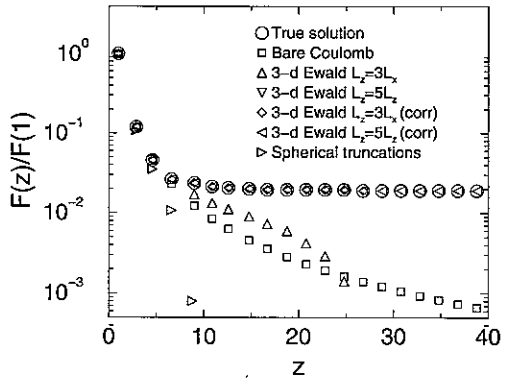

e 12.8: Comparison of various methods for approximating the long- interaction for two charges of the slab geometry shown in Figure 12.6.

jure 12.8 we compare the various approximations with the true two- ısional solution. The bare Coulomb potential and the shifted and trun- Coulomb potential both give a zero force in the limit $z \to \infty$ and ore do not lead to the correct limiting behavior. Although the three- ısional Ewald summation gives a better approximation of the correct on, it still has the incorrect limiting behavior for both a small and a large l slab of vacuum. The corrected three-dimensional Ewald summation, ver, does reproduce the correct solution, for both a slab of vacuum of nd that of $5L_x$.

Chapter 13

Biased Monte Carlo Schemes

Up to this point, we have not addressed a fairly obvious question: what is the point of using the Monte Carlo technique in simulations? After all, Molecular Dynamics simulations can be used to study the static properties of many-body systems and, in addition, MD provides information about their dynamical behavior. Moreover, a standard MD simulation is computationally no more expensive than the corresponding MC simulation. Hence, it would seem tempting to conclude that the MC method is an elegant but outdated scheme:

As the reader may have guessed, we believe that there are good reasons to use MC rather than MD in certain cases. But we stress the phrase *in certain cases*. All other things being equal, MD is clearly the method of choice. Hence, if we use the Monte Carlo technique, we should always be prepared to justify our choice. Of course, the reasons may differ from case to case. Sometimes it is simply a matter of ease of programming: in MC simulations there is no need to compute forces. This is irrelevant if we work with pair potentials, but for many-body potentials, the evaluation of the forces may be nontrivial. Another possible reason is that we are dealing with a system that has no natural dynamics. For instance, this is the case in models with discrete degrees of freedom (e.g., Ising spins). And, indeed, for simulations of lattice models, MC is almost always the technique of choice. But even in off-lattice models with continuous degrees of freedom, it is sometimes better, or even essential, to use Monte Carlo sampling. Usually, the reason to choose the MC technique is that it allows us to perform *unphysical* trial moves, that is, moves that cannot occur in nature (and, therefore, have no counterpart in Molecular Dynamics) but are essential for the equilibration of the system.

This introduction is meant to place our discussion of Monte Carlo techniques for simulating complex fluids in a proper perspective: in most published simulations of complex (often macromolecular) fluids, Molecular Dy-

namics is used, and rightly so. The Monte Carlo techniques that we discuss here have been developed for situations where either MD cannot be used at all or the natural dynamics of the system are too slow to allow the system to equilibrate on the time scale of a simulation.

Examples of such simulations are Gibbs ensemble and grand-canonical Monte Carlo simulations. Both techniques require the exchange of particles, either between a reservoir and the simulation box or between the two boxes. Such particle exchanges are not related to any real dynamics and therefore require the use of Monte Carlo techniques. But, in the case of complex fluids, in particular fluids consisting of chain molecules, the conventional Monte Carlo techniques for grand-canonical or Gibbs ensemble simulations fail. The reason is that, in the case of large molecules, the probability of acceptance of a random trial insertion in the simulation box is extremely small and hence the number of insertion attempts has to be made prohibitively large. For this reason, the conventional grand-canonical and Gibbs ensemble simulations were limited to the study of adsorption and liquid-vapor phase equilibria of small molecules.

13.1 Biased Sampling Techniques

In this chapter,[1] we discuss extensions of the standard Monte Carlo algorithm that allow us to overcome some of these limitations. The main feature of these more sophisticated Monte Carlo trial moves is that they are no longer completely random: the moves are biased in such a way that the molecule to be inserted has an enhanced probability to "fit" into the existing configuration. In contrast, no information about the present configuration of the system is used in the generation of normal (unbiased) MC trial moves: that information is used only to accept or reject the move (see Chapters 3 and 5). Biasing a Monte Carlo trial move means that we are no longer working with a symmetric *a priori* transition matrix. To satisfy detailed balance, we therefore also should change the acceptance rules. This point is discussed in some detail. Clearly, the price we pay for using configurationally biased MC trial moves is a greater complexity of our program. However, the reward is that, with the help of these techniques, we can sometimes speed up a calculation by many orders of magnitude. To illustrate this, we shall discuss examples of simulations that were made possible only through the use of bias sampling.

[1] Readers who are not familiar with the Rosenbluth scheme are advised to read section 11.2 first.

13.1.1 Beyond Metropolis

The general idea of biased sampling is best explained by considering a simple example. Let us assume that we have developed a Monte Carlo scheme that allows us to generate trial configurations with a probability that depends on the potential energy of that configuration:

$$\alpha(o \to n) = f[\mathcal{U}(n)].$$

For the reverse move, we have

$$\alpha(n \to o) = f[\mathcal{U}(o)].$$

Suppose we want to sample the N,V,T ensemble, which implies that we have to generate configurations with a Boltzmann distribution (5.2.2). Imposing detailed balance (see section 5.1) yields, as a condition for the acceptance rule,

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

A possible acceptance rule that obeys this condition is

$$\mathrm{acc}(o \to n) = \min\left(1, \frac{f[\mathcal{U}(o)]}{f[\mathcal{U}(n)]} \exp\{-\beta[\mathcal{U}(n) - \mathcal{U}(o)]\}\right). \qquad (13.1.1)$$

This derivation shows that we can introduce an arbitrary biasing function $f(\mathcal{U})$ in the sampling scheme and generate a Boltzmann distribution of configurations, provided that the acceptance rule is modified in such a way that the bias is removed from the sampling scheme. Ideally, by biasing the probability to generate a trial conformation in the right way, we could make the term on the right-hand side of equation (13.1.1) always equal to unity. In that case, every trial move will be accepted. In Chapter 14.3, we have seen that it is sometimes possible to achieve this ideal situation. However, in general, biased generation of trial moves is simply a technique for enhancing the acceptance of such moves without violating detailed balance.

We now give some examples of the use of non-Metropolis sampling techniques to demonstrate how they can be used to enhance the efficiency of a simulation.

13.1.2 Orientational Bias

To perform a Monte Carlo simulation of molecules with an intermolecular potential that depends strongly on the relative molecular orientation (e.g., polar molecules, hydrogen-bond formers, liquid-crystal forming molecules), it is important to find a position that not only does not overlap with the other molecule but also has an acceptable orientation. If the probability of finding a suitable orientation by chance is very low, we can use biased trial moves to enhance the acceptance.

Algorithm

Let us consider a Monte Carlo trial move in which a randomly selected particle has to be moved and reoriented. We denote the old configuration by o and the trial configuration by n. We use standard random displacement for the translational parts of the move, but we bias the generation of trial orientations, as follows:

1. Move the center of mass of the molecule over a (small) random distance and determine all those interactions that do not depend on the orientations. These interactions are denoted by $u^{pos}(n)$. In practice, there may be several ways to separate the potential into orientation-dependent and orientation-independent parts.
2. Generate k trial orientations $\{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_k\}$ and for each of these trial orientations, calculate the energy $u^{or}(\mathbf{b}_i)$.
3. We define the Rosenbluth[2] factor

$$W(n) = \sum_{j=1}^{k} \exp[-\beta u^{or}(\mathbf{b}_j)]. \tag{13.1.2}$$

   Out of these k orientations, we select one, say, n, with a probability

$$p(\mathbf{b}_n) = \frac{\exp[-\beta u^{or}(\mathbf{b}_n)]}{\sum_{j=1}^{k} \exp[-\beta u^{or}(\mathbf{b}_j)]}. \tag{13.1.3}$$

4. For the old configuration, o, the part of the energy that does not depend on the orientation of the molecules is denoted by $u^{pos}(o)$. The orientation of the molecule in the old position is denoted by $\mathbf{b}_o$, and we generate $k-1$ trial orientations denoted by $\mathbf{b}_2, \cdots, \mathbf{b}_k$. Using these k orientations, we determine

$$W(o) = \exp[-\beta u^{or}(\mathbf{b}_o)] + \sum_{j=2}^{k} \exp[-\beta u^{or}(\mathbf{b}_j)]. \tag{13.1.4}$$

5. The move is accepted with a probability

$$\mathrm{acc}(o \to n) = \min\left(1, \frac{W(n)}{W(o)} \exp\{-\beta[u^{pos}(n) - u^{pos}(o)]\}\right). \tag{13.1.5}$$

It is clear that equation (13.1.3) ensures that energetically favorable configurations are more likely to be generated. An example implementation of this scheme is shown in Algorithm 22. Next, we should demonstrate that the sampling scheme is correct.

[2]Since this algorithm for biasing the orientation of the molecules is very similar to an algorithm developed by Rosenbluth and Rosenbluth in 1955 [295] for sampling configurations of polymers (see section 11.2), we refer to the factor W as the Rosenbluth factor.

Algorithm 22 (Orientational Bias)

```
PROGRAM orien_bias                  move a particle to a random
                                    position using an orient. bias
o=int(ranf()*npart)+1               select a particle at random
xt=ranf()*box                       start: generate new configuration
call ener(xt,en)                    calculate u^pos
wn=exp(-beta*en)
sumw=0
do j=1,k                            generate k trial orientations
  call ranor(b(j))                  random vector on a sphere
  call enero(xt,b(j),eno)           calculate trial orientation j u^or(j)
  w(j)= exp(-beta*eno)              calculate Rosenbluth factor (13.1.2)
  sumw=sumw+w(j)
enddo
call select(w,sum,n)                select one of the orientations
bn=b(n)                             n is the selected conformation
wn=wn*sumw                          Rosenbluth factor new configuration

call ener(x(o),en)                  consider the old conformation
wo=exp(-beta*en)                    calculate u^pos
sumw=0
do j=1,k                            consider k trial orientations
 if (j.eq.1) then
   b(j)=u''(o)                      use actual orientation of particle o
 else
   call ranor(b(j))                 generate a random orientation
 endif
 call enero(x(o),b(j),eno)          calculate energy of trial orientation j
 sumw=sumw+exp(-beta*eno)           calculate Rosenbluth factor (13.1.4)
enddo
wo=wo*sumw                          Rosenbluth factor old configuration
if (ranf().lt.wn/wo)                acceptance test (13.1.5)
+    call accept                    accepted: do bookkeeping
end
```

Comments to this algorithm:

1. *The subroutine* `ener` *calculates the energy associated with the position, the subroutine* `enero` *the energy associated with the orientations.*
2. *The subroutine* `ranor` *generates a random vector on a unit sphere (Algorithm 42), subroutine* `accept` *does the bookkeeping associated with the acceptance of a new configuration, and the subroutine* `select` *selects one of the orientations with probability* $p(i) = w(i)/\sum_j w(j)$ *(see, Algorithm 41).*

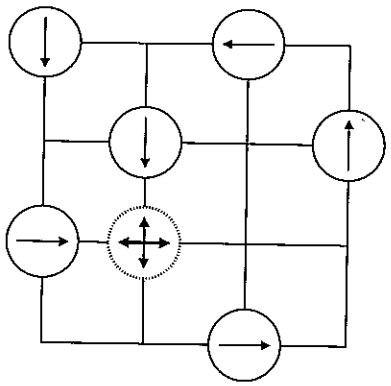

Figure 13.1: Lattice model in which the molecules can take four orientations (indicated by arrows, $k = 4$). The dotted circle indicates the trial position of the particle that we attempt to move.

Justification of Algorithm

To show that the orientational-bias Monte Carlo scheme just described is correct, that is, generates configurations according to the desired distribution, it is convenient to consider lattice models and continuum models separately. For both cases we assume that we work in the canonical ensemble, for which the distribution of configurations is given by equation (5.2.2)

$$\mathcal{N}(\mathbf{q}^N) \propto \exp[-\beta \mathcal{U}(\mathbf{q}^N)],$$

where $\mathcal{U}(\mathbf{q}^N)$ is the sum of orientational and nonorientational part of the energy:

$$\mathcal{U} = u^{or} + u^{pos}.$$

We first consider a lattice model.

Lattice Models

We assume that the molecules in our lattice model can have k discrete orientations (see Figure 13.1). We impose the condition of detailed balance (5.1.1):

$$K(o \to n) = K(n \to o).$$

The flow of configurations o to n is (equation (5.1.2))

$$K(o \to n) = \mathcal{N}(o) \times \alpha(o \to n) \times acc(o \to n). \qquad (13.1.6)$$

In the orientational-bias scheme, the probability of selecting conformation n is (see equation (13.1.3))

$$\alpha(o \to n) = \frac{\exp[-\beta u^{or}(n)]}{W(n)}.$$

Imposing detailed balance and substitution of the desired distribution for $\mathcal{N}(n)$ and $\mathcal{N}(o)$ imposes the following condition on the acceptance rules:

$$\begin{aligned} \frac{acc(o \to n)}{acc(n \to o)} &= \frac{\exp[-\beta \mathcal{U}(n)]}{\exp[-\beta \mathcal{U}(o)]} \times \frac{\exp[-\beta u^{or}(o)]}{W(o)} \times \frac{W(n)}{\exp[-\beta u^{or}(n)]} \\ &= \frac{W(n)}{W(o)} \exp\{-\beta [u^{pos}(n) - u^{pos}(o)]\}. \end{aligned} \qquad (13.1.7)$$

Acceptance rule (13.1.5) satisfies this condition. This demonstrates that for a lattice model detailed balance is fulfilled.

Continuum Model

If the orientation of a molecule is described by a continuous variable, then there is an essential difference with the previous case. In the lattice model all the possible orientations can be considered explicitly, and the corresponding Rosenbluth factor can be calculated exactly. For the continuum case, we can never hope to sample *all* possible orientations. It is impossible to determine the exact Rosenbluth factor since an infinite number of orientations are possible.[3] Hence, the scheme for lattice models, in which the Rosenbluth factor for all orientations is calculated, cannot be used for a continuum model. A possible solution would be to use a large but finite number of trial directions. Surprisingly, this is not necessary. It is possible to devise a *rigorous* algorithm using an *arbitrary subset* of all possible trial directions. The answer we get does *not* depend on the number of trial directions we choose but the statistical accuracy does.

Let us consider the case in which we use a set of k trial orientations; this set is denoted by

$$\{\mathbf{b}\}_k = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_k\}.$$

Conformation $\mathbf{b}_n$ can be selected only if it belongs to the set $\{\mathbf{b}\}_k$. The set of all sets $\{\mathbf{b}\}_k$ that includes conformation n is denoted by

$$\mathcal{B}_n = \{\{\mathbf{b}\}_k | \mathbf{b}_n \in \{\mathbf{b}\}_k\}.$$

Every element of $\mathcal{B}_n$ can be written as $(\mathbf{b}_n, b^*)$, where b^* is the set of $k-1$ additional trial orientations. In the flow of configuration o to n, we have to

[3] In Example 17 we discuss a special case for which the Rosenbluth factor *can* be calculated exactly.

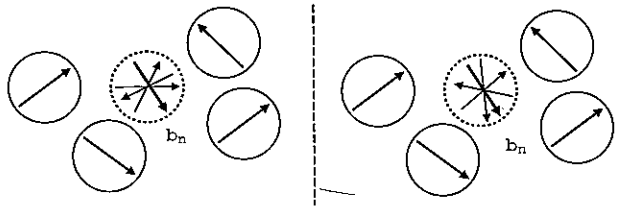

Figure 13.2: Continuum model in which the molecule can have an arbitrary orientation (indicated by arrows). The figure shows two different sets of four trial orientations that both include orientation $\mathbf{b}_n$.

consider the sum over all sets in $\mathcal{B}_n$

$$K(o \to n) = \mathcal{N}(o) \sum_{i \in \mathcal{B}_n} \alpha(o \to n, i) \times \mathrm{acc}(o \to n, i), \tag{13.1.8}$$

in which the probability of generating configuration n and the acceptance depend on the particular set of trial orientations i.

Similarly, for the reverse move, we define the set $\mathcal{B}_o$

$$\mathcal{B}_0 = \{\{\mathbf{b}\}_k | \mathbf{b}_o \in \{\mathbf{b}\}_k\},$$

for which each element can be written as $(\mathbf{b}_o, b'^*)$. The expression for the reverse flow then becomes

$$K(n \to o) = \mathcal{N}(n) \sum_{j \in \mathcal{B}_o} \alpha(n \to o, j) \times \mathrm{acc}(n \to o, j). \tag{13.1.9}$$

It should be stressed that infinitely many different sets of orientations include $\mathbf{b}_n$, and the same holds for sets that include $\mathbf{b}_o$. Moreover, the probability of selecting $\mathbf{b}_n$ from such a set depends on the remainder of the set b^* (see Figure 13.2). Hence, the acceptance probability must also depend on the sets b^* and b'^*.

Detailed balance is certainly obeyed if we impose a much stronger condition, "super-detailed balance," which states that for every particular choice of the sets b^* and b'^*, detailed balance should be obeyed,

$$\begin{aligned} K(o \to n, b^*, b'^*) &= K(n \to o, b'^*, b^*), \\ \mathcal{N}(o)\, \alpha(o \to n, b^*, b'^*)\, \mathrm{acc}(o \to n, b^*, b'^*) &= \mathcal{N}(n)\, \alpha(n \to o, b'^*, b^*)\, \mathrm{acc}(n \to o, b'^*, b^*), \end{aligned} \tag{13.1.10}$$

in which b^* and b'^* are two sets of $k-1$ arbitrary additional trial orientations. It may seem strange that the sets b^* and b'^* show up on *both* sides of the equations. However, bear in mind that, to decide on the acceptance of the forward move, one should generate both the set b^* that includes the new orientation *and* the set b'^* around the old orientation. Hence, the construction of a trial move includes both sets of trial orientations. As the probabilities of generating b^* and b'^* appear on both sides of the equations, they cancel each other. Moreover, the *a priori* probability of generating a random orientation $\mathbf{b}_n$ in the forward move is equal to the *a priori* probability of generating $\mathbf{b}_o$ in the reverse move. So these generation probabilities also cancel each other. This leads to a great simplification of the acceptance criterion. For the canonical ensemble, substitution of equations (13.1.2) and (13.1.3) yields

$$\begin{aligned} \frac{\mathrm{acc}(o \to n, b^*, b'^*)}{\mathrm{acc}(n \to o, b'^*, b^*)} &= \frac{\exp[-\beta \mathcal{U}(n)]}{\exp[-\beta \mathcal{U}(o)]} \frac{\exp[-\beta u^{or}(o)]}{W(\mathbf{b}_o, b'^*)} \frac{W(\mathbf{b}_n, b^*)}{\exp[-\beta u^{or}(n)]} \\ &= \frac{W(\mathbf{b}_n, b^*)}{W(\mathbf{b}_o, b'^*)} \exp\{-\beta [u^{pos}(n) - u^{pos}(o)]\}. \end{aligned} \tag{13.1.11}$$

As acceptance rule (13.1.5) satisfies this condition, detailed balance is indeed obeyed.

Note that, in this demonstration, we did not have to assume that the number of trial orientations k had to be large. In fact, the result is *independent* of the number of trial orientations.

Example 16 (Orientational Bias of Water)
Cracknell *et al.* [353] used an orientational-bias scheme to simulate liquid water. At ambient temperature, water has a relatively open structure, in which the water molecules form a network due to the hydrogen bonds. To insert a water molecule successfully, one has not only to place the molecule in an empty spot but also find a good orientation. The method used by Cracknell *et al.* to find this optimum orientation is similar to the one introduced in this section, in the sense that a bias in the orientation is introduced and is subsequently removed by adjusting the acceptance rules. Yet, the philosophy behind the approach of Cracknell *et al.* is fundamentally different.

In the scheme of Cracknell *et al.*, a random position of a water molecule r is generated and one trial orientation ω is drawn from a distribution $f(r, \omega)$. The problem is that the optimum distribution $f(r, \omega)$ is not known *a priori* and depends on the conformations of the other water molecules. However, as we have shown, any distribution can be used (as long as detailed balance and microscopic reversibility are obeyed). Since the construction of the true orientational distribution requires too much computer time, Cracknell *et al.* constructed a distribution that was meant to mimic the true distribution. To this end, one axis of the water molecule was given a random orientation and, for the other axis a biasing scheme was used. For this axis, n equidistant angles ψ_i were generated

$$\psi_i = 2\pi p/n, \quad p \in \{1, \cdots, n\}.$$

For each of these, the Boltzmann factor of the energy was calculated

$$f_i = C \exp(-\beta u_{\psi_i}).$$

Assuming that the Boltzmann weight varies linearly between test points, these n points span an approximate orientational distribution $f(\psi)$. For instance, for $\psi \in [2\pi p/n, 2\pi(p+1)/n]$, the distribution f is given by

$$f(\psi) = \frac{C}{2\pi/n}\{(2\pi(p+1)/n - \psi)f_p + (\psi - 2\pi p/n)f_{p+1}\}.$$

The constant C was fixed by the requirement that the orientational distribution be normalized. Using a standard rejection scheme, a trial orientation is generated according to the distribution specified by $f(\psi)$. For liquid water under ambient conditions, this method gives an improvement of a factor 2–3 over the conventional random insertion.

The main difference between the scheme of Cracknell *et al.* and the algorithm just discussed is that in Cracknell *et al.*'s scheme an attempt is made to construct a continuous distribution that approaches the true distribution in the limit of large n. In contrast, for the scheme of section 13.1.2, the shape of the true distribution does not matter. In particular, it is not necessary to reconstruct the distribution or to calculate a normalization factor.

Example 17 (Dipoles Embedded in Spherical Atoms)
In systems with dipoles, the energy depends on the mutual orientation of the molecules and a bias in the sampling of the orientation can be useful. For models of dipoles embedded in an otherwise spherical particle (e.g., the dipolar hard-sphere fluid) the scheme of section 13.1.2 can be implemented elegantly as pointed out by Caillol [225]. In equations (13.1.2) and (13.1.4), the Rosenbluth factor is calculated by sampling k trial orientations. For a dipolar hard sphere (or any point dipole), we can calculate the Rosenbluth factors exactly once the electric field ($\mathbf{E}$) at the position of the inserted particle and that at the position of the old configuration are known:

$$\begin{aligned} W(\mathbf{r}) &= \int db \exp[-\beta\mu \cdot \mathbf{E}(\mathbf{r})] \\ &= \frac{\sinh[\beta|\mu|\,|\mathbf{E}(\mathbf{r})|]}{\beta|\mu|\,|\mathbf{E}(\mathbf{r})|}, \end{aligned}$$

where μ is the dipole moment of the molecule.[4] A trial orientation can now be drawn directly from the distribution

$$p(\mathbf{r}, \omega) = \frac{\exp[-\beta\mu \cdot \mathbf{E}(\mathbf{r})]}{W(\mathbf{r})}.$$

[4] In fact, there is a subtlety with this expression. It assumes that the component of the local electric field in the direction of the dipole does not depend on the absolute orientation of the dipole. This seems obvious. But, in the case of an Ewald summation, where the long-range interaction of a molecule with its periodic images is represented by a Fourier sum, this condition is not quite satisfied.

13.2 Chain Molecules

The sampling of equilibrium conformations of polymers is usually time consuming. The main reason is that the natural dynamics of polymers are dominated by topological constraints (for example, chains cannot cross) and hence any algorithm based on the real motion of macromolecules will suffer from the same problem. For this reason, many "unphysical" Monte Carlo trial moves have been proposed to speed up the sampling of polymer conformations (see, e.g., [299]). In this section we introduce the configurational-bias Monte Carlo scheme [293, 297, 354, 355]. This simulation technique can be used for systems where it is not possible to change the conformation of a macromolecule by successive small steps.

13.2.1 Configurational-Bias Monte Carlo

The starting point for the configurational-bias Monte Carlo technique is the scheme introduced by Rosenbluth and Rosenbluth in 1955 [295]. The Rosenbluth scheme itself also was designed as a method to sample polymer conformations.[5] A drawback of the Rosenbluth scheme is, however, that it generates an unrepresentative sample of all polymer conformations; that is, the probability of generating a particular conformation using this scheme is *not* proportional to its Boltzmann weight. Rosenbluth and Rosenbluth corrected for this bias in the sampling of polymer conformations by introducing a conformation-dependent weight factor W. However, as was shown in detail by Batoulis and Kremer [300], this correction procedure, although correct in principle, in practice works only for relatively short chains (see Example 13).

The solution of this problem is to bias the Rosenbluth sampling in such a way that the correct (Boltzmann) distribution of chain conformations is recovered in a Monte Carlo sequence. In the configurational-bias scheme to be discussed next, the Rosenbluth weight is used to bias the *acceptance* of trial conformations generated by the Rosenbluth procedure. As we shall show, this guarantees that all chain conformations are generated *with the correct Boltzmann weight*.

[5] The Rosenbluth scheme is discussed in some detail in the context of a free energy calculation of a chain molecule in Chapter 11.

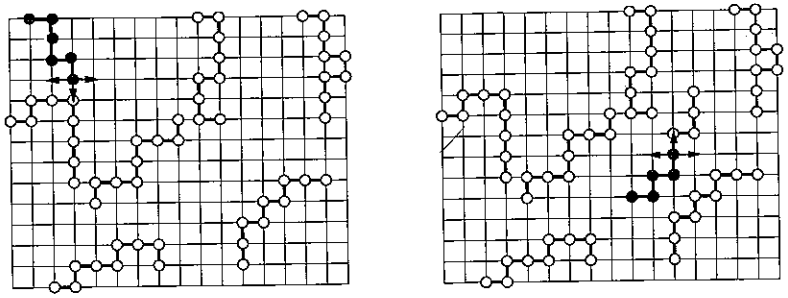

Figure 13.3: Sketch of the configurational-bias Monte Carlo scheme. The left figure shows the generation of a new configuration and the right figure shows the retracing of the old conformation. The arrows indicate the three trial positions.

13.2.2 Lattice Models

Algorithm

The configurational-bias Monte Carlo algorithm consists of the following steps:

1. Generate a trial conformation using the Rosenbluth scheme (see Figure 13.3, left) to grow the entire molecule, or part thereof, and compute its Rosenbluth weight $W(n)$.
2. "Retrace" the old conformation (see Figure 13.3, right) and determine its Rosenbluth factor.
3. Accept the trial move with a probability

$$\mathrm{acc}(o \to n) = \min[1, W(n)/W(o)]. \tag{13.2.1}$$

The generation of a trial conformation n of a polymer consisting of ℓ monomers is generated using an algorithm based on the method of Rosenbluth and Rosenbluth (see Figure 13.3):

1. The first atom is inserted at random, and its energy is denoted by $u_1(n)$, and[6] $w_1(n) = k\exp[-\beta u_1(n)]$, where k is the coordination number of the lattice, for example, $k = 6$ for a simple cubic lattice.
2. For the next segment, with index i, there are k possible trial directions. The energy of trial direction j is denoted by $u_i(j)$. From the k possible directions, we select one, say, n, with a probability

$$p_i(n) = \frac{\exp[-\beta u_i(n)]}{w_i(n)}, \tag{13.2.2}$$

where $w_i(n)$ is defined as

$$w_i(n) = \sum_{j=1}^{k} \exp[-\beta u_i(j)]. \tag{13.2.3}$$

The interaction energy $u_i(j)$ includes all interactions of segment i with other molecules in the system and with segments 1 through $i-1$ of the same molecule. It does not include the interactions with segments $i+1$ to ℓ. Hence, the total energy of the chain is given by $U(n) = \sum_{i=1}^{\ell} u_i(n)$.

3. Step 2 is repeated until the entire chain is grown and we can determine the Rosenbluth factor of configuration n:

$$W(n) = \prod_{i=1}^{\ell} w_i(n). \tag{13.2.4}$$

Similarly, to determine the Rosenbluth factor of the old configuration, o, we use the following steps (see Figure 13.3).

1. One of the chains is selected at random. This chain is denoted by o.
2. We measure the energy of the first monomer $u_1(o)$ and compute $w_1(o) = k\exp[-\beta u_1(o)]$.
3. To compute the Rosenbluth weight for the remainder of the chain, we determine the energy of monomer i at its actual position, and also the energy it would have had had it been placed in any of the other $k-1$ sites neighboring the actual position of monomer $i-1$ (see Figure 13.3). These energies are used to calculate

$$w_i(o) = \exp[-\beta u_i(o)] + \sum_{j=2}^{k} \exp[-\beta u_i(j)].$$

4. Once the entire chain has been retraced, we determine its Rosenbluth factor:

$$W(o) = \prod_{i=1}^{\ell} w_i(o). \tag{13.2.5}$$

[6] The factor k in the definition of the Rosenbluth weight of the first segment, strictly speaking, is unnecessary. We introduce it only here to make the subsequent notation more compact.

Algorithm 23 (Basic Configurational-Bias Monte Carlo)

```
PROGRAM CBMC                    configurational-bias Monte Carlo

new_conf=.false.                first retrace (part of) the old conf.
call grow(new_conf,wo)          to calculate its Rosenbluth factor

new_conf=.true.                 next consider the new configuration
call grow(new_conf,wn)          grow (part of) a chain and calculate
                                the Rosenbluth factor of the new conf.
if (ranf().lt.wn/wo)            acceptance test (13.2.6)
+   call accept                 accept and do bookkeeping
end
```

Comments to this algorithm:

1. *This algorithm shows the basic structure of the configurational-bias Monte Carlo method. The details of the model are considered in the subroutine* `grow` *(see Algorithm 24 for a polymer on a lattice).*
2. *The subroutine* `accept` *takes care of the bookkeeping of the new configuration.*

Finally the trial move from o to n is accepted with a probability given by

$$\mathrm{acc}(o \to n) = \min[1, W(n)/W(o)]. \tag{13.2.6}$$

A schematic example of the implementation of this scheme is given in Algorithms 23 and 24. We now have to demonstrate that the acceptance rule (13.2.6) correctly removes the bias of generating new segments in the chain introduced by using equation (13.2.2).

Justification of the Algorithm

The demonstration that this algorithm samples a *Boltzmann* distribution is similar to the one for the orientational-bias algorithm for lattice models (section 13.1.2).

The probability of generating a particular conformation n follows from the repetitive use of equation (13.2.2):

$$\alpha(o \to n) = \prod_{i=1}^{\ell} \frac{\exp[-\beta u_i(n)]}{w_i(n)} = \frac{\exp[-\beta \mathcal{U}(n)]}{W(n)}. \tag{13.2.7}$$

Algorithm 24 (Growing a Chain on a Lattice)

```
SUBROUTINE grow(new_conf,w)     grow an ℓ bead polymer on a lattice
                                with coordination number k and
                                calculate its Rosenbluth factor w
if (new_conf) then
  xn(1)=ranf()*box              insert the first monomer
else
  o=ranf()*npart+1              select old chain at random
  xn(1)=x(o,1)
endif ,
call ener(xn(1),en)             calculate energy
w=k*exp(-beta*en)               Rosenbluth factor first monomer
do i=2,ell
  sumw=0
  do j=1,k                      consider the k trial directions
    xt(j)=xn(i-1)+b(j)          determine trial position
    call ener(xt(j),en)         determine energy trial position j
    w(j)=exp(-beta*en)
    sumw=sumw+w(j)
  enddo
  if (new_conf) then
    call select(w,sumw,n)       select one of the trial position
    xn(i)=xt(n)                 direction n is selected
  else
    xn(i)=x(o,i)
  endif
  w=w*sumw                      update Rosenbluth factor
enddo
return
end
```

Comments to this algorithm:

1. *If* `new_conf=.true.` *generate a new configuration, if* `new_conf = .false.` *retrace an old one.*
2. *In a lattice model we consider all possible trial positions, denoted by* `b(j)`*, therefore, for the old configuration, the actual position is automatically included.*
3. *The subroutine* `select` *(Algorithm 41) selects one of the trial positions with probability* $p(i) = w(i)/\sum_j w(j)$*. The subroutine* `ener` *calculates the energy of the monomer at the given position with the other polymers and the monomers of the chain that already have been grown.*

Similarly, for the reverse move,

$$\alpha(n \to o) = \frac{\exp[-\beta \mathcal{U}(o)]}{W(o)}. \tag{13.2.8}$$

The requirement of detailed balance (5.1.1) imposes the following condition on the acceptance criterion:

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{W(n)}{W(o)}. \tag{13.2.9}$$

Clearly, the proposed acceptance criterion (13.2.6) satisfies this condition.

It should be stressed that the value of factor $W(o)$ depends on the direction in which the old configuration is retraced: if we start from monomer 1, we find a different numerical value for $W(o)$ than if we start from monomer ℓ. As a consequence the probability of such a move depends on the way the factor $W(o)$ has been calculated. Although such a dependence is at first sight counterintuitive, both ways of retracing the old conformation—starting with monomer 1 or with monomer ℓ—result in the correct distribution of states, as long as both ways occur with equal probability during the simulation. This is automatically satisfied in the case of linear chains of identical segments where the labeling of the terminal groups is completely arbitrary.

13.2.3 Off-lattice Case

Next we consider configurational-bias Monte Carlo for off-lattice systems. As with the orientational moves described in section 13.1.2, some aspects in a continuum version of configurational-bias Monte Carlo require special attention. In section 13.1.2 we already showed that it may be possible to develop a configurational-bias sampling scheme even when it is impossible to calculate the Rosenbluth factor exactly. For chain molecules, we can follow basically the same approach.

The other important point that we have to consider is the way in which trial conformations of a chain molecule are generated. In a lattice model, the number of trial conformations is dictated by the lattice. In an off-lattice system, one could generate trial segments with orientations distributed uniformly on a unit sphere. However, for many models of interest this procedure is not very efficient, in particular when there are strong intramolecular interactions (e.g., bending and torsion potentials). The efficiency of a configurational-bias Monte Carlo algorithm depends to a large extent on the method used of generating the trial orientations. For example, an isotropic distribution of trial directions is well suited for completely flexible chains. In contrast, for a stiff chain (e.g., liquid-crystal forming polymer), such a trial position will almost always be rejected because of the intramolecular interactions.

Algorithm

From the preceding discussion, it follows that the intramolecular interactions should be taken into account in generating the set of trial conformations. Here, we consider the case of a flexible molecule *with* contributions to the internal energy due to bond bending and torsion. The fully flexible case then follows trivially. Consider a chain of ℓ linear segments, the potential energy of a given conformation $\mathcal{U}$ has two contributions:

1. The *bonded potential energy* $\mathcal{U}^{\mathrm{bond}}$ is equal to the sum of the contributions of the individual joints. A joint between segments i and $i+1$ (say) has a potential energy u_i^{bond} that depends on the angle θ between the successive segments. For instance, $u_i^{\mathrm{bond}}(\theta)$ could be of the form $u_i^{\mathrm{bond}}(\theta)=k_\theta(\theta-\theta_0)^2$. For realistic models for polyatomic molecules, u_i^{bond} includes all local bonded potential energy changes due to the bending and torsion of the bond from atom $i-1$ to atom i.
2. The *external potential energy* $\mathcal{U}^{\mathrm{ext}}$ accounts for all interactions with other molecules and for all the nonbonded intramolecular interactions. In addition, interactions with any external field that may be present are also included in $\mathcal{U}^{\mathrm{ext}}$.

In what follows we shall denote a chain in the absence of the external interactions as the *ideal* chain. Note that this is a purely fictitious concept, as real chains always have nonbonded intramolecular interactions.

To perform a configurational-bias Monte Carlo move, we apply the following "recipe" to construct a conformation of a chain of ℓ segments. The construction of chain conformations proceeds segment by segment. Let us consider the addition of one such segment. To be specific, let us assume that we have already grown $i-1$ segments and are trying to add segment i. This is done in two steps. First we generate a trial conformation n, next we consider the old conformation o. A trial conformation is generated as follows:

1. Generate a fixed number (say k) trial segments. The orientations of the trial segments are distributed according to the Boltzmann weight associated with the bonded interactions of monomer i (u_i^{bond}). We denote this set of k different trial segments by

$$\{\mathbf{b}\}_k = \{\mathbf{b}_1, \cdots, \mathbf{b}_k\},$$

where the probability of generating a trial segment $\mathbf{b}$ is given by

$$p_i^{\mathrm{bond}}(\mathbf{b})d\mathbf{b} = \frac{\exp[-\beta u_i^{\mathrm{bond}}(\mathbf{b})]d\mathbf{b}}{\int d\mathbf{b}\exp[-\beta u_i^{\mathrm{bond}}(\mathbf{b})]} = C\exp[-\beta u_i^{\mathrm{bond}}(\mathbf{b})]d\mathbf{b}. \tag{13.2.10}$$

2. For all k trial segments, we compute the external Boltzmann factors $\exp[-\beta u_i^{ext}(\mathbf{b}_i)]$, and out of these, we select one, denoted by n, with a probability

$$p_i^{ext}(\mathbf{b}_n) = \frac{\exp[-\beta u_i^{ext}(\mathbf{b}_n)]}{w_i^{ext}(n)}, \tag{13.2.11}$$

where we have defined

$$w_i^{ext}(n) = \sum_{j=1}^{k} \exp[-\beta u_i^{ext}(\mathbf{b}_j)]. \tag{13.2.12}$$

3. The selected segment n becomes the ith segment of the trial conformation of the chain.
4. When the entire chain is grown, we calculate the Rosenbluth factor of the chain:

$$W^{ext}(n) = \prod_{i=1}^{\ell} w_i^{ext}(n), \tag{13.2.13}$$

where Rosenbluth factor of the first monomer is defined by

$$w_1^{ext}(n) = k \exp[-\beta u_1^{ext}(\mathbf{r}_1)], \tag{13.2.14}$$

where $\mathbf{r}_1$ is the position of the first monomer.

For the old configuration, a similar procedure to calculate its Rosenbluth factor is used.

1. One of the chains is selected at random. This chain is denoted o.
2. The external energy of the first monomer is calculated. This energy involves only the external interactions. The Rosenbluth weight of this first monomer is given by

$$w_1^{ext}(o) = k \exp[-\beta u_1^{ext}(o)]. \tag{13.2.15}$$

3. The Rosenbluth factors of the other $\ell - 1$ segments are calculated as follows. We consider the calculation of the Rosenbluth factor of segment i. We generate a set of $k-1$ orientations with a distribution prescribed by the bonded interactions (13.2.10). These orientations, together with the actual bond between segment $i-1$ and i, form the set of k orientations $(\mathbf{b}_o, \mathbf{b}'^*)$. These orientations are used to calculate the external Rosenbluth factor:

$$w_i^{ext}(o) = \sum_{j=1}^{k} \exp[-\beta u_i^{ext}(\mathbf{b}_j)]. \tag{13.2.16}$$

4. For the entire chain the Rosenbluth factor of the old conformation is defined by

$$W^{ext}(o) = \prod_{i=1}^{\ell} w_i^{ext}(o). \tag{13.2.17}$$

After the new configuration has been generated and the Rosenbluth factor of the old configuration has been calculated, the move is accepted with a probability

$$\mathrm{acc}(o \to n) = \min[1, W^{ext}(n)/W^{ext}(o)]. \tag{13.2.18}$$

We still have to show that this sampling scheme is correct.

Justification of Algorithm

Comparison with the lattice version shows that for the off-lattice case, two aspects are different. First, for a model with continuous degrees of freedom, we cannot calculate the Rosenbluth factor exactly. This point has been discussed in detail in section 13.1.2 for the orientational-bias scheme. As in section 13.1.2, we impose super-detailed balance. Second, the way in which we generate trial conformations is different for off-lattice than for lattice models. In a lattice model there is no need to separate the interactions in bonded and external ones. We have to show that the way in which we treat bonded interactions does not perturb the sampling.

The probability of generating a chain of length ℓ is the product of the probability of generating a trial orientation (13.2.10) and the probability of selecting this orientation (13.2.11); for all monomers this gives, as a probability of generating conformation n,

$$\alpha(o \to n) = \prod_{i=1}^{\ell} p_i(o \to n) = \prod_{i=1}^{\ell} p_i^{bond}(n) p_i^{ext}(n). \tag{13.2.19}$$

In the following, we consider the expressions for one of the ℓ segments, to keep the equations simple. A given set of k trial orientations, which includes orientation n, is denoted by $(\mathbf{b}_n, \mathbf{b}^*)$ (see section 13.1.2). As before, we stress that the generation of the additional trial orientations $(\mathbf{b}'^*)$ around the old segment $(\mathbf{b}_o)$ is an essential part of the *generation* of the trial move. We denote the probability of generating the combined set $\mathbf{b}^*, \mathbf{b}'^*$ by

$$\mathcal{P}^{bond}(\mathbf{b}^*, \mathbf{b}'^*).$$

Hence, the flow of configurations is given by

$$\begin{aligned} K(o \to n, b^*, b'^*) &= \mathcal{N}(o) \times \alpha(o \to n, b^*, b'^*) \times \mathrm{acc}(o \to n, b^*, b'^*) \\ &= \exp[-\beta u(o)] \times C \exp[-\beta u^{\mathrm{bond}}(n)] \times \frac{\exp[-\beta u^{\mathrm{ext}}(n)]}{w^{\mathrm{ext}}(\mathbf{b}_n, b^*)} \\ &\quad \times \mathrm{acc}(o \to n, b^*, b'^*) \mathcal{P}^{\mathrm{bond}}(b^*, b'^*). \end{aligned} \tag{13.2.20}$$

For the reverse move, we have

$$\begin{aligned} K(n \to o, b'^*, b^*) &= \mathcal{N}(n) \times \alpha(n \to o, b'^*, b^*) \times \mathrm{acc}(n \to o, b'^*, b^*) \\ &= \exp[-\beta u(n)] \times C \exp[-\beta u^{\mathrm{bond}}(o)] \times \frac{\exp[-\beta u^{\mathrm{ext}}(o)]}{w^{\mathrm{ext}}(\mathbf{b}_o, b'^*)} \\ &\quad \times \mathrm{acc}(n \to o, b'^*, b^*) \mathcal{P}^{\mathrm{bond}}(b^*, b'^*). \end{aligned} \tag{13.2.21}$$

Recall that the total energy of a monomer is the sum of the bonded and external contributions:

$$u(n) = u^{\mathrm{bond}}(n) + u^{\mathrm{ext}}(n).$$

We now impose super-detailed balance (13.1.10). The factors $\mathcal{P}^{\mathrm{bond}}(b^*, b'^*)$ on both sides of the equation cancel each other, and we get the following simple criterion for the acceptance rule:

$$\frac{\mathrm{acc}(o \to n, b^*, b'^*)}{\mathrm{acc}(n \to o, b'^*, b^*)} = \frac{w^{\mathrm{ext}}(\mathbf{b}_n, b^*)}{w^{\mathrm{ext}}(\mathbf{b}_o, b'^*)}. \tag{13.2.22}$$

This demonstration was only for a single segment in a chain. For the entire chain, the corresponding acceptance criterion is obtained analogously. It is simply the product of the terms for all segments:

$$\frac{\mathrm{acc}[o \to n, (b_1^*, \cdots, b_\ell^*)]}{\mathrm{acc}[n \to o, (b'^*_1, \cdots, b'^*_\ell)]} = \frac{\prod_{i=1}^{\ell} w_i^{\mathrm{ext}}(\mathbf{b}_n, b^*)}{\prod_{i=1}^{\ell} w_i^{\mathrm{ext}}(\mathbf{b}_o, b'^*)} = \frac{W[n, (b^*_1, \cdots, b^*_\ell)]}{W[o, (b'^*_1, \cdots, b'^*_\ell)]}. \tag{13.2.23}$$

And, indeed, our acceptance rule (13.2.18) satisfies this condition. The equation shows that, because the trial orientations are generated with a probability (13.2.10) prescribed by the bonded energy, this energy does *not* appear in the acceptance rules. In Case Study 19, a detailed discussion is given on the advantages of this approach. It is important to note that we do not need to know the normalization constant C of equation (13.2.10).

The basic structure of an algorithm for configurational-bias Monte Carlo for continuum models is very similar to the lattice version (Algorithm 23); the main difference is the way in which configurations are generated.

Case Study 18 (Equation of State of Lennard-Jones Chains)
To illustrate the configurational-bias Monte Carlo technique described in this section, we determine the equation of state of a system consisting of eight-bead chains of Lennard-Jones particles. The nonbonded interactions are described by a truncated and shifted Lennard-Jones potential. The potential is truncated at $R_c = 2.5\sigma$. The bonded interactions are described with a harmonic spring

$$u^{\mathrm{vib}}(l) = \begin{cases} 0.5 k_{\mathrm{vib}}(l-1)^2 & 0.5 \le l \le 1.5 \\ \infty & \text{otherwise} \end{cases},$$

where l is the bond length, the equilibrium bond length has been set to 1, and $k_{\mathrm{vib}} = 400$.

The simulations are performed in cycles. In each cycle, we perform on average N_{dis} attempts to displace a particle, N_{cbmc} attempts to (partly) regrow a chain, and N_{vol} attempts to change the volume (only in the case of N,P,T simulations). If we regrow a chain, the configurational-bias Monte Carlo scheme is used. In this move we select at random the monomer from which we start to regrow. If this happens to be the first monomer, the entire molecule is regrown at a random position. For all the simulations, we used eight trial orientations. The lengths of trial bonds are generated with a probability prescribed by the bond-stretching potential (see Case Study 19).

In Figure 13.4 the equation of state as obtained from N,V,T simulations is compared with one obtained from N,P,T simulations. This isotherm is well above the critical temperature of the corresponding monomeric fluid ($T_c = 1.085$, see Figure 3.3), but the critical temperature of the chain molecules is appreciably higher [356].

13.3 Generation of Trial Orientations

The efficient generation of good trial conformations is an essential aspect of the configurational-bias Monte Carlo scheme for continuum models with strong intramolecular interactions. For some models (for example, Gaussian chains) it is possible to generate this distribution directly. For an arbitrary model we can use the acceptance-rejection technique [33] of generating the trial orientations.

Here, we show how a rejection technique can be used to generate trial positions efficiently. The number of trial directions in the CBMC scheme can be chosen at will. Often, the optimal number of trial directions is determined empirically. However, more systematic techniques exist to compute this optimal number [357].

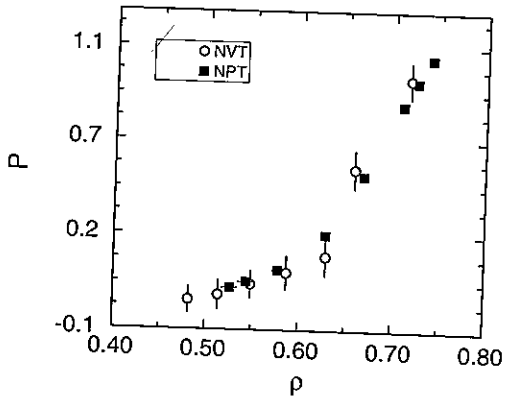

Figure 13.4: Equation of state of an eight-bead Lennard-Jones chain as obtained from N,V,T and N,P,T simulations using the configurational-bias Monte Carlo scheme. The simulations are performed with 50 chains at a temperature $T^* = 1.9$.

13.3.1 Strong Intramolecular Interactions

Let us consider as an example a model of a molecule in which the bonded interactions include bond stretching, bond bending, and torsion. The external interactions are the nonbonded interactions. A united atom model of an alkane is a typical example of such a molecule.

The probability that we generate a trial configuration $\mathbf{b}$ is given by, (see equation (13.2.10))

$$P(\mathbf{b})d\mathbf{b} = C\exp[-\beta u^{bond}(\mathbf{b})]d\mathbf{b}. \tag{13.3.1}$$

It is convenient to represent the position of an atom using the bond length r, bond angle θ, and torsional angle ϕ (see Figure 13.5). With these coordinates the volume element $d\mathbf{b}$ is given by

$$d\mathbf{b} = r^2\,dr d\cos\theta d\phi. \tag{13.3.2}$$

The bonded energy is the sum of the bond-stretching potential, the bond-bending potential, and the torsion potential:

$$u^{bond}(r,\theta,\phi) = u_{vib}(r) + u_{bend}(\theta) + u_{tors}(\phi). \tag{13.3.3}$$

Substitution of equations (13.3.3) and (13.3.2) into equation (13.3.1) gives

$$\begin{aligned} P(\mathbf{b})\,d\mathbf{b} &= P(r,\theta,\phi)r^2\,dr d\cos\theta d\phi \\ &= C\exp[-\beta u_{vib}(r)]r^2dr \times \exp[-\beta u_{bend}(\theta)]d\cos\theta \\ &\quad \times \exp[-\beta u_{tors}(\phi)]d\phi. \end{aligned} \tag{13.3.4}$$

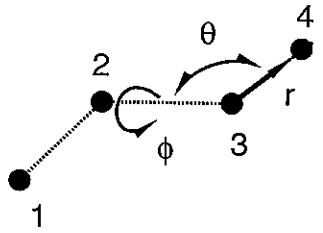

Figure 13.5: Schematic sketch of a part of a molecule.

Many models use a fixed bond length, in which case the first term in equation (13.3.4) is a constant.

Let us consider the molecule shown in Figure 13.5. The first atom is placed at a random position and we now have to add the second atom. For convenience, it is assumed that the model has a fixed bond length. The second atom has no bonded interactions other than the constraints on the bond length. The distribution of trial orientations, equation (13.3.4), reduces to

$$P_2(\mathbf{b})d\mathbf{b} \propto d\cos\theta d\phi. \tag{13.3.5}$$

Hence, the trial orientations are randomly distributed on the surface of a sphere (such a distribution can be generated with Algorithm 42 in Appendix J).

For the third atom, the bonded energy contains the bond-bending energy as well. This gives, for the distribution of trial orientations,

$$P_3(\mathbf{b})d\mathbf{b} \propto \exp[-\beta u_{bend}(\theta)]\,d\cos\theta d\phi. \tag{13.3.6}$$

To generate k trial orientations distributed according to equation (13.3.6), we again generate a random vector on a unit sphere and determine the angle θ. This vector is accepted with a probability $\exp[-\beta u_{bend}(\theta)]$. If rejected, this procedure is repeated until a value of θ has been accepted. In [33], this acceptance-rejection method is shown to indeed give the desired distribution of trial orientations. In this way, k (or $k-1$, for the old conformation) trial orientations are generated.

An alternative scheme would be to generate angle θ uniformly ($\theta \in [0,\pi]$) and to determine the bond-bending energy corresponding to this angle. This angle θ is accepted with a probability $\sin(\theta)\exp[-\beta u_{bend}(\theta)]$. If rejected, this procedure is repeated until a value of θ has been accepted. The selected value of θ is supplemented with a randomly selected angle ϕ. These two angles determine a new trial orientation.

Algorithm 25 (Growing an Alkane)

```
  SUBROUTINE grow(new_conf,w)          grow or retrace an alkane and
                                       calculate its Rosenbluth factor w
  if (new_conf) then                   new_conf =.true.: new conf.
     ib=int(ranf()*ell)+1              start to grow from position ib
     ibnewconf=ib                      store starting position
  else                                 new_conf =.false.: old conf.
     ib=ibnewconf                      same starting position to regrow
  endif                                as used for the new configuration
  do i=1,ib-1
    xn(i)=x(i)                         store positions that are not regrown
  enddo
  w=1
  do i=ib,ell
   if (ib.eq.1) then                   first atom
     if (new_conf) then
      xt(1)=ranf()*box                 generate random position
     else
      xt(1)=xn(1)                      use old position
     endif
     call enerex(xt(1),eni)            calculate (external) energy
     w=k*exp(-beta*eni)                and Rosenbluth factor
   else                                second and higher atoms
     sumw=0
     do j=1,k
      if (.not.new_conf
+       .and. j.eq.1) then
       xt(1)=x(i)                      actual position as trial orientation
      else
       call next_ci(xt(j),xn,i)        generate trial position
      endif
      call enerex(xt(j),eni)           (external) energy of this position
      wt(j)= exp(-beta*eni)
      sumw=sumw+wt(j)
     enddo
     w=w*sumw                          update Rosenbluth factor
     if (new_conf) then
       call select(wt,sumw,n)          select one of the trial orientations
       xn(i)=xt(n)
       xstore(i)=xt(n)                 store selected configuration
     else                              for bookkeeping
       xn(i)=x(i)
     endif
   endif
  enddo
  return
  end
```

Comments to this algorithm:

1. *Subroutine* `enerex` *calculates the external energy of an atom at the given position, and subroutine* `select` *selects one of the trial positions with probability* $p(i) = w(i)/\sum_j w(j)$ *(Algorithm 41).*
2. *Subroutine* `next_ci` *adds the next atom to the chain as prescribed by the bonded interactions (Algorithms 26, 27, and 28 are examples for ethane, propane, and higher alkanes, respectively).*

For the fourth and higher atoms, the bonded energy includes both bond-bending and torsion energy. This gives, for equation (13.3.4),

$$p_l^{\text{bond}}(\mathbf{b})d\mathbf{b} \propto \exp[-\beta u_{\text{bend}}(\theta)] \exp[-\beta u_{\text{tors}}(\phi)]\, d\cos\theta d\phi. \qquad (13.3.7)$$

We again generate a random vector on a sphere and calculate the bond-bending angle θ and torsion ϕ. These angles are accepted with a probability $\exp\{-\beta[u_{\text{bend}}(\theta) + u_{\text{tors}}(\phi)]\}$. If these angles are rejected, new vectors are generated until one gets accepted.

Again an alternative scheme is to determine first a bond-bending angle θ by generating θ uniformly on $[0, \pi]$ and calculating the bond-bending energy corresponding to this angle. This angle θ is then accepted with a probability $\sin(\theta)\exp[-\beta u_{\text{bend}}(\theta)]$. This procedure is continued until we have accepted an angle. Next we generate a torsion angle randomly on $[0, 2\pi]$ and accept this angle with a probability $\exp[-\beta u_{\text{tors}}(\phi)]$, again repeating this until a value has been accepted. In this scheme the bond angle and torsion are generated independently, which can be an advantage in cases where the corresponding potentials are sharply peaked.

The acceptance-rejection technique is illustrated in Algorithms 25–28 for different n-alkanes. For all-atom or explicit-hydrogen models of hydrocarbons, a different strategy is needed for which we refer the reader to the relevant literature [358,359].

Case Study 19 (Generation of Trial Configurations of Ideal Chains)
In section 13.2.3, we emphasized the importance of efficiently generating trial segments for molecules with strong intramolecular interactions. In this case study, we quantify this. We consider the following bead-spring model of a polymer. The nonbonded interactions are described with a Lennard-Jones potential and the bonded interactions with a harmonic spring:

$$u^{\text{vib}}(l) = \begin{cases} 0.5k_{\text{vib}}(l-1)^2 & 0.5 \le l \le 1.5 \\ \infty & \text{otherwise} \end{cases},$$

where l is the bond length, the equilibrium bond length has been set to 1, and $k_{\text{vib}} = 400$. The bonded interaction is only the bond stretching. The external (nonbonded) interactions are the Lennard-Jones interactions. We consider the following two schemes of generating a set of trial positions:

Algorithm 26 (Growing Ethane)

```
SUBROUTINE next_c2(xn,xt,i)      generate a trial position for ethane
                                 position of the first atom is known
call bondl(l)                    generate bond length
call ranor(b)                    generate vector on unit sphere
xt(i)=xn(i-1)+l*b
return
end
```

Comment to this algorithm:

1. *The subroutine* `ranor` *generates a random vector on a unit sphere (Algorithm 42), and the subroutine* `bondl` *(Algorithm 43) generates the bond length prescribed by the bonded interactions.*

Algorithm 27 (Growing Propane)

```
SUBROUTINE next_c3(xn,           generate a trial position for ith atom
+                  xt,i)         position of the (i − 1)th atom is known
call bondl(l)                    generate bond length
if (i.eq.2) then                 second atom
  call next_c2(xn,xt,i)          use Algorithm 26
else if (i.eq.3) then            third atom
  call bonda(xn,b,i)             generate orientation of the
  xt=xn(2)+l*b                   new position with desired bond angle
else
  STOP 'error'
endif
return
end
```

Comment to this algorithm:

1. *The subroutine* `ranor` *generates a random vector on a unit sphere (Algorithm 42), the subroutine* `bondl` *(Algorithm 43) generates the bond length prescribed by the bonded interactions (for the second atom, only bond stretching), and the subroutine* `bonda` *generates a vector on a unit sphere with bond angle prescribed by the bond-bending potential (Algorithm 45).*

Algorithm 28 (Generating a Trial Position for an Alkane)

```
SUBROUTINE next_cn(xn,xt,i)      generate a trial position for ith atom
                                 position of atoms (i − 1) are known
call bondl(l)                    generate bond length
if (i.eq.2) then                 second atom
  call next_c2(xn,xt,i)          use Algorithm 26
else if (i.eq.3) then            third atom
  call next_c3(xn,xt,i)          use Algorithm 27
else if (i.ge.4) then            fourth and higher atoms
  call tors_bonda(xn,b,i)        generate vector with prescribed
  xt=xn(i-1)+l*b                 bond and torsional angles
endif
return
end
```

Comment to this algorithm:

1. *The subroutine* `tors_bonda` *(Algorithm 46) generates bond bending and a torsional angle prescribed by the corresponding potentials.*

1. Generate a random orientation with bond length uniformly distributed in the spherical shell between limits chosen such that they bracket all acceptable bond lengths. For instance, we could consider limits that correspond to a 50% stretching or compression of the bond. In that case, the probability of generating bond length l is given by

$$p_1(l)\begin{cases} \propto C dl \propto l^2 dl & 0.5 \le l \le 1.5 \\ 0 & \text{otherwise} \end{cases}$$

2. Generate a random orientation and the bond length prescribed by the bond-stretching potential (as described in Algorithm 26). The probability of generating bond length l with this scheme is

$$p_2(l)\begin{cases} \propto C\exp[-\beta u^{vib}(l)]dl = C\exp[-\beta u^{vib}(l)]l^2 dl & 0.5 \le l \le 1.5 \\ 0 & \text{otherwise} \end{cases}$$

Let us consider a case in which the system consists of ideal chains. Ideal chains are defined (see section 13.2.3) as chains having only *bonded* interactions.

Suppose we use method 1 to generate the set of k trial orientations with bond lengths $l_1, \cdots, l_k$, then the Rosenbluth factor for atom i is given by

$$w_i(n) = \sum_{j=1}^{k} \exp[-\beta u^{vib}(l_j)].$$

The Rosenbluth factor of the entire chain is

$$W(n) = \prod_{i=1}^{\ell} w_i(n).$$

For the old conformation a similar procedure is used to calculate its Rosenbluth factor:

$$W(o) = \prod_{i=1}^{\ell} w_i(o).$$

In absence of external interactions the Rosenbluth factor of the first atom is defined to be $w_1 = k$.

In the second scheme, we generate the set of k trial orientations with a bond length distribution $p_2(l)$. If we use this scheme, we have to consider only the external interaction. Since, for an ideal chain, the external interactions are by definition 0, the Rosenbluth factor for each atom is given by

$$w_i^{ext}(n) = \sum_{j=1}^{k} \exp[-\beta u^{ext}(l_j)] = k,$$

and similarly, for the old conformation

$$w_i^{ext}(o) = k.$$

Hence, the Rosenbluth weight is the same for the new and the old conformations:

$$W^{ext}(n) = \prod_{i=1}^{\ell} w_i^{ext}(n) = k^{\ell}$$

and

$$W^{ext}(o) = \prod_{i=1}^{\ell} w_i^{ext}(o) = k^{\ell}.$$

The acceptance rule for the first scheme is

$$acc(o \to n) = \min[1, W(n)/W(o)]$$

and for the second scheme is

$$acc(o \to n) = \min[1, W^{ext}(n)/W^{ext}(o)] = 1.$$

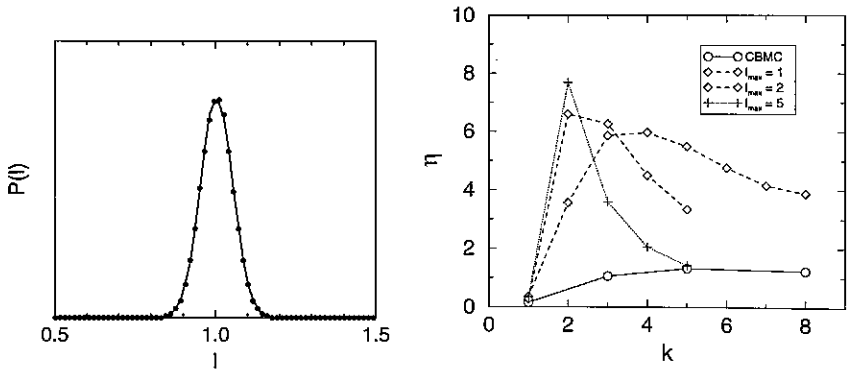

Figure 13.6: Comparison of methods 1 and 2 for the distribution of bond lengths l (left) and the distribution of the radius of gyration R_g (right). The solid lines represent the results for method 1, the dots for method 2 ($\ell = 5$ and $k = 5$).

Inspection of these acceptance rules shows that, in the second scheme, *all* configurations generated are accepted, whereas in the first scheme this probability depends on the bond-stretching energy and therefore will be less than 1. Hence, it is clearly useful to employ the second scheme.

To show that the results of schemes 1 and 2 are indeed equivalent, we compare the distribution of the bond length of the chain and the distribution of the radius of gyration in Figure 13.6. The figure shows that the results for the two methods are indeed indistinguishable. The efficiency of the two methods, however, is very different. In Table 13.1, the difference in acceptance probability is given for some values of the bond-stretching force constant and various chain lengths. The table shows that if we use method 1 and generate a uniformly distributed bond length, we need to use at least 10 trial orientations to have a reasonable acceptance for chains longer than 20 monomers. Note that the corresponding table for the second method has a 100% acceptance for all values of k independent of the chain length.

Most of the simulations, however, do not involve ideal chains but chains with external interactions. For chains with external interactions, the first method performs even worse. First of all, we generate the chains the same way as in the case of the ideal chains. The bonded interactions are the same and we need to generate at least the same number of trial directions to get a reasonable acceptance. In addition, if there are external interactions, we have to calculate the nonbonded interactions for *all* of these trial positions. The calculation of the nonbonded interactions takes most of the CPU time; yet, in the first method, most of the trial orientations are doomed to be re-

k	$\ell = 5$	$\ell = 10$	$\ell = 20$	$\ell = 40$	$\ell = 80$	$\ell = 160$
1	0.6	$\ll$0.01	$\ll$0.01	$\ll$0.01	$\ll$0.01	$\ll$0.01
5	50	50	10	$\ll$0.01	$\ll$0.01	$\ll$0.01
10	64	58	53	42	$\ll$0.01	$\ll$0.01
20	72	66	60	56	44	$\ll$0.01
40	80	72	67	62	57	40
80	83	78	72	68	62	60

Table 13.1: Probability of acceptance (%) for ideal chains using uniformly distributed bond lengths (method 1), where ℓ is the chain length, and k is the number of trial orientations. The value for the spring constant is $k_{vib} = 400$ (see [289]). For method 2, the acceptance would have been 100% for all values of k and ℓ.

jected solely on the basis of the bonded energy. These two reasons make the second scheme much more attractive than the first.

13.3.2 Generation of Branched Molecules

The generation of trial configurations for branched alkanes requires some care. Naively, one might think that it is easiest to grow a branched alkane atom by atom. However, at the branchpoint we have to be careful. Suppose we have grown the backbone shown in Figure 13.7 and we now have to add the branches b_A and b_B. The total bond-bending potential has three contributions, given by

$$u_{bend} = u_{bend}(c_1, c_2, b_A) + u_{bend}(c_1, c_2, b_B) + u_{bend}(b_A, c_2, b_B).$$

Vlugt [360] pointed out that, because of the term $u_{bend}(b_A, c_2, b_B)$, it is better not to generate the positions of b_A and b_B independently. Suppose that we would try to do this anyway. We would then generate the first trial position, b_A, according to

$$P(b_A) \propto \exp\left[-\beta u_{bend}(c_1, c_2, b_A)\right],$$

next we would generate the second trial position, b_B, using

$$P(b_B|b_A) \propto \exp\{-\beta\left[u_{bend}(c_1, c_2, b_B) + u_{bend}(b_A, c_2, b_B)\right]\},$$

where $P(b_B|b_A)$ denotes the probability of generating b_B for a given position of segment b_A. However, if we would generate both positions at the same time, then the probability is given by

$$\begin{aligned} &P(b_A, b_B) \\ &\quad \propto \exp\{-\beta\left[u_{bend}(c_1, c_2, b_A) + u_{bend}(c_1, c_2, b_B) + u_{bend}(b_A, c_2, b_B)\right]\}. \end{aligned}$$

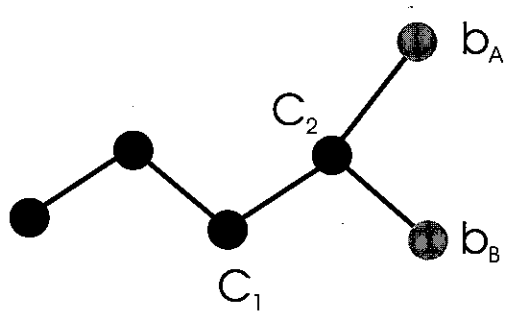

Figure 13.7: Growth of a banched alkane.

The two schemes are only equivalent if

$$P(b_A, b_B) = P(b_B|b_A)P(b_A).$$

In general this equality does not hold. To see this, compare the probability of generating configuration b_A for the two schemes. This probability is obtained by integrating over all orientations b_B. If both chains are inserted at the same time, we find that

$$\begin{aligned} P(b_A) &= \int db_B P(b_A, b_B) \\ &\propto \exp\left[-\beta u_{bend}(c_1, c_2, b_A)\right] \\ &\quad \times \int db_B \exp\{-\beta\left[u_{bend}(c_1, c_2, b_A) + u_{bend}(b_A, c_2, b_B)\right]\}. \end{aligned}$$

For the sequential scheme, we would have obtained

$$\begin{aligned} P(b_A) &= \int db_B P(b_B|b_A)P(b_A) \\ &= P(b_A) \\ &\propto \exp\left[-\beta u_{bend}(c_1, c_2, b_A)\right] \end{aligned}$$

as, in this scheme, segment b_A is inserted before segment b_B. Therefore the probability $P(b_A)$ cannot depend on b_B.

We can now easily see that if we use a model in which the two branches are equivalent, for example, isobutane, the sequential scheme does not generate equivalent *a priori* distributions for the two branches. Of course, the generation of trial segments is but one step in the CBMC scheme. Any bias introduced at this stage can be removed by incorporating the ratio of the true and the biased distributions in the acceptance criterion. However, the resulting algorithm may be inefficient. Vlugt *et al.* [361] have shown that simply