balance. However, Manousiouthakis and Deem have shown that such a sequential updating scheme does obey balance and does therefore (usually — see Ref. [54]) result in correct MC sampling.

We stress that the detailed-balance condition remains an important guiding principle in developing novel Monte Carlo schemes. Moreover, most algorithms that do not satisfy detailed balance are simply wrong. This is true in particular for "composite" algorithms that combine different trial moves. Therefore, we suggest that it is good practice to impose detailed balance

[5] In what follows we will always use reduced units, unless explicitly indicated otherwise. We, therefore, omit the superscript * to denote reduced units.

when writing a code. Of course, if subsequently it turns out that the performance of a working program can be improved considerably by using a "balance-only" algorithm, then it is worth implementing it. At present, we are not aware of examples in the literature where a "balance-only" algorithm is shown to be much faster than its "detailed-balance" counterpart.

3.3 Trial Moves

Now that we have specified the general structure of the Metropolis algorithm, we should consider its implementation. We shall not go into the problem of selecting intermolecular potentials for the model system under study. Rather, we shall simply assume that we have an atomic or molecular model system in a suitable starting configuration and that we have specified all intermolecular interactions. We must now set up the underlying Markov chain, that is, the matrix α. In more down to earth terms: we must decide how we are going to generate trial moves. We should distinguish between trial moves that involve only the molecular centers of mass and those that change the orientation or possibly even the conformation of a molecule.

3.3.1 Translational Moves

We start our discussion with trial moves of the molecular centers of mass. A perfectly acceptable method for creating a trial displacement is to add random numbers between $-\Delta/2$ and $+\Delta/2$ to the x, y, and z coordinates of the molecular center of mass:

$$\begin{aligned} x_i' &\rightarrow x_i + \Delta\,(\text{Ranf} - 0.5) \\ y_i' &\rightarrow y_i + \Delta\,(\text{Ranf} - 0.5) \\ z_i' &\rightarrow z_i + \Delta\,(\text{Ranf} - 0.5), \end{aligned} \tag{3.3.1}$$

where Ranf are random numbers uniformly distributed between 0 and 1. Clearly, the reverse trial move is equally probable (hence, α is symmetric).[6] We are now faced with two questions: how large should we choose Δ? and should we attempt to move all particles simultaneously or one at a time? In the latter case we should pick the molecule that is to be moved at random to ensure that the underlying Markov chain remains symmetric. All

[6] Although almost all published MC simulations on atomic and molecular systems generate trial displacements in a cube centered around the original center of mass position, this is by no means the only possibility. Sometimes, it is more convenient to generate trial moves in a spherical volume, and it is not even necessary that the distribution of trial moves in such a volume be uniform, as long as it has inversion symmetry. For an example of a case where another sampling scheme is preferable, see ref. [55].

other things being equal, we should choose the most efficient sampling procedure. But, to this end, we must first define what we mean by *efficient sampling*. In very vague terms, sampling is efficient if it gives you good value for money. Good value in a simulation corresponds to high statistical accuracy, and "money" is simply *money*: the money that buys your computer time and even your own time. For the sake of the argument, we assume the average scientific programmer is poorly paid. In that case we have to worry only about your computer budget.[7] Then we could use the following definition of an optimal sampling scheme: a Monte Carlo sampling scheme can be considered optimal if it yields the lowest statistical error in the quantity to be computed for a given expenditure of computing budget. Usually, computing budget is equivalent to CPU time.

From this definition it is clear that, in principle, a sampling scheme may be optimal for one quantity but not for another. Actually, the preceding definition is all but useless in practice (as are most definitions). For instance, it is just not worth the effort to measure the error estimate in the pressure for a number of different Monte Carlo sampling schemes in a series of runs of fixed length. However, it is reasonable to assume that the mean-square error in the observables is inversely proportional to the number of uncorrelated configurations visited in a given amount of CPU time. And the number of independent configurations visited is a measure for the distance covered in configuration space. This suggests a more manageable, albeit rather ad hoc, criterion to estimate the efficiency of a Monte Carlo sampling scheme: the sum of the squares of all accepted trial displacements divided by computing time. This quantity should be distinguished from the mean-squared displacement per unit of computing time, because the latter quantity goes to 0 in the absence of diffusion (e.g., in a solid or a glass), whereas the former does not.

Using this criterion it is easy to show that for simulations of condensed phases it is usually advisable to perform random displacements of one particle at a time (as we shall see later, the situation is different for correlated displacements). To see why random single-particle moves are preferred, consider a system of N spherical particles, interacting through a potential energy function $\mathcal{U}(\mathbf{r}^N)$. Typically, we expect that a trial move will be rejected if the potential energy of the system changes by much more than k_BT. At the same time, we try to make the Monte Carlo trial steps as large as is possible without having a very low acceptance. A displacement that would, on average, give rise to an increase of the potential energy by k_BT would still have a reasonable acceptance. In the case of a single-particle trial move, we

[7] Still, we should stress that it is not worthwhile to spend a lot of time developing a fancy computational scheme that will be only marginally better than existing, simpler schemes, unless your program will run very often and speed is crucial.

then have

$$\begin{aligned}\langle \Delta \mathcal{U} \rangle &= \left\langle \frac{\partial \mathcal{U}}{\partial r_i^\alpha} \right\rangle \overline{\Delta r_i^\alpha} + \frac{1}{2} \left\langle \frac{\partial^2 \mathcal{U}}{\partial r_i^\alpha \partial r_i^\beta} \right\rangle \overline{\Delta r_i^\alpha \Delta r_i^\beta} + \cdots \\ &= 0 + f(\mathcal{U})\, \overline{\Delta r_i^2} + \mathcal{O}(\Delta^4), \end{aligned} \tag{3.3.2}$$

where the angle brackets denote averaging over the ensemble and the horizontal bar denotes averaging over random trial moves. The second derivative of $\mathcal{U}$ has been absorbed into the function $f(\mathcal{U})$, the precise form of which does not concern us here. If we now equate $\langle \Delta \mathcal{U} \rangle$ on the left-hand side of equation (3.3.2) to $k_B T$, we find the following expression for $\overline{\Delta r_i^2}$:

$$\overline{\Delta r_i^2} \approx k_B T / f(\mathcal{U}). \tag{3.3.3}$$

If we attempt to move N particles, one at a time, most of the computation involved is spent on the evaluation of the change in potential energy. Assuming that we use a neighbor list or a similar time-saving device (see Appendix F), the total time spent on evaluating the potential energy change is proportional to nN, where n is the average number of interaction partners per molecule. The sum of the mean-squared displacements will be proportional to $N\overline{\Delta r^2} \sim N k_B T / f(\mathcal{U})$. Hence, the mean-squared displacement per unit of CPU time will be proportional to $k_B T / (n\, f(\mathcal{U}))$. Now suppose that we try to move all particles at once. The cost in CPU time will still be proportional to nN. But, using the same reasoning as in equations (3.3.2) and (3.3.3), we estimate that the sum of the mean-squared displacements is smaller by a factor $1/N$. Hence the total efficiency will be down by this same factor. This simple argument explains why most simulators use single-particle, rather than collective trial moves. It is important to note that we have assumed that a collective MC trial move consists of N independent trial displacements of the particles. As will be discussed in section 14.2, efficient collective MC moves *can* be constructed if the trial displacements of the individual particles are not chosen independently.

Next, consider the choice of the parameter Δ which determines the size of the trial move. How large should Δ be? If it is very large, it is likely that the resulting configuration will have a high energy and the trial move will probably be rejected. If it is very small, the change in potential energy is probably small and most moves will be accepted. In the literature, one often finds the mysterious statement that an acceptance of approximately 50% should be optimal. This statement is not necessarily true. The optimum acceptance ratio is the one that leads to the most efficient sampling of configuration space. If we express efficiency as mean-squared displacement per CPU time, it is easy to see that different Monte Carlo codes will have different optimal acceptance ratios. The reason is that it makes a crucial difference whether the amount of computing required to test whether a trial

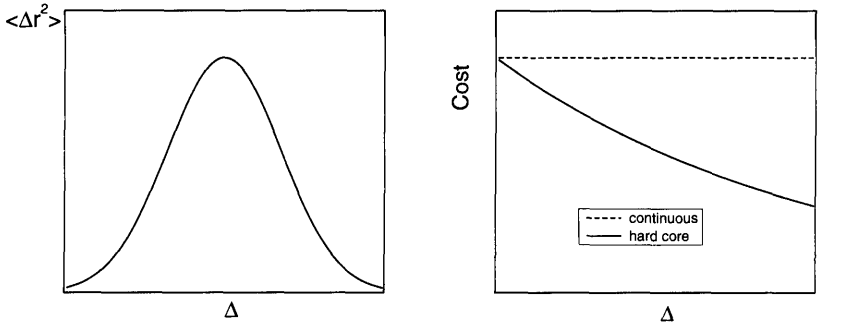

Figure 3.4: (left) Typical dependence of the mean-squared displacement of a particle on the average size Δ of the trial move. (right) Typical dependence of the computational cost of a trial move on the step-size Δ. For continuous potentials, the cost is constant, while for hard-core potentials it decreases rapidly with the size of the trial move.

move is accepted depends on the magnitude of the move (see Figure 3.4). In the conventional Metropolis scheme, all continuous interactions have to be computed before a move can be accepted or rejected. Hence, for continuous potentials, the amount of computation does not depend on the size of a trial move. In contrast, for simulations of molecules with hard repulsive cores, a move can be rejected as soon as overlap with any neighbor is detected. In that case, a rejected move is cheaper than an accepted one, and hence the average computing time per trial move goes down as the step size is increased. As a result, the optimal acceptance ratio for hard-core systems is appreciably lower than for systems with continuous interactions. Exactly how much depends on the nature of the program, in particular on whether it is a scalar or a vector code (in the latter case, hard-core systems are treated much like continuous systems), on how the information about neighbor lists is stored, and even on the computational "cost" of random numbers and exponentiation. The consensus seems to be that for hard-core systems the optimum acceptance ratio is closer to 20 than to 50%, but this is just another rule of thumb that should be checked.[8]

A distinct disadvantage of the efficiency criterion discussed previously is that it does not allow us to detect whether the sampling of configuration space is ergodic. To take a specific example, suppose that our system consists of a number of particles that are trapped in different potential energy min-

[8]In section 14.3.1, we show how, even in the case of continuous potentials, it is possible to reject trial moves before all interactions have been evaluated. With such a sampling scheme, the distinction between the sampling of hard-core and continuous potentials all but disappears.

ima. Clearly, we can sample the vicinity of these minima quite well and still have totally inadequate sampling of the whole of the configuration space. A criterion that would detect such nonergodicity has been proposed by Mountain and Thirumalai [56]. These authors consider the difference between the variance of the time average of the (potential) energy of all particles. Let us denote the time average of the energy of particle j in time interval t by $e_j(t)$:

$$e_j(t) = \frac{1}{t}\int_0^t dt'\, e_j(t').$$

And the average single-particle energy for this interval is

$$\overline{e}(t) \equiv \frac{1}{N}\sum_{j=1}^{N} e_j(t).$$

The variance of interest is

$$\sigma_E^2(t) \equiv \frac{1}{N}\sum_{j=1}^{N} \left[e_j(t) - \overline{e}(t)\right]^2.$$

If all particles sample the whole of configuration space, $\sigma_E^2(t)$ will approach zero as $t \to \infty$:

$$\sigma_E^2(t)/\sigma_E^2(0) \to \tau_E/t,$$

where τ_E is a measure for the characteristic time to obtain uncorrelated samples. However, if the system is nonergodic, as in a (spin) glass, σ_E will not decay to 0. The work of Mountain and Thirumalai suggests that a good method for optimizing the efficiency of a Monte Carlo scheme is to minimize the product of τ_E and the computer time per trial move. Using this scheme, Mountain and Thirumalai concluded that, even for the Lennard-Jones system, a trial move acceptance of 50% is far from optimal. They found that an acceptance probability of 20% was twice as efficient.

Of course, a scheme based on the energy fluctuations of a particle is not very useful to monitor the rate of convergence of simulations of hard-core systems. But the essence of the method is not that one measures the energy but any quantity that is sensitive to the local environment of a particle. For instance, a robust criterion would look at the convergence of the time-averaged Voronoi signature of a particle. Different environments yield different signatures. Only if every particle samples all environments will the variance of Voronoi signatures decay to 0.

Of course, in some situations an efficiency criterion based on ergodicity is not useful. By construction, it cannot be used to optimize simulations of glasses. But also when studying interfaces (e.g., solid-liquid or liquid-vapor) the ergodicity criterion would suggest that every particle should have ample

time to explore both coexisting phases. This is clearly unnecessary: ice can be in equilibrium with water, even though the time of equilibration is far too short to allow complete exchange of the molecules in the two phases.

3.3.2 Orientational Moves

If we are simulating molecules rather than atoms we must also generate trial moves that change the molecular orientation. As we discussed already, it almost requires an effort for generating translational trial moves with a distribution that does not satisfy the symmetry requirement of the underlying Markov chain. For rotational moves, the situation is very different. It is only too easy to introduce a systematic bias in the orientational distribution function of the molecules by using a nonsymmetrical orientational sampling scheme. Several different strategies to generate rotational displacements are discussed in [19]. Here we only mention one possible approach.

Rigid, Linear Molecules

Consider a system consisting of N linear molecules. We specify the orientation of the ith molecule by a unit vector $\hat{\mathbf{u}}_i$. One possible procedure to change $\hat{\mathbf{u}}_i$ by a small, random amount is the following. First, we generate a unit vector $\hat{\mathbf{v}}$ with a random orientation. This is quite easy to achieve (see Algorithm 42). Next we multiply this random unit vector $\hat{\mathbf{v}}$ by a scale factor γ. The magnitude of γ determines the magnitude of the trial rotation. We now add $\gamma\hat{\mathbf{v}}$ to $\hat{\mathbf{u}}_i$. Let us denote the resulting sum vector by $\mathbf{t}$: $\mathbf{t} = \gamma\hat{\mathbf{v}} + \hat{\mathbf{u}}_i$. Note that $\mathbf{t}$ is not a unit vector. Finally, we normalize $\mathbf{t}$, and the result is our trial orientation vector $\hat{\mathbf{u}}_i'$. We still have to fix γ, which determines the acceptance probability for the orientational trial move. The optimum value of γ is determined by essentially the same criteria as for translational moves. We have not yet indicated whether the translational and orientational trial moves should be performed simultaneously. Both procedures are acceptable. However, if rotation and translation correspond to separate moves, then the selection of the type of move should be probabilistic rather than deterministic.

Rigid, Nonlinear Molecules

Only slightly more complex is the case of a nonlinear, rigid molecule. It is conventional to describe the orientation of nonlinear molecules in terms of the Eulerian angles (ϕ, θ, ψ). However, for most simulations, use of these angles is less convenient because all rotation operations should then be expressed in terms of trigonometric functions, and these are computationally expensive. It is usually better to express the orientation of such a molecule

in terms of quaternion parameters (for a discussion of quaternions in the context of computer simulation, see [19]). The rotation of a rigid body can be specified by a quaternion of unit norm $\mathbf{Q}$. Such a quaternion may be thought of as a unit vector in four-dimensional space:

$$\mathbf{Q} \equiv (q_0, q_1, q_2, q_3) \quad \text{with } q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1. \tag{3.3.4}$$

There is a one-to-one correspondence between the quaternion components q_α and the Eulerian angles,

$$\begin{aligned} q_0 &= \cos\frac{\theta}{2}\cos\left(\frac{\phi+\psi}{2}\right) \\ q_1 &= \sin\frac{\theta}{2}\cos\left(\frac{\phi-\psi}{2}\right) \\ q_2 &= \sin\frac{\theta}{2}\sin\left(\frac{\phi-\psi}{2}\right) \\ q_3 &= \cos\frac{\theta}{2}\sin\left(\frac{\phi+\psi}{2}\right), \end{aligned} \tag{3.3.5}$$

and the rotation matrix R, which describes the rotation of the molecule-fixed vector in the laboratory frame, is given by (see, e.g., [57])

$$\mathsf{R} = \begin{pmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{pmatrix}. \tag{3.3.6}$$

To generate trial rotations of nonlinear, rigid bodies, we must rotate the vector (q_0, q_1, q_2, q_3) on the four-dimensional (4D) unit sphere. The procedure just described for the rotation of a 3D unit vector is easily generalized to 4D. An efficient method for generating random vectors uniformly on the 4D unit sphere has been suggested by Vesely [57].

Nonrigid Molecules

If the molecules under consideration are not rigid then we must also consider Monte Carlo trial moves that change the internal degrees of freedom of a molecule. In practice, it makes an important difference whether we have frozen out some of the internal degrees of freedom of a molecule by imposing rigid constraints on, say, bond lengths and possibly even some bond angles. If not, the situation is relatively simple: we can carry out normal trial moves on the Cartesian coordinates of the individual atoms in the molecule (in addition to center-of-mass moves). If some of the atoms are strongly bound, it is advisable to carry out small trial moves on those particles (no rule forbids the use of trial moves of different size for different

atoms, as long as the moves for one particular atom are always sampled from the same distribution).

However, when the bonds between different atoms become very stiff, this procedure does not sample conformational changes of the molecule efficiently. In Molecular Dynamics simulations it is common practice to replace very stiff intramolecular interactions with rigid constraints (see Chapter 15). For Monte Carlo simulations this is also possible. In fact, elegant techniques have been developed for this purpose [58]. However, the corresponding MD techniques [59] are so much easier to use, in particular for large molecules, that we cannot recommend the use of the Monte Carlo technique for any but the smallest flexible molecules with internal constraints.

To understand why Monte Carlo simulations of flexible molecules with a number of stiff (or even rigid) bonds (or bond angles) can become complicated, let us return to the original expression (3.1.2) for a thermal average of a function $A(\mathbf{r}^N)$:

$$\langle A \rangle = \frac{\int d\mathbf{p}^N d\mathbf{r}^N \, A(\mathbf{r}^N) \exp[-\beta \mathcal{H}(\mathbf{p}^N, \mathbf{r}^N)]}{\int d\mathbf{p}^N d\mathbf{r}^N \, \exp[-\beta \mathcal{H}(\mathbf{p}^N, \mathbf{r}^N)]}.$$

If we are dealing with flexible molecules, it is convenient to perform Monte Carlo sampling not on the Cartesian coordinates $\mathbf{r}^N$ but on the generalized coordinates $\mathbf{q}^N$, where q may be, for instance, a bond length or an internal angle. We must now express the Hamiltonian in equation (3.1.2) in terms of these generalized coordinates and their conjugate momenta. This is done most conveniently by first considering the Lagrangian $\mathcal{L} = \mathcal{K} - \mathcal{U}$, where $\mathcal{K}$ is the kinetic energy of the system ($\mathcal{K} = \sum(1/2)m\dot{r}^2$) and $\mathcal{U}$ the potential energy. When we transform from Cartesian coordinates $\mathbf{r}$ to generalized coordinates $\mathbf{q}$, $\mathcal{L}$ changes to

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^{N} \frac{1}{2} m_i \frac{\partial \mathbf{r}_i}{\partial q_\alpha} \frac{\partial \mathbf{r}_i}{\partial q_\beta} \dot{q}_\alpha \dot{q}_\beta - \mathcal{U}(\mathbf{q}^N) \\ &\equiv \frac{1}{2} \dot{\mathbf{q}} \cdot \mathbf{G} \cdot \dot{\mathbf{q}} - \mathcal{U}(\mathbf{q}^N). \end{aligned} \tag{3.3.7}$$

In the second line of equation (3.3.7) we have defined the matrix $\mathbf{G}$. The momenta conjugate to $\mathbf{q}^N$ are easily derived using

$$p^\alpha \equiv \frac{\partial \mathcal{L}}{\partial \dot{q}_\alpha}.$$

This yields $p^\alpha = G_{\alpha\beta}\dot{q}_\beta$. We can now write down the Hamiltonian $\mathcal{H}$ in terms of the generalized coordinates and conjugate momenta:

$$\mathcal{H}(\mathbf{p}, \mathbf{q}) = \frac{1}{2} \mathbf{p} \cdot \mathbf{G}^{-1} \cdot \mathbf{p} + \mathcal{U}(\mathbf{q}^N). \tag{3.3.8}$$

If we now insert this form of the Hamiltonian into equation (3.1.2), and carry out the (Gaussian) integration over the momenta, we find that

$$\begin{aligned}\langle A \rangle &= \frac{\int d\mathbf{q}^N \exp[-\beta \mathcal{U}(\mathbf{q}^N)] A(\mathbf{q}^N) \int d\mathbf{p}^N \exp(-\beta \mathbf{p} \cdot \mathsf{G}^{-1} \cdot \mathbf{p}/2)}{\int d\mathbf{q}^N d\mathbf{p}^N \exp(-\beta \mathcal{H})} \\ &= \frac{\int d\mathbf{q}^N \exp[-\beta \mathcal{U}(\mathbf{q}^N)] A(\mathbf{q}^N) |\mathsf{G}|^{\frac{1}{2}}}{\int d\mathbf{q}^N d\mathbf{p}^N \exp(-\beta \mathcal{H})}. \end{aligned} \tag{3.3.9}$$

The problem with equation (3.3.9) is the term $|\mathsf{G}|^{\frac{1}{2}}$. Although the determinant $|\mathsf{G}|$ can be computed fairly easily for small flexible molecules, its evaluation can become quite an unpleasant task in the case of larger molecules.

Thus far we have considered the effect of introducing generalized coordinates only on the form of the expression for thermal averages. If we are considering a situation where some of the generalized coordinates are actually constrained to have a fixed value, then the picture changes again, because such hard constraints are imposed at the level of the Lagrangian equations of motion. Hard constraints therefore lead to a different form for the Hamiltonian in equation (3.3.8) and to another determinant in equation (3.3.9). Again, all this can be taken into account in the Monte Carlo sampling (see [58]). An example of such a Monte Carlo scheme is the concerted rotation algorithm that has been developed by Theodorou and co-workers [60] to simulate polymer melts and glasses (see section 13.4.4). The idea of this algorithm is to select a set of adjacent skeletal bonds in a chain (up to seven bonds). These bonds are given a collective rotation while the rest of the chain is unaffected. By comparison, Molecular Dynamics simulations of flexible molecules with hard constraints have the advantage that these constraints enter directly into the equations of motion (see [59]). The distinction between Molecular Dynamics and Monte Carlo, however, is more apparent than real, since it is possible to use MD techniques to generate collective Monte Carlo moves (see section 14.2). In Chapter 13, we shall discuss other Monte Carlo sampling schemes that are particularly suited for flexible molecules.

3.4 Applications

In this section we give several case studies using the basic NVT Monte Carlo algorithm.

Case Study 1 (Equation of State of the Lennard-Jones Fluid)
One of the more important applications of molecular simulation is to compute the phase diagram of a given model system. In fact, in Chapter 8 several numerical techniques that have been developed specifically to study

phase transitions will be discussed. It may not be immediately obvious to the reader, however, that there is any need for the sophisticated numerical schemes presented in Chapter 8. In this Case Study, we illustrate some of the problems that occur when we use standard Monte Carlo simulation to determine a phase diagram. As an example, we focus on the vapor-liquid curve of the Lennard-Jones fluid. Of course, as was already mentioned in section 3.2.2, the phase behavior is quite sensitive to the detailed form of the intermolecular potential that is used. In this Case Study, we approximate the full Lennard-Jones potential as follows:

$$u(r) = \begin{cases} u^{lj}(r) & r \leq r_c \\ 0 & r > r_c, \end{cases}$$

where the cutoff radius r_c is set to half the box length. The contribution of the particles beyond this cutoff is estimated with the usual tail corrections; that is, for the energy

$$u^{tail} = \frac{8}{3}\pi\rho\left[\frac{1}{3}\left(\frac{1}{r_c}\right)^9 - \left(\frac{1}{r_c}\right)^3\right]$$

and for the pressure

$$p^{tail} = \frac{16}{3}\pi\rho^2\left[\frac{2}{3}\left(\frac{1}{r_c}\right)^9 - \left(\frac{1}{r_c}\right)^3\right].$$

The equation of state of the Lennard-Jones fluid has been investigated by many groups using Molecular Dynamics or Monte Carlo simulations starting with the work of Wood and Parker [49]. A systematic study of the equation of state of the Lennard-Jones fluid was reported by Verlet [13]. Subsequently, many more studies have been published. In 1979, the data available at that time were compiled by Nicolas *et al.* [61] into an accurate equation of state. This equation has been refitted by Johnson *et al.* [62] in the light of more recent data. In the present study we compare our numerical results with the equation of state by Johnson *et al.*

We performed several simulations using Algorithms 1 and 2. During the simulations we determined the energy per particle and the pressure. The pressure was calculated using the virial

$$P = \frac{\rho}{\beta} + \frac{vir}{V}, \tag{3.4.1}$$

where the virial is defined by

$$vir = \frac{1}{3}\sum_i \sum_{j>i} \mathbf{f}(\mathbf{r}_{ij}) \cdot \mathbf{r}_{ij}, \tag{3.4.2}$$

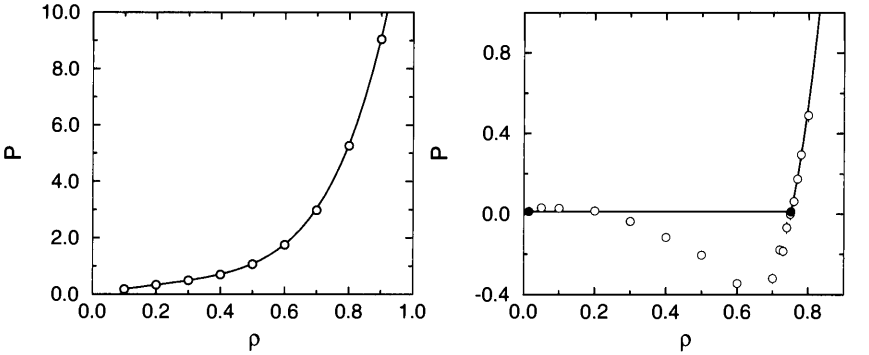

Figure 3.5: Equation of state of the Lennard-Jones fluid. (left) Isotherm at T = 2.0. (right) Isotherm below the critical temperature (T = 0.9); the horizontal line is the saturated vapor pressure and the filled circles indicate the densities of the coexisting vapor and liquid phases. The solid curve represents the equation of state of Johnson *et al.* [62] and the circles are the results of the simulations (N = 500). The errors are smaller than the symbol size.

where $\mathbf{f}(\mathbf{r}_{ij})$ is the intermolecular force. Figure 3.5 (left) compares the pressure as obtained from a simulation above the critical temperature with the equation of state of Johnson *et al.* [62]. The agreement is excellent (as is to be expected).

Figure 3.5 (right) shows a typical isotherm below the critical temperature. If we cool the system below the critical temperature, we should expect to observe vapor-liquid coexistence. However, conventional Monte Carlo or Molecular Dynamics simulations of small model systems are not suited to study the coexistence between two phases. Using the Johnson equation of state, we predict how the pressure of a macroscopic Lennard-Jones system would behave in the two-phase region (see Figure 3.5). For densities inside the coexistence region the pressure is expected to be constant and equal to the saturated vapor pressure. If we now perform a Monte Carlo simulation of a finite system (500 LJ particles), we find that the computed pressure is not at all constant in the coexistence region (see Figure 3.5). In fact we observe that, over a wide density range, the simulated system is metastable and may even have a negative pressure. The reason is that, in a finite system, a relatively important free-energy cost is associated with the creation of a liquid-vapor interface. So much so that, for sufficiently small systems, it is favorable for the system not to phase separate at all [63]. Clearly these problems will be most severe for small systems and in cases where the interfacial free energy is large. For this reason, standard NVT-simulations are not recommended to determine the vapor-liquid coexistence curve or, for that matter, any strong first-order phase transition.

To determine the liquid-vapor coexistence curve we should determine the equation of state for a large number of state points outside the coexistence region. These data can then be fitted to an analytical equation of state. With this equation of state we can determine the vapor-liquid curve (this is exactly the procedure used by Nicolas *et al.* [61] and Johnson *et al.* [62]).

Of course, if we simulate a system consisting of a very large number of particles, it is possible to simulate a liquid phase in coexistence with its vapor. However, such simulations are quite time consuming, because it takes a long time to equilibrate a two-phase system.

Case Study 2 (Importance of Detailed Balance)
For a Monte Carlo simulation to sample points in configuration space according to their correct Boltzmann weight, it is sufficient, but not necessary, to impose the detailed-balance condition on the sampling algorithm. Of course, as the condition of detailed balance is stronger than strictly necessary, it is not excluded that correct sampling schemes exist that violate detailed balance. However, unless one can actually prove that a non-detailed-balance scheme yields the correct distribution, the use of such schemes is strongly to be discouraged. Even seemingly reasonable schemes may give rise to serious, systematic errors.

Here we give an example of such a scheme. Consider an ordinary N,V,T move; a new position is generated by giving a randomly selected particle, say i, a random displacement:

$$x_n(i) = x_o(i) + \Delta_x(\mathrm{Ranf} - 0.5),$$

where Δ_x is twice the maximum displacement. We now make a small error and generate a new position using

$$x_n(i) = x_o(i) + \Delta_x(\mathrm{Ranf} - 0.0) \qquad \text{wrong!}$$

We give the particles only a *positive* displacement. With such a move detailed balance is violated, since the reverse move — putting the particle back at x_o — is not possible.

For the Lennard-Jones fluid we can use the program of Case Study 1 to compare the two sampling schemes. The results of these simulations are shown in Figure 3.6. Note that, at first sight, the results of the incorrect scheme look reasonable; in fact, at low densities the results of the two schemes do not show significant differences. But at high densities the wrong scheme overestimates the pressure. It is important to note that the incorrect scheme leads to a systematic error that does not disappear when we perform longer simulations.

This example illustrates that one can generate numerical results that *look* reasonable, even with an incorrect sampling scheme. For this reason, it is

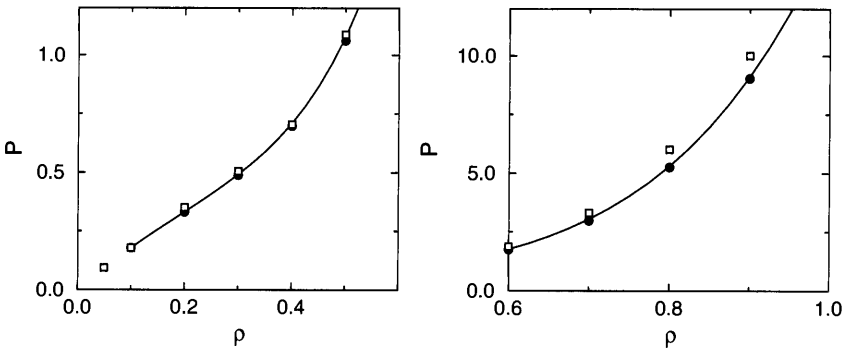

Figure 3.6: Equation of state of the Lennard-Jones fluid (T = 2.0); comparison of a displacement scheme that obeys detailed balance (circles) and one that does not (squares). Both simulations have been performed with 500 particles. The solid curve is the equation of state of Johnson *et al.* [62]. The figure at the left corresponds to the low-pressure regime. The high-pressure regime is shown in the right-hand figure.

important always to compare the results obtained with a new Monte Carlo program with known numerical results or, better still, with exact results that may be known in some limiting case (dilute vapor, dense solid, etc.).

In the present example, the error due to the neglect of detailed balance is quite obvious. In many cases, the effects are less clear. The most common source of non-detailed-balance sampling schemes is the following: in many programs, we can choose from a repertoire of trial moves (e.g., translation, rotation, volume changes). It is recommended that these trial moves are not carried out in fixed order, because then the reverse sequence is impossible and detailed balance is no longer satisfied.[9]

In practice one often does not know *a priori* the optimal maximum displacement in a Monte Carlo simulation. A practical solution is to adjust during the simulation the maximum displacement in such a way that the optimum acceptance probability is obtained. The ideal situation is to determine this optimum during the equilibration. However, if one would keep adjusting the maximum step-size during a production run, then one would violate detailed balance [65]. For example, if from one move to the next, the maximum displacement is decreased, then the *a priori* probability for a particle to return to its previous position could be zero. Hence, if one would change the maximum displacement after every Monte Carlo step serious errors are to be expected. Of course, if one changes the maximum displacement only a few

[9]It has been shown [64] that in this case the detailed-balance condition is indeed sufficient but not necessary to maintain equilibrium.

times during the simulation, then the error will be negligible. Yet, it is better to stay on the safe side and never change the maximum displacement during the projection run.

Case Study 3 (Why Count the Old Configuration Again?)
A somewhat counterintuitive feature of the Metropolis sampling scheme is that, if a trial move is rejected, we should once again count the contributions of the old configuration to the average that we are computing (see acceptance rule (3.1.18)). The aim of this Case Study is to show that this recounting is really essential. In the Metropolis scheme the acceptance rule for a move from o to n is

$$\begin{aligned} \mathrm{acc}(o \to n) &= \exp\{-\beta[\mathcal{U}(n) - \mathcal{U}(o)]\} && \mathcal{U}(n) \geq \mathcal{U}(o) \\ &= 1 && \mathcal{U}(n) < \mathcal{U}(o). \end{aligned}$$

These acceptance rules lead to a transition probability

$$\begin{aligned} \pi(o \to n) &= \exp\{-\beta[\mathcal{U}(n) - \mathcal{U}(o)]\} && \mathcal{U}(n) \geq \mathcal{U}(o) \\ &= 1 && \mathcal{U}(n) < \mathcal{U}(o). \end{aligned}$$

Note that this transition probability must be normalized:

$$\sum_n \pi(o \to n) = 1.$$

From this normalization it follows that the probability that we accept the old configuration again is by definition

$$\pi(o \to o) = 1 - \sum_{n \neq o} \pi(o \to n).$$

This last equation implies that we should count the contribution of the old configuration again.

It is instructive to use the Lennard-Jones program from Case Study 1 to investigate numerically the error that is made when we only include accepted configurations in our averaging. In essence, this means that in Algorithm 2 we continue attempting to displace the selected particle until a trial move has been accepted.[10] In Figure 3.7 we compare the results of the correct scheme with those obtained by the scheme in which we continue to displace a particle until a move is accepted. Again the results look reasonable, but the figure shows that large, systematic errors are being made.

[10]It is easy to see that this approach leads to the wrong answer if we try to compute the average energy of a two-level system with energy levels E_0 and E_1. If we include only accepted trial moves in our averaging, we would find that $\langle E \rangle = (E_0 + E_1)/2$, independent of temperature.

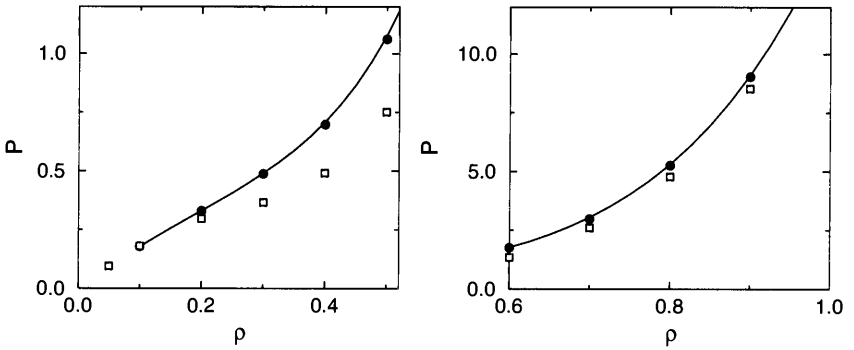

Figure 3.7: Equation of state of the Lennard-Jones fluid ($T = 2.0$); comparison of a scheme in which particles are displaced until a move is accepted (squares) with the conventional scheme (circles). Both simulations have been performed with 108 particles. The solid curve is the equation of state of Johnson *et al.* [62]. The left figure is at low pressure and the right one at high pressure.

One of the important disadvantages of the Monte Carlo scheme is that it does not reproduce the natural dynamics of the particles in the system. However, sometimes this limitation of the method can be made to work to our advantage. In Example 1 we show how the equilibration of a Monte Carlo simulation can be speeded up by many orders of magnitude through the use of unphysical trial moves.

Example 1 (Mixture of Hard Disks)
In a Molecular Dynamics simulation of, for instance, a binary $(A - B)$ mixture of hard disks (see Figure 3.8), the efficiency with which configuration space is sampled is greatly reduced by the fact that concentration fluctuations decay very slowly (typically the relaxation time $\tau \sim D_{AB}/\lambda^2$, where D_{AB} is the mutual diffusion coefficient and λ is the wavelength of the concentration fluctuation). This implies that very long runs are needed to ensure equilibration of the local composition of the mixture. In solids, equilibration may not take place at all (even on time scales accessible in nature). In contrast, in a Monte Carlo simulation, it is permissible to carry out trial moves that swap the identities of two particles of species A and B. Such moves, even if they have only a moderate rate of acceptance (a few percent will do), greatly speed up the sampling of concentration fluctuations.

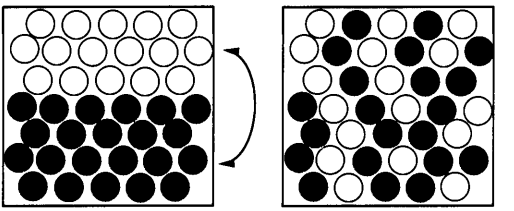

Figure 3.8: A mixture of hard disks, where the identities of two particles are swapped.

3.5 Questions and Exercises

Question 7 (Reduced Units) *Typical sets of Lennard-Jones parameters for argon and krypton are* $\sigma_{Ar} = 3.41$ Å, $\epsilon_{Ar}/k_B = 119.8$ *K and* $\sigma_{Kr} = 3.38$ Å, $\epsilon_{Kr}/k_B = 164.0$ *K [19].*

1. *At the reduced temperature* $T^* = 2.0$, *what is the temperature in kelvin of argon and krypton?*
2. *A typical time step for MD is* $\Delta t^* = 0.001$. *What is this in SI units for argon and krypton?*
3. *If we simulate argon at* $T = 278$ *K and density* $\rho = 2000$ *kg/m*^{3} *with a Lennard-Jones potential, for which conditions of krypton can we use the same data? If we assume ideal gas behavior, compute the pressure in reduced and normal units.*
4. *List the main reasons to use reduced units.*

Question 8 (Heat Capacity) *Heat capacity can also be calculated from fluctuations in the total energy in the canonical ensemble:*

$$C_v = \frac{\langle U^2 \rangle - \langle U \rangle^2}{k_B T^2}. \tag{3.5.1}$$

1. *Derive this equation.*
2. *In a MC NVT simulation, one does not calculate fluctuations in the total energy but in the potential energy. Is it then still possible to calculate the heat capacity? Explain.*
3. *Heat capacity can be also calculated from differentiating the total energy of a system with respect to temperature. Discuss the advantages or disadvantages of this approach.*

Question 9 (A New Potential) *On the planet Krypton, the pair potential between two Gaia atoms is given by the Lennard-Jones* 10-5 *potential*

$$U(r) = 5\epsilon \left[\left(\frac{\sigma}{r}\right)^{10} - \left(\frac{\sigma}{r}\right)^{5} \right].$$

Kryptonians are notoriously lazy and it is therefore up to you to derive the tail corrections for the energy, pressure, and chemical potential. If we use this potential in an MD simulation in the truncated and shifted form we still have a discontinuity in the force. Why? If you compare this potential with the Lennard-Jones potential, will there be any difference in efficiency of the simulation? (Hint: there are two effects!)

Exercise 6 (Calculation of π)
Consider a circle of diameter d surrounded by a square of length l ($l \geq d$). Random coordinates are generated within the square. The value of π can be calculated from the fraction of points that fall within the circle.

1. How can π be calculated from the fraction of points that fall in the circle? Remark: the "exact" value of π can be computed numerically using $\pi = 4 \times \arctan(1)$.
2. Complete the small Monte Carlo program to calculate π using this method.
3. How does the accuracy of the result depend on the ratio l/d and the number of generated coordinates? Derive a formula to calculate the relative standard deviation of the estimate of π.
4. Why is this not a very efficient method for computing π accurately?

Exercise 7 (The Photon Gas)
The average occupancy number of state j of the photon gas, $\langle n_j \rangle$, can be calculated analytically; see equation (2.3.5). It is possible to estimate this quantity using a Monte Carlo scheme. In this exercise, we will use the following procedure to calculate $\langle n_j \rangle$:

(i) Start with an arbitrary n_j.

(ii) Decide at random to perform a trial move to increase or decrease n_j by 1.

(iii) Accept the trial move with probability

$$\mathrm{acc}(o \to n) = \min\left(1, \exp\left[-\beta\left(U(n) - U(o)\right)\right]\right).$$

Of course, n_j cannot become negative!

1. Does this scheme obey detailed balance when $n_j = 0$?

2. Is the algorithm still correct when trial moves are performed that change n_j with a random integer from the interval $[-5, 5]$? What happens when only trial moves are performed that change n_j with either -3 or $+3$?
3. Assume that $N = 1$ and $\epsilon_j = \epsilon$. Write a small Monte Carlo program to calculate $\langle n_j \rangle$ as a function of $\beta\epsilon$. Compare your result with the analytical solution.
4. Modify the program in such a way that the averages are not updated when a trail move is rejected. Why does this lead to erroneous results? At which values of β does this error become more pronounced?
5. Modify the program in such a way that the distribution of n_j is calculated as well. Compare this distribution with the analytical expression.

Exercise 8 (Monte Carlo Simulation of a Lennard-Jones System)
In this exercise, we study a 3D Lennard-Jones system. See also Case Study 1.

1. In the code that you can find on the book's website, the pressure of the system is not calculated. Modify the code in such a way that the average pressure can be calculated. You will only have to make some changes in the subroutine *ener.f*.
2. Perform a simulation at $T = 2.0$ and at various densities. Up to what density is the ideal gas law
$$\beta p = \rho \tag{3.5.2}$$
a good approximation?
3. The program produces a sequence of snapshots of the state of the system. Try to visualize these snapshots using, for example, the program *MOLMOL*.
4. For the heat capacity at constant volume one can derive
$$C_v = \frac{\langle U^2 \rangle - \langle U \rangle^2}{k_B T^2}$$
in which U is the total energy of the system. Derive a formula for the dimensionless heat capacity. Modify the program (only in *mc_nvt.f*) in such a way that C_v is calculated.
5. Instead of performing trial moves in which one particle at a time is displaced, one can make trial moves in which all particles are displaced. Compare the maximum displacements of these moves when 50% of all displacements are accepted.
6. Instead of using a uniformly distributed displacement, one can also use a Gaussian displacement. Does this increase the efficiency of the simulation?

Exercise 9 (Scaling as a Monte Carlo Move)
Consider a system in which the energy is a function of one variable (x) only,

$$\exp\left[-\beta U\left(x\right)\right] = \theta\left(x\right)\theta\left(1-x\right),$$

in which $\theta\left(x\right)$ is the Heaviside step function: $\theta\left(x<0\right)=0$ and $\theta\left(x>0\right)=1$. We wish to calculate the distribution of x in the canonical ensemble. We will consider two possible algorithms (we will use $\delta>0$):

(i) Generate a random change in x between $[-\delta,\delta]$. Accept or reject the new x according to its energy.

(ii) Generate a random number ϕ between $[1,1+\delta]$. With a probability of 0.5, invert the value ϕ thus obtained. The new value of x is obtained by multiplying x with ϕ.

1. Derive the correct acceptance/rejection rules for both schemes.
2. Complete the computer code to calculate the probability density of x. The program writes this distribution to *distri.dat*.
3. What happens when the acceptance rule of method (i) is used in the algorithm of method (ii)?