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 Hence, it would seem tempting to conclude that the MC method is an elegant but outdated scheme.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-
Chapter 13. Biased Monte Carlo Schemes

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, 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 \textit{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.

\footnote{Readers who are not familiar with the Rosenbluth scheme are advised to read section 11.2 first.}
13.1 Biased Sampling Techniques

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:

\[ \alpha(o \rightarrow n) = f[U(n)] \]

For the reverse move, we have

\[ \alpha(n \rightarrow o) = f[U(o)] \]

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,

\[ \frac{\text{acc}(o \rightarrow n)}{\text{acc}(n \rightarrow o)} = \frac{f[U(o)]}{f[U(n)]} \exp\{-\beta[U(n) - U(o)]\} \]

A possible acceptance rule that obeys this condition is

\[ \text{acc}(o \rightarrow n) = \min\left( 1, \frac{f[U(o)]}{f[U(n)]} \exp\{-\beta[U(n) - U(o)]\} \right) \]

This derivation shows that we can introduce an arbitrary biasing function \( f(U) \) 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.
Chapter 13. Biased Monte Carlo Schemes

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 \( u_{\text{pos}}(n) \). In practice, there may be several ways to separate the potential into orientation-dependent and orientation-independent parts.

2. Generate \( k \) trial orientations \( \{b_1, b_2, \ldots, b_k\} \) and for each of these trial orientations, calculate the energy \( E(b_i) \).

3. We define the Rosenbluth\(^2\) factor

\[
W(n) = \sum_{j=1}^{k} \exp[-\beta u_{\text{or}}(b_j)]. \tag{13.1.2}
\]

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

\[
p(b_n) = \frac{\exp[-\beta u_{\text{or}}(b_n)]}{\sum_{j=1}^{k} \exp[-\beta u_{\text{or}}(b_j)]}. \tag{13.1.3}
\]

4. For the old configuration, \( o \), the part of the energy that does not depend on the orientation of the molecules is denoted by \( u_{\text{pos}}(o) \). The orientation of the molecule in the old position is denoted by \( b_o \), and we generate \( k - 1 \) trial orientations denoted by \( b_2, \ldots, b_k \). Using these \( k \) orientations, we determine

\[
W(o) = \exp[-\beta u_{\text{or}}(b_o)] + \sum_{j=2}^{k} \exp[-\beta u_{\text{or}}(b_j)]. \tag{13.1.4}
\]

5. The move is accepted with a probability

\[
\text{acc}(o \rightarrow n) = \min \left( 1, \frac{W(n)}{W(o)} \exp[-\beta (u_{\text{pos}}(n) - u_{\text{pos}}(o))] \right). \tag{13.1.5}
\]

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

o=int(ranf()*npart)+1
xt=ranf()*box
call ener(xt,en)
wn=exp(-beta*en)
sumw=0

do j=l,k
    call ranor(b(j))
call enero(xt,b(j),eno)
w(j)= exp(-beta*eno)
    sumw=sumw+w(j)
endo
call select(w,sum,n)
bn=b(n)
wn=wn*sumw

call ener(x(o),en)
wo=exp(-beta*en)
sumw=0

do j=l,k
    if (j.eq.l) then
        b(j)=u(o)
    else
        call ranor(b(j))
    endif
call enero(x(o),b(j),eno)
sumw=sumw+exp(-beta*eno)
endo
wo=wo*sumw
if (ranf() lt wn/wo) +
    call accept
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)

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

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

\[
U = U^{or} + U^{pos}.
\]

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):

\[
K(o \rightarrow n) = K(n \rightarrow o).
\]

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

\[
K(o \rightarrow n) = \mathcal{N}(o) \times \alpha(o \rightarrow n) \times \text{acc}(o \rightarrow n). \tag{13.1.6}
\]
In the orientational-bias scheme, the probability of selecting conformation \( n \) is (see equation (13.1.3))

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

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:

\[
\frac{\text{acc}(o \rightarrow n)}{\text{acc}(n \rightarrow o)} = \frac{\exp[-\beta \mathcal{U}(n)]}{\exp[-\beta \mathcal{U}(o)]} \times \frac{\exp[-\beta u^{or}(o)]}{W(o)} \times \frac{W(n)}{\exp[-\beta u^{or}(n)]} = \frac{W(n)}{W(o)} \exp[-\beta (u^{pos}(n) - u^{pos}(o))].
\]

(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 cannot hope to sample all possible orientations. It is impossible to determine the exact Rosenbluth factor since an infinite number of orientations are possible. 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

\[
\{b\}_k = \{b_1, b_2, \ldots, b_k\}.
\]

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

\[
\mathcal{B}_n = \{ \{b\}_k | b_n \in \{b\}_k \}.
\]

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

---

3In Example 17 we discuss a special case for which the Rosenbluth factor can be calculated exactly.
consider the sum over all sets in $B_n$

$$K(o \rightarrow n) = \mathcal{N}(o) \sum_{i \in B_n} \alpha(o \rightarrow n, i) \times \text{acc}(o \rightarrow n, i),$$  \hspace{1cm} (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 $B_0$

$$B_0 = \{\{b\}_k | \{b\}_o \in \{b\}_k\},$$

for which each element can be written as $(b_o, b''')$. The expression for the reverse flow then becomes

$$K(n \rightarrow o) = \mathcal{N}(n) \sum_{j \in B_o} \alpha(n \rightarrow o, j) \times \text{acc}(n \rightarrow o, j).$$  \hspace{1cm} (13.1.9)$$

It should be stressed that infinitely many different sets of orientations include $b_n$, and the same holds for sets that include $b_o$. Moreover, the probability of selecting $b_n$ 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,

$$K(o \rightarrow n, b^*, b''') = \mathcal{N}(n) \alpha(o \rightarrow n, b^*, b''') \text{acc}(o \rightarrow n, b^*, b''')$$

$$= \mathcal{N}(n) \alpha(n \rightarrow o, b''', b^*) \text{acc}(n \rightarrow o, b''', b^*),$$  \hspace{1cm} (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 $b_n$ in the forward move is equal to the a priori probability of generating $b_o$ 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

$$\frac{\text{acc}(o \rightarrow n, b^*, b'')} \text{acc}(n \rightarrow o, b'', b^*) = \frac{\exp[-\beta u_l(n)] \exp[-\beta u^\text{or}(o)] \ W(b_n, b^*)}{\exp[-\beta u_l(o)] W(b_o, b'') \ \exp[-\beta u^\text{or}(n)]} \frac{W(b_n, b^*)}{W(b_o, b'')} \ \exp[-\beta [u^{\text{pos}}(n) - u^{\text{pos}}(o)]].$$

(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 $\omega$ 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 $\psi_i$ 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

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

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 \( (E) \) at the position of the inserted particle and that at the position of the old configuration are known:

\[ W(r) = \int dB \exp[-\beta \mathbf{\mu} \cdot E(r)] = \frac{\sinh[\beta |\mathbf{\mu}| E(r)]}{|\beta |^2 |\mathbf{\mu}| E(r)}, \]

where \( \mathbf{\mu} \) is the dipole moment of the molecule.\(^4\) A trial orientation can now

\(^4\)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.
be drawn directly from the distribution

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

### 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.
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

$$\text{acc}(o \rightarrow n) = \min[1, W(n)/W(o)].$$

The generation of a trial conformation $n$ of a polymer consisting of $\ell$ 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 $w_1(n) = k \exp[-\beta u_1(n)]$, 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

---

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.
directions, we select one, say, \( n \), with a probability

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

(13.2.2)

where \( w_i(n) \) is defined as

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

(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 \( \ell \). 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 \( n \):

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

(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:

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

(13.2.5)
Algorithm 23 (Basic Configurational-Bias Monte Carlo)

Program CBMC

new_conf=.false.
call grow(new_conf, wo)
new_conf=.true.
call grow(new_conf, wn)
if (ranf()>wn/wo)
+   call accept
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

$$\text{acc}(o \rightarrow n) = \min[1, W(n)/W(o)]. \quad (13.2.6)$$

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):

$$\alpha(o \rightarrow n) = \prod_{i=1}^{\ell} \frac{\exp[-\beta u_i(n)]}{w_i(n)} = \frac{\exp[-\beta U(n)]}{W(n)}. \quad (13.2.7)$$
Algorithm 24 (Growing a Chain on a Lattice)

SUBROUTINE grow(new_conf,w)

if (new_conf) then
    xn(1)=ranf() * box
else
    o=ranf() * npart + 1
    xn(1)=x(o,1)
endif

call ener(xn(1),en)
w=k*exp(-beta*en)
do i=2,ell
    sumw=0
    do j=1,k
        xt(j)=xn(i-1) + b(j)
        call ener(xt(j),en)
        w(j)=exp(-beta*en)
        sumw=sumw+w(j)
    enddo
    if (new_conf) then
        call select(w, sumw,n)
        xn(i)=xt(n)
    else
        xn(i)=x(o,i)
    endif
    w=w*sumw
endo
return
end

grow an ℓ bead polymer on a lattice
with coordination number k and
calculate its Rosenbluth factor w
insert the first monomer
select old chain at random
calculate energy
Rosenbluth factor first monomer
consider the k trial directions
determine trial position
determine energy trial position j
select one of the trial position
direction n is selected
update Rosenbluth factor

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)/ ∑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,

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

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

$$\frac{\text{acc}(o \rightarrow n)}{\text{acc}(n \rightarrow o)} = \frac{W(n)}{W(o)}.$$  \hfill (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 $\ell$. 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 $\ell$—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 \( \ell \) linear segments, the potential energy of a given conformation \( \mathcal{U} \) has two contributions:

1. The *bonded potential energy* \( \mathcal{U}^{\text{bond}} \) 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 \( u_{i}^{\text{bond}} \) that depends on the angle \( \theta \) between the successive segments. For instance, \( u_{i}^{\text{bond}}(\theta) \) could be of the form \( u_{i}^{\text{bond}}(\theta) = k\theta(\theta - \theta_0)^2 \). For realistic models for polyatomic molecules, \( u_{i}^{\text{bond}} \) 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* \( \mathcal{U}^{\text{ext}} \) 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 \( \mathcal{U}^{\text{ext}} \).

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 \( \ell \) 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 \( \mathcal{C} \), next we consider the old conformation \( \mathcal{O} \). 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 \( i \) (\( u_{i}^{\text{bond}} \)). We denote this set of \( k \) different trial segments by

\[
\{b\}_k = \{b_1, \cdots, b_k\},
\]

where the probability of generating a trial segment \( b \) is given by

\[
p_{i}^{\text{bond}}(b)db = \frac{\exp[-\beta u_{i}^{\text{bond}}(b)]db}{\int db \exp[-\beta u_{i}^{\text{bond}}(b)]} = C \exp[-\beta u_{i}^{\text{bond}}(b)]db.
\]

(13.2.10)
Chapter 13. Biased Monte Carlo Schemes

2. For all \( k \) trial segments, we compute the external Boltzmann factors \( \exp[-\beta u_{i}^{\text{ext}}(b_i)] \), and out of these, we select one, denoted by \( r_t \), with a probability

\[
p_i^\text{ext}(b_n) = \frac{\exp[-\beta u_i^{\text{ext}}(b_n)]}{w_i^\text{ext}(n)},
\]

where we have defined

\[
w_i^\text{ext}(n) = \sum_{j=1}^{k} \exp[-\beta u_i^{\text{ext}}(b_j)].
\]

3. The selected segment \( r_t \) becomes the \( i \)th segment of the trial conformation of the chain.

4. When the entire chain is grown, we calculate the Rosenbluth factor of the chain:

\[
W_i^\text{ext}(n) = \prod_{i=1}^{\ell} w_i^\text{ext}(n),
\]

where Rosenbluth factor of the first monomer is defined by

\[
w_1^\text{ext}(n) = k \exp[-\beta u_1^{\text{ext}}(r_1)],
\]

where \( r_1 \) 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

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

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 \( (b_o, b'^*) \). These orientations are used to calculate the external Rosenbluth factor:

\[
w_i^\text{ext}(o) = \sum_{j=1}^{k} \exp[-\beta u_i^{\text{ext}}(b_j)].
\]
4. For the entire chain the Rosenbluth factor of the old conformation is defined by

\[ W^{\text{ext}}(o) = \prod_{i=1}^{\ell} w_i^{\text{ext}}(o). \]  

(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

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

(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 \( \ell \) 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 \( n \),

\[ \alpha(o \rightarrow n) = \prod_{i=1}^{\ell} p_i(o \rightarrow n) = \prod_{i=1}^{\ell} p_i^{\text{bond}}(n)p_i^{\text{ext}}(n). \]  

(13.2.19)

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

\[ P^{\text{bond}}(b^*, b^{**}). \]
Hence, the flow of configurations is given by

\[
K(o \rightarrow n, b^*, b'^*) = \mathcal{N}(o) \times \alpha(o \rightarrow n, b^*, b'^*) \times \text{acc}(o \rightarrow n, b^*, b'^*) \\
= \exp[-\beta u(o)] \times C \exp[-\beta u^{\text{bond}}(n)] \times \frac{\exp[-\beta u^{\text{ext}}(n)]}{\omega^{\text{ext}}(b_n, b^*)} \\
\times \text{acc}(o \rightarrow n, b^*, b'^*) \mathcal{P}^{\text{bond}}(b^*, b'^*). 
\] (13.2.20)

For the reverse move, we have

\[
K(n \rightarrow o, b'^*, b^*) = \mathcal{N}(n) \times \alpha(n \rightarrow o, b'^*, b^*) \times \text{acc}(n \rightarrow o, b'^*, b^*) \\
= \exp[-\beta u(n)] \times C \exp[-\beta u^{\text{bond}}(o)] \times \frac{\exp[-\beta u^{\text{ext}}(o)]}{\omega^{\text{ext}}(b_o, b'^*)} \\
\times \text{acc}(n \rightarrow o, b'^*, b^*) \mathcal{P}^{\text{bond}}(b^*, b'^*). 
\] (13.2.21)

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

\[
u(n) = u^{\text{bond}}(n) + u^{\text{ext}}(n).
\]

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

\[
\frac{\text{acc}(o \rightarrow n, b^*, b'^*)}{\text{acc}(n \rightarrow o, b'^*, b^*)} = \frac{\omega^{\text{ext}}(b_n, b^*)}{\omega^{\text{ext}}(b_o, b'^*)}. 
\] (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:

\[
\frac{\text{acc}[o \rightarrow n, (b_1^*, \ldots, b_\ell^*)]}{\text{acc}[n \rightarrow o, (b_1'^*, \ldots, b_\ell'^*)]} = \frac{\prod_{i=1}^\ell \omega^{\text{ext}}(b_n, b_i^*)}{\prod_{i=1}^\ell \omega^{\text{ext}}(b_o, b_i'^*)} = \frac{W[n, (b_1^*, \ldots, b_\ell^*)]}{W[o, (b_1'^*, \ldots, b_\ell'^*)]}.
\] (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.
13.3 Generation of Trial Orientations

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

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

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

The simulations are performed in cycles. In each cycle, we perform on average $N_{\text{dis}}$ attempts to displace a particle, $N_{\text{cbmc}}$ attempts to (partly) regrow a chain, and $N_{\text{vol}}$ 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 b is given by, (see equation (13.2.10))

\[ P(b)db = C \exp[-\beta u^{\text{bond}}(b)]db. \]  

(13.3.1)

It is convenient to represent the position of an atom using the bond length r, bond angle \( \theta \), and torsional angle \( \phi \) (see Figure 13.5). With these coordinates the volume element \( db \) is given by

\[ db = r^2 drd \cos \theta d\phi. \]  

(13.3.2)

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

\[ u^{\text{bond}}(r, \theta, \phi) = u_{\text{vib}}(r) + u_{\text{bend}}(\theta) + u_{\text{tors}}(\phi). \]  

(13.3.3)

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

\[
P(b) \, db = P(r, \theta, \phi) r^2 \, dr \, d\cos \theta \, d\phi \\
= C \exp[-\beta u_{\text{vib}}(r)]r^2 \, dr \times \exp[-\beta u_{\text{bend}}(\theta)] \, d\cos \theta \\
\times \exp[-\beta u_{\text{tors}}(\phi)] \, d\phi. \]

(13.3.4)
13.3 Generation of Trial Orientations

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

\[ \frac{1}{\sin \theta} \frac{d\theta}{\sin \phi} \frac{d\phi}{d\Omega}, \]

which is a constant.

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

\[ P_3(b)db \propto \sin \theta e^{-\beta \mu_{\text{bend}}(\theta)} d\cos \theta d\phi. \]

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,

\[ P_3(b)db \propto \sin \theta e^{-\beta \mu_{\text{bend}}(\theta)} d\cos \theta d\phi. \]

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 \( \theta \). This vector is accepted with a probability \( e^{-\beta \mu_{\text{bend}}(\theta)} \). If rejected, this procedure is repeated until a value of \( \theta \) 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 \( \theta \) uniformly (\( \theta \in [0, \pi] \)) and to determine the bond-bending energy corresponding to this angle. This angle \( \theta \) is accepted with a probability \( \sin(\theta) e^{-\beta \mu_{\text{bend}}(\theta)} \). If rejected, this procedure is repeated until a value of \( \theta \) has been accepted. The selected value of \( \theta \) is supplemented with a randomly selected angle \( \phi \). These two angles determine a new trial orientation.

\[ \frac{1}{\sin \theta} \frac{d\theta}{\sin \phi} \frac{d\phi}{d\Omega}, \]

which is a constant.
Algorithm 25 (Growing an Alkane)

```
SUBROUTINE grow (new_conf, w)
  if (new_conf) then
    ib=int(ranf ()*ell)+1
    ibnewconf=ib
  else
    ib=ibnewconf
  endif
  do i=l, ib-L
    xn(i) =x(i)
  enddo
  w=l
  do i=ib, ell
    if (ib.eq.l) then
      if (new_conf) then
        xt (i) =ranf () *box
      else
        xt (i) =xn (i)
      endif
      call enerex(xt(1),eni)
      w=k* exp ( -beta*eni )
    else
      sumw= 0
      do j=l,k
        if (.not.new_conf and. j.eq.l) then
          xt (i) =x(i)
        else
          call next_cI(xt(j), xn, i)
        endif
        call enerex(xt(j),eni)
        wt(j)= exp(-beta*eni)
        sumw=sumw+wt (j)
      enddo
      W=W* sumw
      if (new_conf) then
        call select(wt,sumw,n)
        xn ( i ) =xt (n)
        xstore (i) =xt (n)
      else
        xn(i) =x(i)
      endif
    endif
  enddo
  return
end
```

grow or retrace an alkane and calculate its Rosenbluth factor w
new_conf =.true.: new conf.
start to grow from position ib
store starting position
new_conf =.false.: old conf.
same starting position to regrow as used for the new configuration
store positions that are not regrown

first atom
generate random position
use old position
calculate (external) energy and Rosenbluth factor
second and higher atoms
actual position as trial orientation
generate trial position
(external) energy of this position
update Rosenbluth factor
select one of the trial orientations
store selected configuration for bookkeeping
13.3 Generation of Trial Orientations

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 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),

\[
\frac{\sum b}{\exp[-\beta v_{bend}(\theta)] \exp[-\beta v_{tors}(\phi)]} \cos \theta d\phi.
\]

We again generate a random vector on a sphere and calculate the bond-bending angle \( \theta \) and torsion \( \phi \). These angles are accepted with a probability \( \exp[-\beta(v_{bend}(\theta) + v_{tors}(\phi))] \). 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 \( \theta \) by generating \( \theta \) uniformly on \([0, \pi]\) and calculating the bond-bending energy corresponding to this angle. This angle \( \theta \) is then accepted with a probability \( \sin(\theta) \exp[-\beta v_{bend}(\theta)] \). 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 \( \exp[-\beta v_{tors}(\phi)] \), 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:

\[
u_{vib}(l) = \begin{cases} 0.5k_{vib}(l - 1)^2 & 0.5 \leq l \leq 1.5 \\ \infty & \text{otherwise} \end{cases}
\]

where \( l \) is the bond length, the equilibrium bond length has been set to 1, and \( k_{vib} = 400 \). 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)
    call bondl(1)
    call ranor(b)
    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, xt, i)
    call bondl(1)
    if (i.eq.2) then
        call next_c2(xn, xt, i)
    else if (i.eq.3) then
        call bonda(xn, b, i)
        xt = xn(2) + l*b
    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)  
  call bondl(1)
  if (i.eq.2) then
    call next_c2(xn, xt, i)
  else if (i.eq.3) then
    call next_c3(xn, xt, i)
  else if (i.ge.4) then
    call tors_bonda(xn, b, i)
    xt = xn(i-1) + l*b
  endif
  return
end

generate a trial position for ith atom
position of atoms (i - 1) are known
generate bond length
second atom
use Algorithm 26
third atom
use Algorithm 27
fourth and higher atoms
generate vector with prescribed
bond and torsional angles

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

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

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

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

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

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

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

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

and similarly, for the old conformation

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

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

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

and

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

The acceptance rule for the first scheme is

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

and for the second scheme is

$$\text{acc}(o \to n) = \min[1, W^{\text{ext}}(n)/W^{\text{ext}}(o)] = 1.$$
13.3 Generation of Trial Orientations

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-
Biased Monte Carlo Schemes

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

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\ell = 5$</th>
<th>$\ell = 10$</th>
<th>$\ell = 20$</th>
<th>$\ell = 40$</th>
<th>$\ell = 80$</th>
<th>$\ell = 160$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.6</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>50</td>
<td>10</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>10</td>
<td>64</td>
<td>58</td>
<td>53</td>
<td>42</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>20</td>
<td>72</td>
<td>66</td>
<td>60</td>
<td>56</td>
<td>44</td>
<td>0.1</td>
</tr>
<tr>
<td>40</td>
<td>80</td>
<td>72</td>
<td>67</td>
<td>62</td>
<td>57</td>
<td>40</td>
</tr>
<tr>
<td>80</td>
<td>83</td>
<td>78</td>
<td>72</td>
<td>68</td>
<td>62</td>
<td>60</td>
</tr>
</tbody>
</table>

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

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

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

Vlugt [360] pointed out that, because of the term $u_{\text{bend}}(b_A, c_2, b_B)$, 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

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

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

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

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

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

\[
P(b_a) = \int db_BP(b_A, b_B) \\
\propto \exp \left\{ -\beta u_{\text{bend}}(c_1, c_2, b_A) \right\} \\
\times \int db_B \exp \left\{ -\beta \left[u_{\text{bend}}(c_1, c_2, b_A) + u_{\text{bend}}(b_A, c_2, b_B)\right]\right\}.
\]

For the sequential scheme, we would have obtained

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

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 \( a \text{ priori} \) 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
ignoring the bias introduced by the “sequential” scheme will result in small, but noticeable, errors in the distribution of the bond angles.

As the insertion of two segments at the same time is less efficient than sequential insertion, several strategies have been proposed to increase the efficiency of the simultaneous generation of branches.

For molecules in which the bond-bending potential has three contributions (as in the example above), the simplest scheme is to generate two random vectors on a sphere and use the conventional rejection scheme to generate configurations with a probability proportional to their Boltzmann weight [362]. One can also use this approach for more complex potentials that include torsion. If the random generation of trial directions becomes inefficient, it may be replaced by a simple Monte Carlo scheme [361].

For some intramolecular potential it may even be necessary to add more than two atoms at the same time to ensure a proper a priori distribution of added segments. In fact, for some molecules that have multiple torsional angles, such as 2,3-dimethylbutane, this approach would imply that all atoms have to be added at the same time. To avoid such many-particle insertions, Martin and Siepmann [363] developed a scheme similar to the multiple-first-bead algorithm (see section 13.5).

The idea is to use a random insertion to generate several trial positions and to use a CBMC scheme to select acceptable candidates using the internal energies only. These configurations that are distributed according to the correct intramolecular Boltzmann weight will subsequently be used in another CBMC scheme that involves the more expensive external energy calculations.

To see how this approach works, assume that we have a model with internal interactions given by $u^{\text{int}}$. A single segment is added using the following steps:

1. First generate a set of $n_{\text{int}}$ random trial positions and for each position compute the internal energy, $U^{\text{int}}(i)$, and calculate the Rosenbluth factor associated with this internal energy

$$W^{\text{int}}(n) = \sum_{j=1}^{n_{\text{int}}} \exp \left[-\beta U^{\text{int}}(j)\right].$$

A possible orientation is then selected using

$$p^{\text{int}}(j) = \frac{\exp \left[-\beta U^{\text{int}}(j)\right]}{W^{\text{int}}(n)}.$$  

2. Step 1 is repeated to generate $k$ trial positions which are then fed into the conventional CBMC scheme to compute the Rosenbluth factor using the external potential $W^{\text{ext}}(n)$. 
3. A similar method is used for the old configuration, giving $W^{\text{int}}(o)$ and $W^{\text{ext}}(o)$.

4. A move is accepted using

$$\text{acc}(o \rightarrow n) = \min\left(1, \frac{W^{\text{int}}(n)W^{\text{ext}}(n)}{W^{\text{int}}(o)W^{\text{ext}}(o)}\right).$$

Depending on the details of the potential, further refinements are possible. One can, for instance, separate the bond-bending potential and the torsion potential. This would imply three nested CBMC steps giving three different Rosenbluth factors. For more details see [363].

### 13.4 Fixed Endpoints

A drawback of the conventional configurational-bias Monte Carlo scheme is that it regrows a chain molecule, either partly or completely, starting from one of the endpoints. For dense systems, where only relatively short segments of the molecule can be regrown successfully, the configurational-bias Monte Carlo scheme reduces to the reptation scheme. This implies that the equilibration of the middle segments of a chain proceeds very slowly—for heteropolymers, where reptation moves are forbidden, the situation is even worse. The same restriction applies to chain molecules that have either end rigidly anchored to a surface. Finally, conventional configurational-bias Monte Carlo cannot be applied at all to ring polymers.

In the present section, we discuss how the configurational-bias Monte Carlo scheme can be extended to include sampling of chain conformations with fixed endpoints. With such a scheme it is possible to relax the interior of a chain as efficiently as the endpoints. Ring polymers can be considered special examples of chain molecules with fixed endpoints. Another interesting example that can be treated in the same way is the sampling of path integrals [364], but this falls outside the scope of this book. In addition, we discuss some alternative Monte Carlo techniques, such as concerted rotations and end-bridging Monte Carlo, which have been developed by Theodorou and co-workers [365].

#### 13.4.1 Lattice Models

Let us first consider configurational-bias Monte Carlo between fixed endpoints for a chain molecule on a simple cubic lattice. If we remove $n$ segments of the molecule between two fixed endpoints $r_1$ and $r_2$, we cannot simply regrow the molecule by the normal Rosenbluth scheme, because this does not ensure that a trial conformation starting at $r_1$ will end at $r_2$. Clearly,
we must bias our regrowth scheme in such a way that the trial conformation is forced to terminate at \( r_2 \). To achieve this, we use the following scheme. Suppose that we start our regrowth at position \( r_1 \). On a three-dimensional lattice, this coordinate is represented by three integer coordinates \( \{k_1, l_1, m_1\} \). The final position is denoted by \( \{k_2, l_2, m_2\} \). The total number of ideal (i.e., nonself-avoiding) random walks of length \( n \) between \( r_1 \) and \( r_2 \) is denoted by \( \Omega(r_1, r_2; n) \). We can always compute the number of ideal random walks between fixed endpoints analytically as it is simply a finite sum of multinomial coefficients [366, 367]. Let us next consider the growth of one segment, starting at \( r_1 \). In the original configurational-bias Monte Carlo scheme, we would consider all \( k \) possible trial directions. And we would select one of these directions, say direction \( j \), with a probability

\[
P(j) = \frac{\exp[-\beta u^{\text{ext}}(j)]}{\sum_{j'}^{k} \exp[-\beta u^{\text{ext}}(j')]},
\]

where \( u^{\text{ext}}(j) \) denotes the potential energy of trial segment \( j \) due to all other particles already in the system. In the present case, we use a different weight factor to select the trial segment, namely,

\[
P(j) = \frac{\exp[-\beta u^{\text{ext}}(j)]\Omega(r_1 + \Delta r(j), r_2; n - 1)}{\sum_{j'}^{k} \exp[-\beta u^{\text{ext}}(j')]\Omega(r_1 + \Delta r(j'), r_2; n - 1)}. \tag{13.4.1}
\]

In other words, the probability of selecting a given trial direction is proportional to the number of ideal random walks of length \( n - 1 \) that start at the position of the trial segment and terminate at \( r_2 \). In this way, we guarantee that we generate only conformations that start at \( r_1 \) and terminate at \( r_2 \). However, as before, we must correct for the bias that we have introduced. We do this by constructing a modified Rosenbluth weight \( W \): \( W = \prod_{i=1}^{n} w_i \) with

\[
w_i = \frac{\sum_{j'=1}^{k} \exp[-\beta u^{\text{ext}}(j')]\Omega(r_i + \Delta r(j'), r_2; n - i)}{\sum_{j'=1}^{k} \exp[-\beta u^{\text{ext}}(j')]\Omega(r_i + \Delta r(j'), r_2; n - i)} \tag{13.4.2}
\]

If we now multiply the probability of generating a given trial conformation \( \Gamma \) with the Rosenbluth weight of that conformation, we find that

\[
P_{\text{gen}}(\Gamma) \times W(\Gamma) = \prod_{i=1}^{n} \left\{ \frac{\exp[-\beta u^{\text{ext}}(j)]\Omega(r_i + \Delta r(j), r_2; n - i)}{\sum_{j'=1}^{k} \exp[-\beta u^{\text{ext}}(j')]\Omega(r_i + \Delta r(j'), r_2; n - i)} \times \frac{\sum_{j'=1}^{k} \exp[-\beta u^{\text{ext}}(j')]\Omega(r_i + \Delta r(j'), r_2; n - i)}{\Omega(r_i, r_2; n - i + 1)} \right\} \tag{13.4.3}
\]
13.4 Fixed Endpoints

The modified Rosenbluth weight has been chosen such that all but one of the factors involving the number of ideal conformations cancel each other:

$$P_{\text{gen}}(\Gamma) \times W(\Gamma) = \prod_{i=1}^{n} \frac{\exp[-\beta u^{\text{ext}}(i)]}{\Omega(r_{1}, r_{2}; n)} = \frac{\exp[-\beta u^{\text{ext}}(\Gamma)]}{\Omega(r_{1}, r_{2}; n)}. \quad (13.4.4)$$

The remaining factor $\Omega$ is the same for all conformations of length $n$ that start at $r_{1}$ and terminate at $r_{2}$; hence, it drops out when we compute the relative probabilities of the old and new conformations. As before, the actual Monte Carlo scheme involves generating the trial conformation using the scheme indicated in equation (13.4.1) and accepting the new conformation with a probability given by

$$\text{acc}(o \rightarrow n) = \min \left[1, \frac{W(n)}{W(o)}\right]. \quad (13.4.5)$$

A total regrowth of a ring polymer of length $\ell$ can be accomplished by choosing $r_{1} = r_{2}$ and $n = \ell$.

13.4.2 Fully Flexible Chain

Again, it is possible to extend configurational-bias Monte Carlo to sample chain conformations between fixed endpoints, using our knowledge of the exact expression for the number (or, more precisely, the probability density) of ideal (nonself-avoiding) conformations of $n$ segments between fixed endpoints $r_{1}$ and $r_{2}$. If we denote the probability density to find segment $i + 1$ at a distance $r$ from segment $i$ by $p_{1}(r)$, then we have the following recursion relation between the probability density of the end-to-end separation of chains of length $n$ and $n + 1$:

$$P(r_{12}; n + 1) = \int d\Delta P(r_{12} - \Delta; n) p_{1}(\Delta). \quad (13.4.6)$$

From equation (13.4.6) and the fact that $p_{1}(r)$ is normalized, we immediately deduce the inverse relation:

$$P(r_{12}; n) = \int d\Delta P(r_{12} + \Delta; n + 1). \quad (13.4.7)$$

In the special case that all segments are of fixed length $a$, the expression for this probability density is [368]

$$P(r_{12}; n) = \sum_{k=0}^{n} \binom{n-r_{12}/a}{n-k} \binom{n}{k} (n-2k-r_{12}/a)^{n-2} \frac{2^{n+1}(n-2)!\pi a^2 r_{12}}{2^{n+1}(n-2)!}, \quad (13.4.8)$$

where $r_{12} \equiv |r_{1} - r_{2}|$. This expression is valid for all $n > 1$. As before, we wish to modify the configurational-bias Monte Carlo sampling of conformations of a fully flexible chain in such a way that the chain is forced to
terminate at \( r_2 \). There are two ways to do this. In one approach, we include the bias in the probability with which we generate trial directions; in the second, the bias is in the acceptance probability. In either case, our approach does not depend on the specific form of \( p_1(\mathbf{r}) \), but only on the existence of the recurrence relation (13.4.7).

In the first approach, we use the following scheme of generating the \( i \)th segment out of \( \ell \) segments to be regrown. We generate \( k \) trial segments, all starting at the current trial position \( \mathbf{r} \), such that the \textit{a priori} probability of generating a given trial direction (say, \( \mathbf{r}_j \)) is proportional to the probability of having an ideal chain conformation of length \( \ell - i \) between this trial segment and the final position \( r_2 \). Let us denote this \textit{a priori} probability by \( p_{\text{bond}}(\mathbf{r}_j) \). By construction, \( p_{\text{bond}}(\mathbf{r}_j) \) is normalized. Using equation (13.4.7) we can easily derive an explicit expression for \( p_{\text{bond}}(\mathbf{r}) \):

\[
\frac{p_{\text{bond}}(\mathbf{r})}{p_{\text{bond}}(\mathbf{r}_j)} = \frac{p_1(\mathbf{r}) p(r + \mathbf{r} - r_2; \ell - i)}{p_1(\mathbf{r}_j) p(r + \mathbf{r} - r_2; \ell - i - 1)} \frac{p_1(\mathbf{r}) p(r + \mathbf{r} - r_2; \ell - i)}{p(r - r_2; \ell - i + 1)}. \tag{13.4.9}
\]

From here on, we treat the problem just like the sampling of a continuously deformable chain, described in section 13.2.3. That is, we select one of the \( k \) trial directions with a probability

\[
p_{\text{sel}}(j) = \frac{\exp[-\beta u^{\text{ext}}(\mathbf{r}_j)]}{\sum_{j' = 1}^{k} \exp[-\beta u^{\text{ext}}(\mathbf{r}_{j'})]}.
\]

The contribution to the total Rosenbluth weight of the set of \( k \) trial directions generated in step \( i \) is

\[
w_i = \frac{\sum_{j' = 1}^{k} \exp[-\beta u^{\text{ext}}(\mathbf{r}_{j'})]}{k}.
\]

The overall probability of moving from the old conformation \( \Gamma_{\text{old}} \) to a new conformation \( \Gamma_{\text{new}} \) is proportional to the product of the probability of generating the new conformation and the ratio of the new to the old Rosenbluth weights. The condition of (super-)detailed balance requires that the product of the probability of generating the new conformation times the Rosenbluth weight of that conformation is (but for a factor that is the same for the old and new conformations) equal to the product of the Boltzmann weight of that conformation and the properly normalized probability of generating the corresponding ideal (i.e., noninteracting) conformation. If we write the
expression for this product, we find that

\[
\prod_{i=1}^{\ell} P_{\text{gen}}[\Gamma_i(i)] w_i = \left( \frac{p_1(r_1 - r_{i-1}) P(r_1 - r_2; \ell - i)}{P(r_{i-1} - r_2; \ell - i + 1)} \right) \frac{\exp\left[-\beta u_{\text{ext}}[\Gamma_i(i)]\right]}{\sum_{j'=1}^{k} \exp\left[-\beta u_{\text{ext}}[\Gamma_{j'}(i)]\right]} \times \left( \frac{\sum_{j'=1}^{k} \exp\left[-\beta u_{\text{ext}}[\Gamma_{j'}(i)]\right]}{k} \right) \exp\left[-\beta U_{\text{ext}}(\mathbf{r}_{\text{total}})\right] \prod_{i=1}^{\ell} p_1(r_i - r_{i-1}) \frac{k^\ell P(r_1 r_2; \ell)}{k^\ell P(r_1 r_2; \ell)}. \tag{13.4.10}
\]

As the last line of this equation shows, the conformations are indeed generated with the correct statistical weight. In ref. [369] this scheme has been applied to simulate model homopolymers, random heteropolymers, and random copolymers consisting of up to 1000 Lennard-Jones beads. For molecules with strong intramolecular interactions, the present scheme will not work and other approaches are needed.

### 13.4.3 Strong Intramolecular Interactions

In the previous section we have shown that we can use the configurational-bias Monte Carlo scheme to grow a chain of length \(n\) between two fixed endpoints \(r_1\) and \(r_2\) if we know the probability density of conformations of length \(n\) between these points. For the special case of a fully flexible chain this probability distribution is known analytically. For chains with strong intramolecular interactions such an analytical distribution is not known. Wick and Siepmann [370] and Chen and Escobedo [371] have shown that one can use an approximated distribution. Chen and Escobedo [371] estimate this distribution using a simulation of an isolated chain with bonded interactions only. Wick and Siepmann [370] proposed a scheme in which this estimated probability distribution is further refined during the simulation.

### 13.4.4 Rebridging Monte Carlo

If we model a realistic polymer or peptide we have to include bond-bending and torsional potentials. Suppose that we rotate in the interior of a polymer a randomly selected torsional angle by an amount \(\Delta \phi\). If we would keep all other torsional angles of the remainder of the chain fixed, a tiny change of this torsional angle would lead to a large displacement of the last atom of the chain. If, on the other hand, one would only displace the neighboring
atoms, the intramolecular interactions of the chain would increase significantly, again limiting the maximum rotation. We would like to ensure that the rotation affects only a small part of the interior of the chain and that it results in a redistribution of atoms that does not result in a large increase in the intramolecular energy. **Concerted rotation and rebridging** and **end-bridging** Monte Carlo are schemes that have been developed by Theodorou and co-workers [60,365,372] to perform such Monte Carlo moves.

In Figure 13.8 the rebridging problem is sketched schematically. Suppose we give the atoms 1 and 5 a new position by a random rotation of the driver angles $\phi_0$ and $\phi_7$. Assume that all bond lengths and bond angles have a prescribed value, for example, their equilibrium value or any other specified value. The rebridging problem is to find all possible conformations of the trimer consisting of the atoms 2, 3, and 4 that rebridge the new positions of atoms 1 and 5 given the constraints of the prescribed bond lengths and angles. Wu and Deem [373] have shown that for the rebridging problem an analytical solution exists and that the maximum number of solutions is strictly limited to 16. Alternatively, in refs. [365,372] it is shown how to numerically locate all solutions of the rebridging problem.

Suppose that we have all solutions of the rebridging problem, either by the analytical solution of Wu and Deem or by the numerical scheme of Theodorou and co-workers. The next step is to use this in a Monte Carlo scheme. The scheme that we discuss here is only valid for the interior segments of a polymer. For the ends of the chains, a slightly different scheme has to be used [60]:

1. The present conformation of the polymer is denoted by $\sigma$. We gener-
ate the new configuration of the polymer, \( n \), by selecting an atom and a direction (forward or backward) at random. This defines the atoms pair 1 and 5. These atoms are given new positions \( 1' \) and \( 5' \) by performing a random rotation around bonds \(-1,0\) and \( 6,7\), respectively (see Figure 13.8).

2. Solve the rebridging problem to locate all possible conformations of the trimer that bridge the new positions of atoms \( 1' \) and \( 5' \). The total number of conformations is denoted by \( N_n \) and out of these we select one conformation, say \( n \), at random.\(^7\) If no such conformation is found the move is rejected.

3. For the old conformation, we also locate all possible conformation, i.e., we solve the rebridging problem to locate the conformations of the trimer that bridge the old positions of atoms 1 and 5. This number of conformation is denoted by \( N_o \).

4. In the rebridging scheme, we use a dihedral angle \( \phi \) to generate a new configuration. This implies a temporary change of coordinate system; a Jacobian is associated with this change. In general, this Jacobian is not equal to 1. This Jacobian has to be taken into account in the acceptance rules [60]. The equations for this Jacobian can be found in refs. [60,372,373]. Here, we assume that these determinants for the old and new conformation have been calculated and are denoted by \( J(o) \) and \( J(n) \), respectively.

5. Of the new and old conformations the energies are calculated, \( U(o) \) and \( U(n) \), respectively.

6. The new conformation is accepted with a probability

\[
\text{acc}(o \rightarrow n) = \min \left( 1, \frac{\exp[-\beta U(n)] J(n)/N_n}{\exp[-\beta U(o)] J(o)/N_o} \right).
\]

In refs. [60,372] the proof is given that this rebridging scheme obeys detailed balance and samples a Boltzmann distribution.

The reason it is important to find all solutions of the rebridging problem is to ensure detailed balance. Suppose that the determinants of the Jacobians are one and the energies are zero, then without the terms \( 1/N_n \) and \( 1/N_o \) the acceptance probability would be one for all possible new conformations. Suppose that we have a single solution for the new conformation, \( N_n = 1 \), and for the old conformation \( N_o = 2 \). Without the correction we would violate detailed balance since the \emph{a priori} probability of the reverse move is only a half. Pant and Theodorou [372] have developed an alternative scheme

\(^7\)One could use the configurational-bias Monte Carlo scheme as an alternative for the random selection.
in which one has to find only a single rebridging conformation, which is the first solution of their numerical scheme. To ensure detailed balance one has to check that the old conformation should also be the first solution to which the numerical scheme converges.

One can also use the rebridging scheme to connect atoms of different chains. The idea of end-bridging Monte Carlo is to alter the connectivity of the chain by bridging atoms from different chains. The simplest form is to rebridge a chain end to an interior segment of another chain with a trimer. Such an end-bridging Monte Carlo move induces a very large jump in configuration space. An important aspect of such an end-bridging move is, however, that it alters the chain lengths of the two chains. Therefore, such a move cannot be used if it is important to keep the chain length fixed. However, in most practical applications of polymers one does not have a single chain length but a distribution of chain lengths. Pant and Theodorou [372] have shown that the resulting chain length distribution resembles a truncated Gaussian distribution.

One can envision a chain length distribution as a mixture of a very large number of components, each component characterized by its chain length $l$. Imposing the chain length distribution is equivalent to imposing the chemical potentials of the various components. This suggests that we could combine these end-bridging moves with the semigrand ensemble simulation technique (see section 9.1) to determine whether a change of the polymer length should be accepted.

In principle one can use two rebridging moves for the interior segments of two chains. This would allow us to perform moves in which the total chain length remains constant. Whether in practice such a scheme will work depends on the probability that two segments of different chains with the same number of end segments connected to it are sufficiently close to each other.

Tests show that the rebridging method is very efficient for polymer melts with chain length up to $C_{30}$. For chains up to $C_{70}$ rebridging Monte Carlo still samples the local structure efficiently, but fails to sample chain characteristics at larger length scales such as the end-to-end vector. End-bridging Monte Carlo effectively relaxes chains up to $C_{500}$ [365]. Another important application of rebridging Monte Carlo is the possibility of simulating cyclic molecules. This application is illustrated by Wu and Deem in their study of cis/trans isomerisation of proline-containing cyclic peptides [373,374].

## 13.5 Beyond Polymers

Thus far, the configurational-bias scheme has been presented exclusively as a method of generating polymer conformations. The method is more general
than that. It can be used as a scheme to perform collective rearrangements of any set of labeled coordinates. In fact, the scheme can be used to carry out Monte Carlo moves to swap \( n \) small particles within a volume \( \Delta V \) with one large particle that occupies the same (excluded) volume. This application of the CBMC scheme has been exploited by Biben et al. [375, 376] to study mixtures of large and small hard spheres. Gibbs ensemble simulations of mixtures of spherical colloids and rodlike polymers were performed by Bolhuis and Frenkel [377] (see Example 18), using CBMC-style particle swaps and a closely related approach was employed by Dijkstra and co-workers to study phase separation [366, 367] in mixtures of large and small hard-core particles on a lattice. An application of CBMC for improving the sampling of ionic solutions has been proposed by Shelley and Patey [378].

A different application of the CBMC ideas is used by Esselink et al. [379] to develop an algorithm to perform Monte Carlo moves in parallel. Parallel Monte Carlo appears to be a contradiction in terms, since the Monte Carlo procedure is an intrinsically sequential process. One has to know whether the current move is accepted or rejected before one can continue with the next move. The conventional way of introducing parallelism is to distribute the energy calculation over various processors or to farm out the calculation by performing separate simulations over various processors. Although the last algorithm is extremely efficient and requires minimum skills to use a parallel computer, it is not a truly parallel algorithm. For example, farming out a calculation is not very efficient if the equilibration of the system takes a significant amount of CPU time. In the algorithm of Esselink et al. several trial positions are generated in parallel, and out of these trial positions the one with the highest probability of being accepted is selected. This selection step introduces a bias that is removed by adjusting the acceptance rules. The generation of each trial move, which includes the calculation of the energy (or Rosenbluth factor in the case of chain molecules), is distributed over the various processors. Loyens et al. have used this approach to perform phase equilibrium calculations in parallel using the Gibbs ensemble technique [380].

An interesting application of this parallel scheme is the multiple-first-bead algorithm. In a conventional CBMC simulation one would have to grow an entire chain before one can reject a configuration that is "doomed" from the start because the very first bead has an unfavorable energy. If the chains are long this can be inefficient and it becomes advantageous to use a multiple-first-bead scheme [379]. Instead of generating a single trial position for the first bead, \( k \) trial positions are generated. The energy of these beads, \( u_1(j) \) with \( j = 1, \ldots, k \), is calculated, and one of these beads, say \( j \), is selected using the Rosenbluth criterion:

\[
P_{1st}(j) = \frac{\exp[-\beta u_1(j)]}{w_1}
\]
362

Chapter 13. Biased Monte Carlo Schemes

where

\[ w_1(n) = \sum_{i=1}^{k} \exp[-\beta u_1(i)]. \]

Also for the old configuration one should use a similar scheme to compute \( w_1(o) \). For some moves the same set of first beads used for the new configuration can be used to compute the Rosenbluth factor for the old configuration [381]. To ensure detailed balance the Rosenbluth factors associated with the multiple-first beads should be taken into account in the acceptance rules:

\[ \text{acc}(o \rightarrow n) = \min \left( 1, \frac{w_1(n)W(n)}{w_1(o)W(o)} \right), \]

where \( W(n) \) and \( W(o) \) are the (conventional) Rosenbluth factors of the new and the old configuration of the chain, respectively, excluding the contribution of the first segment. Vlugt et al. [382] have shown that a multiple-first-bead move can increase the efficiency of simulations of n-alkanes up to a factor of 3.

Another extension of the CBMC principles is the use of a dual-cutoff radius [382]. The idea is that usually in a particular trial conformation is accepted not because it is energetically very favorable, but because its competitors are so unfavorable. This suggests that one can use a much cheaper potential to perform a prescreening of acceptable trial configurations in a CBMC move. Let us split the potential into a contribution that is cheap to compute and the expensive remainder:

\[ U(r) = U_{\text{cheap}}(r) + \Delta U(r). \]

This can be done, for example, by splitting the potential into a long-range and short-range part. We can now use the cheap part in our CBMC scheme to generate trial configurations. The probability of generating a given configuration is then

\[ p_{\text{cheap}}(n) = \frac{\exp[-\beta U_{\text{cheap}}(n)]}{W_{\text{cheap}}(n)}. \]

and the move is accepted using

\[ \text{acc}(o \rightarrow n) = \min \left( 1, \frac{W_{\text{cheap}}(n)}{W_{\text{cheap}}(o)} \exp[-\beta (\Delta U(n) - \Delta U(o))] \right). \]

In ref. [382] it is shown that this scheme obeys detailed balance. The advantage of this algorithm is that the expensive part of the energy calculation has to be performed only once and not for every trial segment. A typical application would be to include the Fourier part of an Ewald summation in \( \Delta U \). Many variations on this theme are possible.
Example 18 (Mixtures of Colloids and Polymers)

Configurational-bias Monte Carlo (CBMC) was presented as a scheme to sample conformations of chain molecules. In fact, the method is more general than that. It can be used to perform collective rearrangements of any set of labeled coordinates. For instance, the scheme can be used to carry out Monte Carlo moves to swap $n$ small particles within a volume $\Delta V$ with one large particle that occupies the same (excluded) volume. This application of the CBMC scheme has been exploited by Biben [375] to study mixtures of large and small hard spheres. Gibbs ensemble simulations of mixtures of spherical colloids and rodlike polymers were performed in ref. [377] using CBMC-style particle swaps, and a closely related approach was employed by Dijkstra et al. [366, 367] to study phase separation of mixtures of large and small hard-core particles on a lattice.

Below, we briefly discuss an example of such a CBMC scheme, related to the phase behavior of colloidal suspensions [377]. Examples of colloidal solutions are milk, paint, and mayonnaise. Since a single colloidal particle may contain more than $10^9$ atoms, it is not practical to model such a particle as a collection of atoms. It is better to describe colloidal solutions using coarse-grained models. For example, a suspension of sterically stabilized silica spheres in a nonpolar solvent can be described surprisingly accurately with a hard-sphere potential. Similar to the hard-sphere fluid, such a colloidal suspension has a “fluid-solid” transition, but not a “liquid-gas” transition. To be more precise, the colloidal particles undergo a transition from a liquid-like arrangement to a crystalline structure. But in either case, the solvent remains liquid. In what follows, we use the terms “crystal,” “liquid,” and “gas” to refer to the state of the colloidal particles in suspension. Experimentally, it is observed that a liquid-gas transition can be induced in a suspension of hard-sphere colloids by adding nonadsorbing polymers.

The addition of polymers induces an effective attraction between the colloidal particles. This attraction is not related to any change in the internal energy of the system, but to an increase in the entropy. It is not difficult to understand the origin of such entropic attractions. Let us assume that the polymers in solution do not interact with each other. This is never rigorously true, but for dilute solutions of long, thin molecules, it is a good first approximation. The translational entropy of $N$ polymers in a volume $V$ is then equal to that of $N$ ideal-gas molecules occupying the same volume: $S_{\text{trans}}^{(0)} = \text{constant} + Nk_B \ln V$, where the constant accounts for all those contributions that do not depend on the volume $V$. In the absence of colloids, the volume accessible to the polymers is equal to $V_0$, the volume of the container. Now suppose that we add one hard colloidal particle with radius $R_c$. As the polymers cannot penetrate the colloidal particle, such a colloid excludes the polymers from a spherical volume with radius $R_{\text{ext}} \equiv R_c + R_p$, where $R_p$ is the effective radius of the polymer (for flexible polymers, $R_p$ is
on the order of the radius of gyration, and for rigid polymers, $R_p$ is of order $O(L)$, where $L$ is the length of the polymer). Let us denote the volume excluded by one colloid by $v_{\text{excl}}$. Clearly, the entropy of $N$ polymers in the system that contains one colloid is $S_{\text{trans}}^{(1)} = \text{constant} + N k_B \ln (V_0 - v_{\text{excl}}^c)$. Now consider what happens if we have two colloidal spheres in the solution. Naively, one might think that the entropy of the polymer solution is now equal to $S_{\text{trans}}^{(2)} = \text{constant} + N k_B \ln (V_0 - 2v_{\text{excl}}^c)$. However, this is only true if the two colloids are far apart. If they are touching, their exclusion zones overlap, and the total excluded volume $v_{\text{excl}}^{\text{pair}}$ is less than $2v_{\text{excl}}^c$. This implies that the entropy of the polymers is larger when the colloids are touching than when they are far apart. Therefore, we can lower the free energy of the polymer solution by bringing the colloids close together. And this is the origin of entropic attraction. The strength of the attraction can be tuned by changing the polymer concentration and, for sufficiently high polymer concentrations, the colloidal suspensions may undergo a “liquid-vapor” phase separation.

In the present example, we consider the phase behavior of a mixture of colloidal hard spheres and thin hard rods [377]. In principle, we can use Gibbs ensemble simulations to study the “vapor-liquid” coexistence in this mixture. However, a conventional Gibbs ensemble simulation is likely to fail as the transfer of a colloidal sphere from one simulation box to the other will, almost certainly, result in an overlap of the sphere with some of the rodlike polymers. We can now use the CBMC approach to perform such a trial move with a higher chance of success. In this scheme, we perform the following steps:

1. Randomly select a sphere in one of the boxes, and insert this sphere at a random position in the other box.

2. Remove all the rods that overlap with this sphere. These rods are inserted in the other box. The positions and orientations of the rods are chosen such that they intersect with the volume vacated by the colloid—but apart from that, they are random. Even though we have thus ensured that the rods are in, or near, the “cavity” left by the colloidal sphere, they are very likely to overlap with one or more of the remaining spheres. However, if one tries several orientations and positions of the rods and selects an acceptable configuration using the configurational-bias Monte Carlo scheme, one can strongly enhance the acceptance probability of such particle swaps.

The results of these Gibbs ensemble simulations are presented in Figure 13.9. This figure shows that if one increases the fugacity (and thereby the concentration) of the rods, a demixing into a phase with a low density of spheres and a phase with a high density of spheres occurs. The longer the rods, the lower the concentration at which this demixing occurs. We
Figure 13.9: Coexistence curves for a mixture of hard spheres and thin rods [377]. The horizontal axis measures the density and the vertical axis the fugacity ($= \exp(\beta \mu)$). $L/\sigma$ is the ratio of the length of the rods to the diameter of the hard spheres.

stress once again that, in this system, only hard-core interactions between the particles exist. Therefore this demixing is driven by entropy alone.

\section*{13.6 Other Ensembles}

\subsection*{13.6.1 Grand-Canonical Ensemble}

In Chapter 5, we introduced the grand-canonical ensemble in the context of simulations of systems in open contact with a reservoir. An essential ingredient of Monte Carlo simulations in this ensemble is the random insertion or removal of particles. Clearly, such simulations will be efficient only if there is a reasonable acceptance probability of particle-insertion moves. In particular for polyatomic molecules, this is usually a problem. Let us consider the system mentioned in Example 2, a grand-canonical ensemble simulation of the adsorption of molecules in the pores of a microporous material such as a zeolite. For single atoms, the probability that we find an arbitrary position that does not overlap with one of the atoms in the zeolite lattice is on the order 1 in $10^3$. For dimers, we have to find two positions that do not overlap, and if we assume that these positions are independent, the probability of success will be 1 in $10^6$. Clearly, for the long-chain molecules, the probability of a successful insertion is so low that to obtain a reasonable number of accepted insertions, the number of attempts needs to be prohibitively large. In the
present section, we demonstrate how configurational-bias Monte Carlo tech-
nique can be used in the grand-canonical ensemble to make the exchange step of chain molecules more probable.

Algorithm

As in the general scheme of the configurational-bias Monte Carlo technique for off-lattice systems, we divide the potential energy of a given confor-
mation into a bonded potential energy \( U_{\text{bond}} \), which includes the local intramolecular interactions, and an \textit{external} potential energy \( U_{\text{ext}} \), which in-
cludes the intermolecular interactions and the nonbonded intramolecular interactions (see section 13.2.3). A chain that has only bonded interactions is
defined as an ideal chain. Let us now consider the Monte Carlo trial moves for the insertion and removal of particles.

Particle Insertion To insert a particle into the system, we use the following steps:

1. For the first monomer, a random position is selected, and the energy of this monomer is calculated. This energy is denoted by \( u_{1}^{\text{ext}}(n) \) and we
define \( w_{1}^{\text{ext}}(n) = k \exp[-\beta u_{1}^{\text{ext}}(n)] \) (as before, the factor \( k \) is introduced only to simplify the subsequent notation).

2. For the following monomers, a set of \( k \) trial positions is generated. We denote these positions by \( \{b\}_k = (b_1, b_2, \cdots, b_k) \). This set of trial orientations is generated using the bonded part of the potential, which results in the following distribution for the \( i \)th monomer:

\[
p_{i}^{\text{bond}}(b)db = C \exp[-\beta u_{i}^{\text{bond}}(b)]db
\]

with

\[
C^{-1} = \int db \exp[-\beta u_{i}^{\text{bond}}(b)].
\]

Note that the way the trial orientations are generated depends on the type of monomer being added (see section 13.3). For each of these trial positions the external energy, \( u_{i}^{\text{ext}}(b_j) \), is calculated, and one of these positions is selected with a probability

\[
p_{i}^{\text{ext}}(b_n) = \frac{\exp[-\beta u_{i}^{\text{ext}}(b_n)]}{w_{i}^{\text{ext}}(n)},
\]

in which

\[
w_{i}^{\text{ext}}(n) = \sum_{j=1}^{k} \exp[-\beta u_{i}^{\text{ext}}(b_j)].
\]
3. Step 2 is repeated until the entire alkane of length $\ell$ has been grown, and the normalized Rosenbluth factor can be calculated:

$$\mathcal{W}_{\text{ext}}^{\ell}(n) = \frac{\mathcal{W}_{\text{ext}}^{\ell}(n)}{k^\ell} = \prod_{i=1}^{\ell} \frac{\mathcal{W}_{\text{ext}}^{\ell}(n)}{k}. \quad (13.6.4)$$

4. The new molecule is accepted with a probability

$$\text{acc}(N \rightarrow N+1) = \min \left( 1, \frac{q(T) \exp(\beta \mu^B) V}{(N+1)} \mathcal{W}_{\text{ext}}^{n}(n) \right), \quad (13.6.5)$$

where $\mu^B$ is the chemical potential of a reservoir consisting of ideal chain molecules and $q(T)$ is the kinetic contribution to the molecular partition function (for atoms, $q(T) = 1/\Lambda^3$).

**Particle Removal** To remove a particle from the system, we use the following algorithm:

1. A particle, say, o, is selected at random, the energy of the first monomer is calculated and is denoted by $u_1^{\text{ext}}(o)$, and we determine $w_1^{\text{ext}}(o) = k \exp[-\beta u_1^{\text{ext}}(o)]$.

2. For the following segments of the chain, the external energy $u_i^{\text{ext}}(o)$ is calculated and a set of $k-1$ trial orientations is generated with a probability given by equation (13.6.1). Using this set of orientations and the actual position, we calculate for monomer $i$:

$$w_i^{\text{ext}}(o) = \exp[-\beta u_i^{\text{ext}}(o)] + \sum_{j=2}^{k-1} \exp[-\beta u_i^{\text{ext}}(b_j)].$$

3. After step 2 is repeated for all $\ell$ monomers and we compute for the entire molecule:

$$\mathcal{W}^{\text{ext}}(o) = \frac{\mathcal{W}^{\text{ext}}(o)}{k^\ell} = \prod_{i=1}^{M} \frac{\mathcal{W}^{\text{ext}}(o)}{k}. \quad (13.6.6)$$

4. The selected molecule is removed with a probability

$$\text{acc}(N \rightarrow N-1) = \min \left( 1, \frac{N}{q(T)V\exp(\beta \mu^B)} \frac{1}{\mathcal{W}^{\text{ext}}(o)} \right). \quad (13.6.7)$$
We have defined $\mu^B$ as the chemical potential of a reservoir consisting of ideal chains. It is often convenient to use as a reference state the ideal gas of nonideal chains (i.e., chains that have both bonded and nonbonded intramolecular interactions). This results in a simple, temperature-dependent shift of the chemical potential:

$$\beta \mu^B \equiv \beta \mu_{\text{id.chain}} = \beta \mu_{\text{nonid.chain}} + \ln \langle \mathcal{W}^{\text{nonbonded}} \rangle,$$

(13.6.8)

where $\langle \mathcal{W}^{\text{nonbonded}} \rangle$ is the average Rosenbluth factor due to the nonbonded intramolecular interactions. This Rosenbluth factor has to be determined in a separate simulation of a single chain molecule. For more details about reference states, see Appendix G. In the same appendix, we also discuss the relation between the chemical potential and the imposed pressure (the latter quantity is needed when comparing with real experimental data). To show that the preceding algorithm does indeed yield the correct distribution, we have to demonstrate, as before, that detailed balance is satisfied. As the proof is very similar to those shown before, we will not reproduce it here. For more details, the reader is referred to [304].

**Example 19 (Adsorption of Alkanes in Zeolites)**

In Example 2 grand-canonical simulations were used to determine the adsorption of methane in the zeolite silicalite. Using the scheme described in the present section, Smit and Maesen computed adsorption isotherms of the longer alkanes [383]. Adsorption isotherms are of interest since they may signal phase transitions, such as capillary condensation or wetting, of the fluid inside the pores [384]. Capillary condensation usually shows up as a step or rapid variation in the adsorption isotherm. It is often accompanied by hysteresis, but not always; for instance, experiments on flat substrates [385] found evidence for steps in the adsorption isotherm without noticeable hysteresis.

Since the pores of most zeolites are of molecular dimensions, adsorbed alkane molecules behave like a one-dimensional fluid. In a true one-dimensional system, phase transitions are not expected to occur. To the extent that zeolites behave as a one-dimensional medium, one therefore might expect that the adsorption isotherms of alkanes in zeolites exhibit no steps. If steps occur, they are usually attributed to capillary condensation in the exterior secondary pore system formed by the space between different crystals. For silicalite, adsorption isotherms have been determined for various $n$-alkanes, and, indeed, for the short-chain alkanes (methane–pentane) the isotherms exhibit no steps. The same holds for decane. For hexane and heptane, however, steplike features are observed (for experimental details, see [383]).

In the simulations of Smit and Maesen [383] the alkane molecules are modeled with a united atom model; that is, $\text{CH}_3$ and $\text{CH}_2$ groups are considered as single interaction centers [386]. The zeolite is modeled as a rigid
13.6 Other Ensembles

Figure 13.10: Adsorption isotherms of butane (left) and heptane (right); the closed symbols are experimental data and the open symbols the results from simulations at $T = 298$ K.

crystal and the zeolite-alkane interactions are assumed to be dominated by the interaction with the oxygen atoms and are described by a Lennard-Jones potential.

Figure 13.10 compares the simulated adsorption isotherms of various alkanes in silicalite with experimental data. For butane, a smooth isotherm is observed and the agreement between experiments and simulation is good. For hexane and heptane, the agreement is good at high pressures but at low pressures deviations indicate that the zeolite-alkane model may need to be refined. It is interesting to note that, for heptane, both the experiments and the simulations show a step at approximately half the loading. Since the simulations are performed on a perfect single crystal, this behavior must be due to a transition of the fluid inside the pores and cannot be attributed to the secondary pore system.

Silicalite has two types of channels, straight and zigzag, which are connected via intersections. It so happens that the length of a hexane molecule is on the order of the length of the period of the zigzag channel. The simulations show that, at low chemical potential, the hexane molecules move freely in these channels and the molecules will spend part of their time at the intersections. If a fraction of the intersections is occupied, other molecules cannot reside in the straight channels at the same time. At high pressures, almost all hexane molecules fit exactly into the zigzag channel. They no longer move freely and keep their noses and tails out of the intersection. In such a configuration the entire straight channel can now be tightly packed with hexane molecules. This may explain the plateau in the adsorption isotherm; to fill the entire zeolite structure neatly, the hexane molecules located in zigzag channels first have to be "frozen" in these channels. This "freezing" of the
positions of the hexane molecules implies a loss of entropy and therefore will occur only if the pressure (or chemical potential) is sufficiently high to compensate for this loss. This also makes it clear why we do not observe a step for molecules shorter or longer than hexane or heptane. If the molecules are longer, they will always be partly in the intersection and nothing can be gained by a collective freezing in the zigzag channels. If the molecules are shorter than one period of the zigzag channel, a single molecule will not occupy an entire period and a second molecule will enter, which results in a different type of packing. The interesting aspect is that after the simulations were published this observation has been confirmed by experiments [387].

Also the adsorption behavior of mixtures of hydrocarbons has many surprising effects [361, 388].

13.6.2 Gibbs Ensemble Simulations

In Chapter 8, the Gibbs ensemble technique was introduced as an efficient tool for simulating vapor-liquid phase equilibria. One of the Monte Carlo steps in the Gibbs ensemble technique is the transfer of molecules between the liquid phase and gas phase. For long-chain molecules, this step, if carried out completely randomly, results in a prohibitively low acceptance of particle exchanges. Therefore, the Gibbs ensemble technique used to be limited to systems containing atoms or small molecules. However, by combining the Gibbs ensemble method with configurational-bias Monte Carlo, the method can be made to work for much longer chain molecules.

Algorithm

Let us consider a continuum system with strong intramolecular interactions. In section 13.2.3 it is shown that for such a system it is convenient to separate the potential energy into two contributions: the bonded intramolecular energy \( U_{\text{bond}} \) and the "external" energy \( U_{\text{ext}} \) that contains the intermolecular interactions and the nonbonded intramolecular interactions. As in the original implementation of the Gibbs ensemble, we attempt to exchange a molecule between the two boxes. However, while in section 8.1 the molecules were inserted at random, we now use the following procedure to grow a molecule atom by atom in a randomly selected box. Let us assume this is box \( I \) with volume \( V_1 \). The number of particles in this box is denoted by \( n_1 \).

1. The first atom is inserted at a random position, and the (external) energy \( u_{1,\text{ext}}(n) \) is calculated together with

\[
\omega_{1,\text{ext}}(n) = k \exp[-\beta u_{1,\text{ext}}(n)]. \tag{13.6.9}
\]
2. To insert the next atom $i$, $k$ trial orientations are generated. The set of $k$ trial orientations is denoted by $\{b\}_k = b_1, b_2, \ldots, b_k$. These orientations are not generated at random but with a probability that is a function of the bonded part of the intramolecular energy:

$$p_{\text{bond}}^b(b_n) = C \exp[-\beta u_{\text{bond}}^b(b_n)].$$

(13.6.10)

Of each of these trial orientations the external energy is calculated $u_i^{\text{ext}}(b_j)$ together with the factor

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

(13.6.11)

Out of these $k$ trial positions, we select one with probability

$$p_i^{\text{ext}}(b_n) = \frac{\exp[-\beta u_i^{\text{ext}}(b_i)]}{w_i^{\text{ext}}(n)}.$$  

(13.6.12)

3. Step 2 is repeated $\ell - 1$ times until the entire molecule is grown and the Rosenbluth factor of the molecule can be calculated:

$$W_i^{\text{ext}}(n) = \prod_{i=1}^\ell w_i^{\text{ext}}(n).$$

(13.6.13)

For the other box, we select a molecule at random and we determine its Rosenbluth factor, using the following procedure:

1. A particle is selected at random.
2. The (external) energy of the first atom is determined $u_i^{\text{ext}}(o)$ together with

$$w_i^{\text{ext}}(o) = k \exp[-\beta u_i^{\text{ext}}(o)].$$

(13.6.14)

3. For the next atom, $k - 1$ trial orientations are generated with a probability given by equation (13.6.10). These trial orientations, together with the actual position of atom $i$ ($b_o$), form the set $\{b'_j\}_k$ for which we determine the factor

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

(13.6.15)

4. Step 2 is repeated $\ell - 1$ times until we have retraced the entire chain and its Rosenbluth factor can be calculated:

$$W_{i}^{\text{ext}}(o) = \prod_{l=1}^\ell w_i^{\text{ext}}(o).$$

(13.6.16)
We then accept this move with probability

$$\text{acc}(o \rightarrow n) = \min \left(1, \frac{V_1(N-n_1)}{(V-V_1)(n_1+1)} \frac{W^{\text{ext}}(n)}{W^{\text{ext}}(o)} \right).$$  \hfill (13.6.17)

The proof of the validity of this algorithm, again, is very similar to those shown earlier in this chapter. We therefore refer the interested reader to [356,389,390]. The combination of the Gibbs ensemble technique with the configurational-bias Monte Carlo method has been used to determine the vapor-liquid coexistence curve of chains of Lennard-Jones beads [356, 389] and alkanes [386,390–392]. In Example 20, an application of this method is described.

Example 20 (Critical Properties of Alkanes)

Alkanes are thermally unstable above approximately 650 K, which makes experimental determination of the critical point of alkanes longer than decane (C\(_{10}\)) extremely difficult. The longer alkanes, however, are present in mixtures of practical importance for the petrochemical industry. In these mixtures, the number of components can be so large that it is not practical to determine all phase diagrams experimentally. One therefore has to rely on predictions made by equations of state. The parameters of these equations of state are directly related to the critical properties of the pure components. Therefore, the critical properties of the long-chain alkanes are used in the design of petrochemical processes, even if they are unstable close to the critical point [393]. Unfortunately, experimental data are scarce and contradictory, and one has to rely on semi-empirical methods to estimate the critical properties [393].

Siepmann et al. [386,390] have used the combination of the Gibbs ensemble technique and configurational-bias Monte Carlo to simulate vapor-liquid equilibria of the \(n\)-alkanes under conditions where experiments are not (yet) feasible. Phase diagrams are very sensitive to the choice of interaction potentials. Most available models for alkanes have been obtained by fitting simulation data to experimental properties of the liquid under standard conditions. In Figure 13.11 the vapor-liquid curve of octane as predicted by some of these models is compared with experimental data. This figure shows that the models of [394,395], which give nearly identical liquid properties, yield estimates of the critical temperature of octane that differ by 100 K. Siepmann et al. [386,390] used these vapor-liquid equilibrium data to improve the existing models.

In Figure 13.12 the critical temperatures and densities as predicted by the model of Siepmann et al. are plotted versus the carbon number. The simulations reproduce the experimental critical points very well. There is considerable disagreement, however, between the various experimental estimates of the critical densities. Much of our current knowledge of the critical
properties of the higher alkanes is based on extrapolations of fits of the experimental data up to C₈. The most commonly used extrapolations assume that the critical density is a monotonically increasing function of the carbon number, approaching a limiting value for the very long alkanes [393, 396]. In contrast to these expectations, the experimental data of Anselme et al. [397] indicate that the critical density has a maximum for C₈ and then decreases.
monotonically. The data of Steele (as reported in [396]), however, do not give any evidence for such a maximum (see Figure 13.12). The simulations indicate the same trend as that observed by Anselme et al. In this context, it is interesting to note that Mooij et al. [356], Sheng et al. [398], and Escobedo and de Pablo [399] used Monte Carlo simulations to study the vapor-liquid curve of a polymeric bead-spring model for various chain lengths. These studies also show a decrease of the critical density as a function of chain length. Such a decrease of the critical density with chain length is a general feature of long-chain molecules, as was already pointed out by Flory.

The Gibbs ensemble technique makes it possible to compute efficiently the liquid-vapor coexistence curve of realistic models for molecular fluids. This makes it possible to optimize the parameters of the model to yield an accurate description of the entire coexistence curve, rather than of a single state point. It is likely, but not inevitable, that a model that describes the phase behavior correctly, will also yield reasonable estimates of other properties, such as viscosity or diffusivity. Mondello and Grest have shown that this is indeed true for the diffusion coefficient of linear hydrocarbons [400, 401], while Cochran, Cummings, and co-workers [402, 403] found the same for the viscosity. The hydrocarbon model that was used in these studies had been optimized to reproduce experimental vapor-liquid coexistence data [386, 390]. Improved force fields have since been proposed for linear alkanes [358, 404, 405], branched alkanes [363], alkenes [406, 407], alkylbenzenes [406], and alcohols [408, 409].

13.7 Recoil Growth

To find numerical schemes that are more efficient than conformational-bias Monte Carlo (CBMC), we should first understand why CBMC works better than a scheme that employs random trial moves. Suppose that we have a system with hard-core interactions and the probability of successfully inserting a monomer is \( a \). If we assume that the insertion of an \( m \)-mer is equivalent to inserting \( m \) independent monomers, then the probability of a successful random insertion of an \( n \)-mer is

\[
p_m^{\text{random}} \approx a^m.
\]

For a dense system, \( a \ll 1 \), and therefore random insertion only works for very short chains. With the CBMC scheme we try \( k \) trial orientations and our growing scheme fails if all of the \( k \) trial orientations result in an overlap. The probability that we grow a chain successfully is therefore

\[
p_m^{\text{CBMC}} \approx a \left[ 1 - (1 - a)^k \right]^{m-1} = ab^{m-1}.
\]
The conformational-bias Monte Carlo scheme fails if the molecule is trapped in a dead alley (left); irrespective of the number of trial orientations the CBMC scheme will never generate an acceptable conformation. In the recoil growth scheme (right) the algorithm "recoils" back to a previous monomer and attempts to regrow from there.

This crude estimate suggests that by increasing $k$, the number of trial orientations, we can make $b$ arbitrarily close to 1 and hence obtain a reasonable insertion probability for any chain length and at any density. In practice, simply increasing $k$ will not solve the problem. First of all, there is a practical limitation: increasing $k$ increases the computational cost. More importantly, the assumption that the probability of a successful insertion of a monomer is equal and independent for each trial position is not correct. For instance, if we have grown into a "dead alley" where there is simply no space for an additional monomer (see Figure 13.13), then no matter how often we try, the insertion will not be accepted. At high densities such dead alleys are the main reason the CBMC method becomes inefficient. This suggests that we need a computational scheme that allows us to escape from these dead alleys.

The recoil growth (RG) scheme is a dynamic Monte Carlo algorithm that was developed with the dead-alley problem in mind [410, 411]. The algorithm is related to earlier static MC schemes due to Meirovitch [412] and Alexandrowicz and Wilding [413]. The basic strategy of the method is that it allows us to escape from a trap by "recoiling back" a few monomers and retrying the growth process using another trial orientation. In contrast, the CBMC scheme looks only one step ahead. Once a trial orientation has been selected, we cannot "deselect" it, even if it turns out to lead into a dead al-
Chapter 13. Biased Monte Carlo Schemes

The recoil growth scheme looks several monomers ahead to see whether traps are to be expected before a monomer is irrevocably added to the trial conformation (see Figure 13.13). In this way we can alleviate (but not remove) the dead-alley problem. In principle, one could also do something similar with CBMC by adding a sequence of \( l \) monomers per step. However, as there are \( k \) possible directions for every monomer, this would involve computing \( k^l \) energies per group. Even though many of these trial monomers do not lead to acceptable conformations, we would still have to compute all interaction energies.

### 13.7.1 Algorithm

In order to explain the practical implementation of the RG algorithm, let us first consider a totally impractical, but conceptually simple scheme that will turn out to have the same net effect. Consider a chain of \( l \) monomers. We place the first monomer at a random position. Next, we generate \( k \) trial positions for the second monomer. From each of these trial positions, we generate \( k \) trial positions for the third monomer. At this stage, we have generated \( k^2 \) "trimer" chains. We continue in the same manner until we have grown \( k^{l-1} \) chains of length \( l \). Obviously, most of the conformations thus generated have a vanishing Boltzmann factor and are, therefore, irrelevant. However, some may have a reasonable Boltzmann weight and it is these conformations that we should like to find. To simplify this search, we introduce a concept that plays an important role in the RG algorithm: we shall distinguish between trial directions that are "open" and those that are "closed." To decide whether a given trial direction, say \( b \), for monomer \( j \) is open, we compute its energy \( u_j(b) \). The probability \(^8 \) that trial position \( b \) is open is given by

\[
p_j^{\text{open}}(b) = \min(1, \exp[-\beta u_j(b)]),
\]

For hard-core interactions, the decision whether a trial direction is open or closed is unambiguous, as \( p_j^{\text{open}}(b) \) is either zero or one. For continuous interactions we compare \( p_j^{\text{open}}(b) \) with a random number between 0 and 1. If the random number is less than \( p_j^{\text{open}}(b) \), the direction is open; otherwise it is closed. We now have a tree with \( k^{l-1} \) branches but many of these branches are "dead," in the sense that they emerge from a "closed" monomer. Clearly, there is little point in exploring the remainder of a branch if it does not correspond to an "open" direction. This is where the RG algorithm comes in. Rather than generating a host of useless conformations, it generates them "on the fly." In addition, the algorithm uses a cheap test to check if a given

---

8This probability can be chosen in many alternative ways and may be used to optimize a simulation. However, the particular choice discussed here appears to work well for Lennard-Jones and hard-core potentials.
branch will "die" within a specified number of steps (this number is denoted by $l_{\text{max}}$). The algorithm then randomly chooses among the available open branches. As we have only looked a distance $l_{\text{max}}$ ahead, it may still happen that we have picked a branch that is doomed. But the probability of ending up in such a dead alley is much lower than that in the CBMC scheme.

In practice, the recoil growth algorithm consists of two steps. The first step is to grow a new chain conformation using only "open" directions. The next step is to compute the weights of the new and the old conformations.

The following steps are involved in the generation of a new conformation:

1. The first monomer of a chain is placed at a random position. The energy of this monomer is calculated ($u_1$). The probability that this position is "open" is given by equation (13.7.1). If the position is closed we cannot continue growing the chain and we reject the trial conformation. If the first position is open, we continue with the next step.

2. A trial position $b_{i+1}$ for monomer $i + 1$ is generated starting from monomer $i$. We compute the energy of this trial monomer $u_{i+1}(b)$ and, using equation (13.7.1), we decide whether this position is open or closed. If this direction is closed, we try another trial position, up to a maximum $^9$ of $k$ trial orientations. As soon as we find an open position we continue with step 3.

If not a single open trial position is found, we make a recoil step. The chain retracts one step to monomer $i - 1$ (if this monomer exists), and the unused directions (if any) from step 2, for $i - 1$, are explored. If all directions at level $i - 1$ are exhausted, we attempt to recoil to $i - 2$. The chain is allowed to recoil a total of $l_{\text{max}}$ steps, i.e., down to length $i - l_{\text{max}} + 1$.

If, at the maximum recoil length, all trial directions are closed, the trial conformation is discarded.

3. We have now found an "open" trial position for monomer $i + 1$. At this point monomer $i - l_{\text{max}}$ is permanently added in the new conformation; i.e., a recoil step will not reach this monomer anymore.

4. Steps 2 and 3 are repeated until the entire chain has been grown.

In the naive version of the algorithm sketched above, we can consider the above steps as a procedure for searching for an open branch on the existing tree. However, the RG procedure does this by generating the absolute minimum of trial directions compatible with the chosen recoil distance $l_{\text{max}}$.

---

^9 The maximum number of trial orientation should be chosen in advance—and may depend on the index $i$—but is otherwise arbitrary.
Once we have successfully generated a trial conformation, we have to decide on its acceptance. To this end, we have to compute the weights, \( W(n) \) and \( W(o) \), of the new and the old conformations, respectively. This part of the algorithm is more expensive. However, we only carry it out once we know for sure that we have successfully generated a trial conformation. In contrast, in CBMC it may happen that we spend much of our time computing the weight factor for a conformation that terminates in a dead alley.

In the RG scheme, the following algorithm is used to compute the weight of the new conformation:

1. Consider that we are at monomer position \( i \) (initially, of course, \( i = 1 \)). In the previous stage of the algorithm, we have already found that at least one trial direction is available (namely, the one that is included in our new conformation). In addition, we may have found that a certain number of directions (say \( k_c \)) are closed—these are the ones that we tried but that died within \( l_{\text{max}} \) steps. We still have to test the remaining \( 1 < k_{\text{rest}} = k - 1 - k_c \) directions. We randomly generate \( k_{\text{rest}} \) trial positions for monomer \( i + 1 \) and use the recoil growth algorithm to test whether at least one “feeler” of length \( l_{\text{max}} \) can be grown in this direction grown (unless \( i + l_{\text{max}} > l \); in that case we only continue until we have reached the end of the chain). Note that, again, we do not explore all possible branches. We only check if there is at least one open branch of length \( l_{\text{max}} \) in each of the \( k_{\text{rest}} \) directions. If this is the case, we call that direction “available.” We denote the total number of available directions (including the one that corresponds to the direction that we had found in the first stage of the algorithm) by \( m_i \). In the next section we shall derive that monomer \( i \) contributes a factor \( w_i(n) \) to the weight of the chain, where \( w_i(n) \) is given by

\[
\frac{w_i(n)}{P_i^{\text{open}}(n)}
\]

and \( P_i^{\text{open}}(n) \) is given by equation (13.7.1).

2. Repeat the previous step for all \( i \) from 1 to \( l - 1 \). The expression for the partial weight of the final monomer seems ambiguous, as \( m_l(n) \) is not defined. An easy (and correct) solution is to choose \( m_l(n) = 1 \).

3. Next compute the weight for the entire chain:

\[
W(n) = \prod_{i=1}^{l} w_i(n) = \prod_{i=1}^{l} \frac{m_i(n)}{P_i^{\text{open}}(n)}.
\]

For the calculation of the weight of the old conformation, we use almost the same procedure. The difference is that, for the old conformation, we have
to generate $k - 1$ additional directions for every monomer $i$. The weight is again related to the total number of directions that start from monomer $i$ and that are "available," i.e., that contain at least one open feeler of length $l_{\text{max}}$: \[ W(o) = \prod_{i=1}^{\ell} w_i(o) = \prod_{i=1}^{\ell} \frac{m_i(o)}{p_i^\text{open}(o)}. \]

Finally, the new conformation is accepted with a probability:

\[ \text{acc}(o \rightarrow n) = \min(1, \exp[-\beta U(n)]W(n)/\exp[-\beta U(o)]W(o)), \] (13.7.3)

where $U(n)$ and $U(o)$ are the energies of the new and old conformations, respectively. In the next section, we demonstrate that this scheme generates a Boltzmann distribution of conformations.

### 13.7.2 Justification of the Method

The best way to arrive at the acceptance rule for the recoil growth scheme is to pretend that we actually carry out the naive brute-force calculation where we first generate the tree of all $k^{1-1}$ trial conformations. We denote this tree by $T_n$ and the a priori probability for generating this tree by $P_T(T_n)$. Next we test which links are "open" or "closed." The decision whether a monomer direction is "open" or "closed" is made on the basis of the probabilities equation (13.7.1) and we denote the probability that we have a particular set $O_n$ of "open" monomers (and all others "closed") by $P_O(O_n|T_n)$. Let us note the number of "open" monomers in this set by $N(O_n)$ and the number of "closed" monomers by $N(C_n)$. It is easy to see that the probability of generating this particular set is given by

\[ P_O(O_n|T_n) = \prod_{j=1}^{N(O_n)} p_j^\text{open}(b) \prod_{k=1}^{N(C_n)} (1 - p_k^\text{open}(b)). \]

Finally we try to select one completely open conformation by randomly selecting, at every step, one of the "available" trial directions, i.e., a direction that is connected to (at least) one feeler that does not "die" within $l_{\text{max}}$ steps. At every step, there are $m_i(n)$ such directions. Hence the probability of selecting a given direction is simply $1/m_i(n)$ and the total probability that a specific conformation will be selected on the previously generated tree of all possible conformations is

\[ P_S(n|O_n) = \prod_{i=1}^{l-1} \frac{1}{m_i(n)}. \]
if all \( m_i \) are nonzero, and

\[ P_S(n | O_n) = 0 \]

otherwise. The fact that the algorithm leaves out many redundant steps (viz. generating the "doomed" branches or checking if there is more than one open feeler in a given direction) is irrelevant for the acceptance rule. The overall probability that we generate a trial conformation \( n \) on the set \( O_n \), \( P_S(n | O_n) \), in a tree \( T_n \) is

\[ P_T(T_n) \times P_O(O_n | T_n) \times P_S(n | O_n). \quad (13.7.4) \]

In order to compute acceptance probability of a trial move, we should consider the reverse move where the old configuration is generated. By analogy to the forward case, this probability is given by

\[ P_T(T_o) \times P_O(O_o | T_o) \times P_S(o | O_o). \quad (13.7.5) \]

We wish our MC scheme to obey detailed balance. However, just as in the CBMC case, it is easier to impose the stronger condition of super-detailed balance. This implies that, in the forward move, we also should consider the probability of generating a complete tree of possible conformations around the "old" conformation and the probability that a subset of all monomers on this tree is "open." We denote the probability of generating this tree by \( P_T'(T_o') \), where the prime indicates that this is the probability of generating all branches of the old tree, except the already existing old conformation. Clearly

\[ P_T(T_o) = P_T'(T_o') \times P_{gen}(o), \quad (13.7.6) \]

where \( P_{gen}(o) \) denotes the probability of generating the old conformation. As in the CBMC scheme, we can include strong intramolecular interactions in the generation of these trial monomers (see section 13.3). \( P_{gen}(o) \) will then be of the form (see section 13.3)

\[ P_{gen}(o) = \left( \prod_{i=1}^{l} p_i^{\text{bond}}(b_o) \right). \quad (13.7.7) \]

Similarly, we have to consider the probability \( P_O'(O_o' | T_o') \) that a set \( O_o' \) on this tree is "open." Again, the prime indicates that we should not include the old conformation itself. Again, it is easy to see that

\[ P_O(O_o | T_o) = \left( \prod_{i=1}^{l} p_i^{\text{open}}(b_o) \right) P_O'(O_o' | T_o'). \quad (13.7.8) \]
The *a priori* probability of generating a trial move from \( o \) to \( n \) is then given by

\[
\alpha(o \to n|T_n, O_n, T_o, O_o) = P_T(T_n) \times P_O(O_n|T_n) \times P_S(n|O_n) \times P_T'(T_o') \times P_O'(O_o'|T_o').
\]  
(13.7.9)

For the reverse move \( n \to o \), we can derive a similar expression:

\[
\alpha(n \to o|T_n, O_n, T_o, O_o) = P_T(T_o) \times P_O(O_o|T_o) \times P_G(o|O_o) \times P_T'(T_n') \times P_O'(O_n'|T_n').
\]  
(13.7.10)

In these equations we have used the notation \((o \to n|T_n, O_n, T_o, O_o)\) to indicate that we consider a transition from \( o \) to \( n \) (or vice versa) for a given set of "embedding" conformations. Clearly, there are many different trees and sets of open orientations that include the same conformations \( n \) and \( o \).

Our super-detailed balance condition now becomes

\[
N(o) \times \alpha(o \to n|T_n, O_n, T_o, O_o) \text{acc}(o \to n|T_n, O_n, T_o, O_o)
= N(n) \times \alpha(n \to o|T_n, O_n, T_o, O_o) \text{acc}(n \to o|T_n, O_n, T_o, O_o).
\]  
(13.7.11)

All terms in this equation are known, except the acceptance probabilities. We now derive an expression for the ratio \( \text{acc}(o \to n|T_n, O_n, T_o, O_o)/\text{acc}(n \to o|T_n, O_n, T_o, O_o) \). To this end, we insert equations (13.7.6) and (13.7.8) (and the corresponding expressions for \( P_T'(T_n') \) and \( P_O'(O_n'|T_n') \)) into our super-detailed balance condition equation (13.7.11). This leads to a huge simplification as there is a complete cancellation of all probabilities for generating "open" or "closed" monomers that do not belong to the new (or the old) conformation. What remains is

\[
N(o) \times P_{gen}(n) \left( \prod_{i=1}^{1} \frac{p_{i, open}(b_n)}{m_i(n)} \right) \text{acc}(o \to n|T_n, O_n, T_o, O_o)
= N(n) \times P_{gen}(o) \left( \prod_{i=1}^{1} \frac{p_{i, open}(b_o)}{m_i(o)} \right) \text{acc}(n \to o|T_n, O_n, T_o, O_o).
\]  
(13.7.12)

In order to simplify the notation, we shall assume that the trial directions are uniformly distributed, i.e., see equation (13.7.7), \( p_{\text{bond}} \) = constant. From equation (13.7.7) it then follows that \( P_{gen}(n) \) and \( P_{gen}(o) \) are identical constants.
Our expression for the ratio of the acceptance probabilities then becomes

\[
\frac{\text{acc}(o \rightarrow n)}{\text{acc}(n \rightarrow o)} = \frac{\mathcal{N}(n) \prod_{i=1}^{\ell} p_i^{\text{open}}(o)/m_i(o)}{\mathcal{N}(o) \prod_{i=1}^{\ell} p_i^{\text{open}}(n)/m_i(n)},
\]

(13.7.13)

where we have dropped the indices \(T_n, O_n, \ldots\). Using the definitions of \(W(n)\) and \(W(o)\) (equation (13.7.2) and below),

\[
\frac{\text{acc}(o \rightarrow n)}{\text{acc}(n \rightarrow o)} = \frac{\mathcal{N}(n)W(n)}{\mathcal{N}(o)W(o)}.
\]

(13.7.14)

This is precisely the acceptance rule given by equation (13.7.3). This concludes our "derivation" of the recoil growth scheme. The obvious question is: how well does it perform? A comparison between CBMC and the RG algorithm was made by Consta et al. [411], who studied the behavior of Lennard-Jones chains in solution. The simulations showed that for relatively short chains (\(\ell = 10\)) at a density of \(\rho = 0.2\), the recoil growth scheme was a factor of 1.5 faster than CBMC. For higher densities \(\rho = 0.4\) and longer chains \(N = 40\) the gain could be as large as a factor 25. This illustrates the fact that the recoil scheme is still efficient, under conditions where CBMC is likely to fail. For still higher densities or still longer chains, the relative advantage of RG would be even larger. However, the bad news is that, under those conditions, both schemes become very inefficient.

While the recoil growth scheme is a powerful alternative to CBMC, the RG strategy is not very useful for computing chemical potentials (see [411]). More efficient schemes for computing the chemical potential are the recursive sampling scheme and the Pruning-Enriched Rosenbluth Method (PERM) (see Chapter 11).

Case Study 20 (Recoil Growth Simulation of Lennard-Jones Chains)

To illustrate the recoil growth (RG) method, we make a comparison between this method and conformational-bias Monte Carlo (CBMC). Consider 20 Lennard-Jones chains of length 15. The monomer density is \(\rho = 0.3\) at temperature \(T = 6.0\). Two bonded monomers have a constant bond length of 1.0, while three successive particles have a constant bond angle of 2.0 radians.

In Figure 13.14 the distribution of the end-to-end vector, \(R_e\), of the chain is plotted. In this figure we compare the results from a CBMC and a RG. Since both methods generate a Boltzmann distribution of conformations, the results are identical (as they should be).

For this specific example, we have compared the efficiency, \(\eta\), of the two methods. The efficiency is defined as the number of accepted trial moves per amount of CPU time. For CBMC we see that the efficiency increases as we increase \(k\), the number of trial orientations, from 1 to 4. From 4 to 8 the
efficiency is more or less constant, and above 8 a decrease in the efficiency is observed.

In the RG scheme we have two parameters to optimize: the number of trial orientations \( k \) and the recoil length \( l_{\text{max}} \). If we use only one trial orientation, recoiling is impossible, since there are no other trial orientations. If we use a recoil length of 1, the optimum number of trial orientations is 4 and for larger recoil lengths the optimum is reached with less trial orientations. Interestingly, the global optimum is 2 trial orientations and a recoil length of 3–5. In this regime, the increase in CPU time associated with a larger recoil length is compensated by a higher acceptance. In the present study, optimal RG was a factor 8 more efficient than optimal CBMC.

13.8 Questions and Exercises

**Question 20 (Biased CBMC)** In a configurational-bias Monte Carlo simulation, trial positions are selected with a probability that is proportional to the Boltzmann factor of each trial segment. However, in principle one can use another probability function [382] to select a trial segment. Suppose that the probability of selecting a trial segment \( i \) is proportional to

\[
p_i \propto \exp \left[ -\beta^* u_i \right]
\]

in which \( \beta^* \neq \beta \).
1. Derive the correct acceptance/rejection rule for this situation.
2. Derive an expression for the excess chemical potential when this modified CBMC method is used to generate configurations of test particles.
3. What happens if $\beta^* \to \infty$ and if $\beta^* \to 0$?

**Exercise 15 (CBMC of a Single Chain)**

In this exercise, we will look at the properties of a single chain molecule. We will compare various sampling schemes. Suppose that we have a chain molecule of length $n$ in which there are the following interactions between beads:

- Two successive beads have a fixed bond length 1. We will use $l = 1$.
- Three successive beads have a bond-bending interaction
  \[ U = \frac{1}{2} k_t (\theta - \theta_0)^2, \]
  in which $\theta$ is the bond angle, $\theta_0$ is the equilibrium bond angle, and $k_t$ is a constant. We will use $\theta_0 = 2.0 \text{ rad} \approx 114.6^\circ$ and $k_t = 2.0$.
- Every pair of beads that is separated by more than two bonds has a soft repulsive interaction
  \[ U(r) = \begin{cases} \frac{A(r-r_{\text{cut}})^2}{r_{\text{cut}}^2} & r \leq r_{\text{cut}} \\ 0 & r > r_{\text{cut}} \end{cases}, \]
  in which $r_{\text{cut}}$ is the cutoff radius (we will use $r_{\text{cut}} = 1.0$ and $A > 0$).

An interesting property of a chain molecule is the distribution of the end-to-end distance, which is the distance between the first and the last segments of the chain. There are several possible schemes for studying this property:

**Dynamic Schemes** In a dynamic scheme, a Markov chain of states is generated. The average of a property $B$ is the average of $B$ over the elements of the Markov chain

\[ \langle B \rangle \approx \frac{\sum_{i=1}^{N} B_i}{N}. \]

In the limit $N \to \infty$ this expression becomes exact. Every new configuration is accepted or rejected using an acceptance criterion:

- When unbiased chains are generated:
  \[ \text{acc} (o \to n) = \min \left(1, \exp \{-\beta [U(n) - U(o)]\} \right), \]
  in which $U$ is the total energy (soft repulsion and bond bending) of a chain.
• When configurational-bias Monte Carlo is used:

\[
\text{acc}(o \rightarrow n) = \min \left( 1, \frac{W(n)}{W(o)} \right),
\]

in which

\[
W = \frac{\prod_{i=2}^{n} \sum_{j=1}^{k} \exp[-\beta U(i,j)]}{k^{n-1}}.
\]

In this equation, \(k\) is the number of trial positions and \(U(i,j)\) is the energy of the \(j\)th trial position of the \(i\)th chain segment. The term \(U(i,j)\) does not contain the bond-bending potential, because that potential has already been used to generate the trial positions.

Static Schemes In a static scheme, all configurations are generated independently. To obtain a canonical average, every configuration is weighted with a factor \(R\)

\[
\langle B \rangle = \frac{\sum_{i=1}^{\text{N}} B_i \times R_i}{\sum_{i=1}^{\text{N}} R_i}.
\]

For \(R_i\) we can write:

• When random chains are generated:

\[
R_i = \exp[-\beta U_i].
\]

Here, \(U_i\) is the total energy of the chain.

• When CBMC is used:

\[
R_i = W. 
\quad (13.8.1)
\]

1. On the book’s website you can find a program for calculating chain properties using these four methods. However, some additional programming has to be done in the file \(grow.f\), which is a subroutine for growing a new chain using either CBMC or random insertion.

2. Compare the end-to-end distance distributions of the four methods. Which method has the best performance? Investigate how the efficiency of CBMC depends on the number of trial directions \(k\).

3. Investigate the influence of chain length on the end-to-end distance distribution. For which chain lengths do the four methods start to fail?

4. For high temperatures (and for low \(k_t\) and \(A\), the end-to-end distance distribution looks like the distribution of a nonself-avoiding random walk. This means that the chain segments are randomly oriented
and the segments are allowed to overlap. For the mean square end-to-end distance, we can write

\[ \frac{\langle r^2 \rangle}{l^2} = \left\langle \left( \sum_{i=1}^{i=n} x_i^2 \right) + \left( \sum_{i=1}^{i=n} y_i^2 \right) + \left( \sum_{i=1}^{i=n} z_i^2 \right) \right\rangle, \]

in which \((x_i, y_i, z_i)\) are the projections of each segment on the \((x, y, z)\) axes

\[
x_i = \sin(\theta_i) \cos(\phi_i)
\]
\[
y_i = \sin(\theta_i) \sin(\phi_i)
\]
\[
z_i = \cos(\theta_i).
\]

This set of equations can be reduced to

\[ \frac{\langle r^2 \rangle}{l^2} = n. \quad (13.8.2) \]

- Derive equation (13.8.2). Hint: the following equations will be very useful:

\[
\cos^2(\theta_i) + \sin^2(\theta_i) = 1
\]
\[
\cos(\theta_i - \theta_j) = \cos(\theta_i) \cos(\theta_j) + \sin(\theta_i) \sin(\theta_j)
\]
\[
\langle \cos(\theta_i - \theta_j) \rangle = 0.
\]

The last equation holds because \(\theta_i - \theta_j\) is uniformly distributed.

- Modify the program in such a way that \(\langle r^2 \rangle\) is calculated for a nonself-avoiding random walk. Compare your results with the analytical solution.

- Does \(\langle r^2 \rangle \propto n\) hold for a chain with a potential energy function described in this exercise? Investigate the influence of \(A\) on the end-to-end distance distribution.

Exercise 16 (CBMC of a Simple System)

Consider a system with three coordinates \((x_1, x_2, x_3)\) and phase space density

\[ \rho(x_1, x_2, x_3) = \exp \left[-(x_1^2 + x_2^2 + x_3^2)\right] = \exp \left[-r^2\right]. \]

We wish to calculate the average \(\langle r^2 \rangle\),

\[ \langle r^2 \rangle = \frac{\iiint dx_1 dx_2 dx_3 r^2 \rho}{\iiint dx_1 dx_2 dx_3 \rho}, \]

using the CBMC algorithm of Falcioni and Deem [414]:
13.8 Questions and Exercises

- Generate $k$ sets of new coordinates $B_1, \cdots, B_k$ by adding random vectors to the old configuration $(A_1)$.
- Select one set (i) with a probability proportional to its Boltzmann factor,
  \[ p_i = \exp[-r_{B_i}]. \]
  The corresponding Rosenbluth factor weight is
  \[ W(n) = \sum_{j=1}^{j=k} \exp[-r_{B_j}^2]. \]
- Starting from the selected configuration $B_i$, $k-1$ configurations $(A_2, \cdots, A_k)$ are generated by adding a uniform vector to $B_i$. $A_1$ is the old configuration. This leads to the Rosenbluth factor of the old configuration
  \[ W(o) = \sum_{j=1}^{j=k} \exp[-r_{A_j}^2]. \]
- The new configuration $B_i$ is accepted with a probability
  \[ \text{acc} (o \rightarrow n) = \min \left( 1, \frac{W(n)}{W(o)} \right). \]

1. Make a sketch of the configurations $A_1, \cdots, A_k$ and $B_1, \cdots, B_k$. Show that for $k = 1$, this algorithm reduces to the standard Metropolis algorithm for particle displacements.
2. Prove that this algorithm obeys detailed balance.
3. Calculate $\langle r^2 \rangle$ analytically using
  \[ \int_0^\infty \exp[-a^2x^2] \, dx = \frac{\sqrt{\pi}}{2|a|}. \]
4. On the book’s website you can find a computer program for this CBMC sampling scheme. This program, however, has to be completed by you (see the file `cbmc.j`). Make sure that your estimate of $\langle r^2 \rangle$ is independent of the number of trial directions ($k$).
5. What happens with the fraction of accepted trial moves when the number of trial directions ($k$) is increased? Make a plot of the fraction of accepted trial moves as a function of $k$ for various maximum displacements. Explain your results.
6. Why is this CBMC method useful when the system is initially far from equilibrium?