John von Neumann Institute for Computing

Introduction to Monte Carlo Methods

Daan Frenkel

published in

Computational Soft Matter: From Synthetic Polymers to Proteins, Lecture Notes,
Norbert Attig, Kurt Binder, Helmut Grubmüller, Kurt Kremer (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **23***, ISBN 3-00-012641-4, pp. 29-60, 2004.*

http://www.fz-juelich.de/nic-series/volume23

Introduction to Monte Carlo Methods

Daan Frenkel

FOM Institute for Atomic and Molecular Physics,
Kruislaan 407, 1098 SJ Amsterdam, The Netherlands
E-mail: frenkel@amolf.nl

These give an introduction to Monte Carlo simulations. After a general introduction of the approach and practical implementation, special attention is paid to the used of biased sampling methods in the context of polymer simulations

1 Introduction

The Monte Carlo techniques that are described in this chapter can be used to compute the equilibrium properties of classical many-body systems. In this context, the word 'classical' means that the core motion of the constituent particles obeys the laws of classical mechanics. This is an excellent approximation for a wide range of materials. Only when we consider the translational or rotational motion of light atoms or molecules (He, H_2, D_2,) or vibrational motion with a frequency such that $h\nu > k_B T$, should we worry about quantum effects.

These lecture notes provide a somewhat selective introduction to the Monte Carlo (MC) method. The selection reflects my own personal bias. It is largely (but not completely) based on the more complete description given in ref.[1]

Before embarking on a description of the MC method, I should first briefly explain the role of computer simulations in general. This topic is best discussed by considering the situation that prevailed before the advent of electronic computers. At that time, there was only one way to predict the outcome of an experiment, namely by making use of a theory that provided an approximate description of the system under consideration. The reason why an approximate theory was almost always used is that there are very few model systems for which the equilibrium properties can be computed exactly (examples are the ideal gas, the harmonic crystal and a number of lattice models, such as the two-dimensional Ising model for ferro-magnets), and even fewer model systems for which the transport properties can be computed exactly. As a result, most properties of real materials were predicted on the basis of approximate theories (examples are the van der Waals equation for dense gases, the Debye-Hückel theory for electrolytes, or the Boltzmann equation to describe the transport properties of dilute gases). Given sufficient information about the intermolecular interactions, these theories will provide us with an estimate of the properties of interest. Unfortunately, our knowledge of the intermolecular interactions of all but the simplest molecules is quite limited. This leads to a problem if we wish to test the validity of a particular theory by comparing directly to experiment. If we find that theory and experiment disagree, it may mean that our theory is wrong, or that we have an incorrect estimate of the intermolecular interactions, or both... Clearly, it would be very nice if we could obtain essentially exact results for a given model system, without having to rely on approximate theories. Computer simulations allow us to do precisely that. On the one hand, we can

now compare the calculated properties of a model system with those of an experimental system: if the two disagree, our model is inadequate, *i.e.* we have to improve on our estimate of the intermolecular interactions. On the other hand, we can compare the result of a simulation of a given model system with the predictions of an approximate analytical theory applied to the same model. If we now find that theory and simulation disagree, we know that the *theory* is flawed. So, in this case, the computer simulation plays the role of the 'experiment' that is designed to test the theory. This method to 'screen' theories before we apply them to the real world, is called a 'computer experiment'. Computer experiments have become standard practice, to the extent that they now provide the first (and often the last) test of a new theoretical result.

In fact, the early history of computer simulation (see *e.g.* ref.[2]) illustrates this role of computer simulation. In some areas of physics there appeared to be little need for simulation because very good analytical theories were available (*e.g.* to predict the properties of dilute gases or of nearly harmonic crystalline solids). However, in other areas, few if any 'exact' theoretical results were known, and progress was much hindered by the fact that there were no unambiguous tests to assess the quality of approximate theories. A case in point was the theory of dense liquids. Before the advent of computer simulations, the only way to model liquids was by mechanical simulation[3–5] of large assemblies of macroscopic spheres (*e.g.* ball bearings). But such mechanical models are rather crude, as they ignore the effect of thermal motion. Moreover, the analysis of the structures generated by mechanical simulation was very laborious and, in the end, had to be performed by computer anyway.

This may explain why, when in 1953 electronic computers were, for the first time, made available for unclassified research, numerical simulation of dense liquids was one of the first problems that was tackled. In fact, the first simulation of a 'liquid' was carried out by Metropolis et al.[6] at Los Alamos, using (or, more properly, introducing) the Monte Carlo method. Almost simultaneously, Fermi, Pasta and Ulam[7] performed a very famous numerical study of the dynamics of an anharmonic, one-dimensional crystal. However, the first proper Molecular Dynamics simulations were reported in 1956 by Alder and Wainwright[8] at Livermore, who studied the dynamics of an assembly of hard spheres. The first MD simulation of a model for a 'real' material was reported in 1959 (and published in 1960) by the group of Vineyard at Brookhaven[9], who simulated radiation damage in crystalline Cu (for a historical account, see[10]). The first MD simulation of a real liquid (argon) was reported in 1964 by Rahman at Argonne[11]. After that, computers were increasingly becoming available to scientists outside the US government labs, and the practice of simulation started spreading to other continents[12–14]. Much of the methodology of MD simulations has been developed since then, although it is fair to say that the basic algorithms for MC and MD have hardly changed since the fifties.

1.1 Statistical mechanics

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

Our aim is to indicate where the Monte Carlo method comes in. We start from the

classical expression for the partition function Q:

$$Q = c \int \mathrm{d}\mathbf{p}^N \mathrm{d}\mathbf{r}^N \exp[-\mathcal{H}(\mathbf{r}^N\ \mathbf{p}^N)/k_B T], \tag{1}$$

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

$$\langle A \rangle = \frac{\int \mathrm{d}\mathbf{p}^N \mathrm{d}\mathbf{r}^N\ A(\mathbf{p}^N, \mathbf{r}^N) \exp[-\beta \mathcal{H}(\mathbf{p}^N, \mathbf{r}^N)]}{\int \mathrm{d}\mathbf{p}^N \mathrm{d}\mathbf{r}^N\ \exp[-\beta \mathcal{H}(\mathbf{p}^N, \mathbf{r}^N)]}, \tag{2}$$

where $\beta = 1/k_B T$. In this equation, the observable A has been expressed as a function of coordinates and momenta. As $\mathcal{K}$ is a quadratic function of the momenta the integration over momenta can be carried out analytically. Hence, averages of functions that depend on momenta only are usually easy to evaluate. The difficult problem is the computation of averages of functions $A(\mathbf{r}^N)$. Only in a few exceptional cases can the multidimensional integral over particle coordinates be computed analytically, in all other cases numerical techniques must be used.

Having thus defined the nature of the numerical problem that we must solve, let us next look at possible solutions. It might appear that the most straightforward approach would be to evaluate $\langle A \rangle$ in equation 2 by numerical quadrature, for instance using Simpson's rule. It is easy to see, however, that such a method is completely useless even if the number of independent coordinates DN (D is the dimensionality of the system) is still very small $\mathcal{O}(100)$. Suppose that we plan to carry out the quadrature by evaluating the integrand on a mesh of points in the DN-dimensional configuration space. Let us assume that we take m equidistant points along each coordinate axis. The total number of points at which the integrand must be evaluated is then equal to m^{DN}. For all but the smallest systems this number becomes astronomically large, even for small values of m. For instance, if we take 100 particles in three dimensions, and $m = 5$, then we would have to evaluate the integrand at 10^{210} points! Computations of such magnitude cannot be performed in the known universe. And this is fortunate, because the answer that would be obtained would have been subject to a large statistical error. After all, numerical quadratures work best on functions that are smooth over distances corresponding to the mesh size. But for most intermolecular potentials, the Boltzmann factor in equation 2 is a rapidly varying function of the particle coordinates. Hence an accurate quadrature requires a small mesh spacing (*i.e.* a large value of m). Moreover, when evaluating the integrand for a dense liquid (say), we would find that for the overwhelming majority of points this Boltzmann factor is vanishingly small. For instance, for a fluid of 100 hard spheres at the freezing point, the Boltzmann factor would be nonzero for 1 out of every 10^{260} configurations!

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

tion itself but in averages of the type

$$\langle A \rangle = \frac{\int \mathrm{d}\mathbf{r}^N \, \exp[-\beta \mathcal{U}(\mathbf{r}^N)] A(\mathbf{r}^N)}{\int \mathrm{d}\mathbf{r}^N \, \exp[-\beta \mathcal{U}(\mathbf{r}^N)]}. \tag{3}$$

Hence, we wish to know the *ratio* of two integrals. What Metropolis *et al.*[6] showed is that it is possible to devise an efficient Monte Carlo scheme to sample such a ratio.[a] To understand the Metropolis method, let us first look more closely at the structure of equation 3. In what follows we denote the configurational part of the partition function by Z:

$$Z \equiv \int \mathrm{d}\mathbf{r}^N \, \exp[-\beta \mathcal{U}(\mathbf{r}^N)]. \tag{4}$$

Note that the ratio $\exp(-\beta \mathcal{U})/Z$ in equation 3 is the probability density to find the system in a configuration around $\mathbf{r}^N$. Let us denote this probability density by

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

Clearly, $\mathcal{N}(\mathbf{r}^N)$ is > 0.

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

$$\langle A \rangle \approx \frac{1}{L} \sum_{i=1}^{L} n_i \, A(\mathbf{r}_i^N). \tag{5}$$

By now the reader is almost certainly confused, so let us therefore try to clarify this method with the help of a simple example (see Figure 1). In this figure, we compare two ways to measure the depth of the river Nile, by conventional quadrature (left) and by Metropolis sampling; that is, the construction of an importance-weighted random walk (right). In the conventional quadrature scheme, the value of the integrand is measured at a predetermined set of points. As the choice of these points does not depend on the value of the integrand, many points may be located in regions where the integrand vanishes. In contrast, in the Metropolis scheme, a random walk is constructed through that region of space where the integrand is nonnegligible (*i.e.* through the Nile itself). In this random walk, a trial move is rejected if it takes you out of the water, and is accepted otherwise. After *every* trial move (accepted or not), the depth of the water is measured. The (unweighted) average of all these measurements yields an estimate of the average depth of the Nile. This, then, is the essence of the Metropolis method. In principle, the conventional quadrature scheme would also give results for the *total* area of the Nile. In the importance sampling scheme, however, information on the total area cannot be obtained directly, since this quantity is similar to Z.

Let us next consider how to generate points in configuration space with a relative probability proportional to the Boltzmann factor. The general approach is first to prepare the

[a] An interesting account of the early history of the Metropolis method may be found in H.L. Anderson, *J. Stat. Phys.* 43:731, 1986; and Wood[15] p. 3.

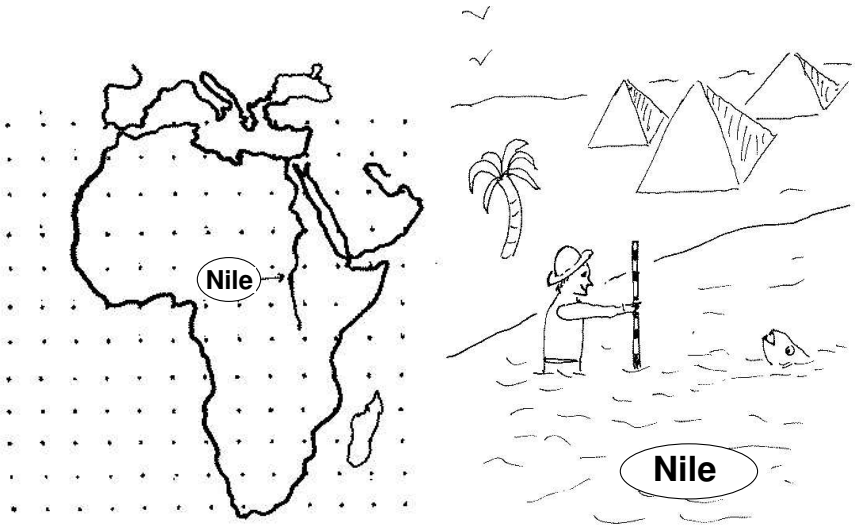

Figure 1. Measuring the depth of the Nile: a comparison of conventional quadrature (left), with the Metropolis scheme (right).

system in a configuration $\mathbf{r}^N$, which we denote by o (old), that has a nonvanishing Boltzmann factor $\exp[-\beta\mathcal{U}(o)]$. This configuration, for example, may correspond to a regular crystalline lattice with no hard-core overlaps. Next, we generate a new trial configuration $\mathbf{r}'^N$, which we denote by n (new), by adding a small random displacement Δ to o. The Boltzmann factor of this trial configuration is $\exp[-\beta\mathcal{U}(n)]$. We must now decide whether we will accept or reject the trial configuration. Many rules for making this decision satisfy the constraint that on average the probability of finding the system in a configuration n is proportional to $\mathcal{N}(n)$. Here we discuss only the Metropolis scheme, because it is simple and generally applicable.

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

$$\mathcal{N}(o)\pi(o \to n) = \mathcal{N}(n)\pi(n \to o). \tag{6}$$

Many possible forms of the transition matrix $\pi(o \to n)$ satisfy equation (6). Let us look how $\pi(o \to n)$ is constructed in practice. We recall that a Monte Carlo move consists of

two stages. First, we perform a trial move from state o to state n. We denote the transition matrix that determines the probability to perform a trial move from o to n by $\alpha(o \rightarrow n)$; where α is usually referred to as the underlying matrix of Markov chain[16]. The next stage is the decision to either accept or reject this trial move. Let us denote the probability of accepting a trial move from o to n by $\mathrm{acc}(o \rightarrow n)$. Clearly,

$$\pi(o \rightarrow n) = \alpha(o \rightarrow n) \times \mathrm{acc}(o \rightarrow n). \tag{7}$$

In the original Metropolis scheme, α is chosen to be a symmetric matrix ($\mathrm{acc}(o \rightarrow n) = \mathrm{acc}(n \rightarrow o)$). However, in later sections we shall see several examples where α is *not* symmetric. If α is symmetric, we can rewrite equation 6 in terms of the $\mathrm{acc}(o \rightarrow n)$:

$$\mathcal{N}(o) \times \mathrm{acc}(o \rightarrow n) = \mathcal{N}(n) \times \mathrm{acc}(n \rightarrow o). \tag{8}$$

From Eqn. 8 follows

$$\frac{\mathrm{acc}(o \rightarrow n)}{\mathrm{acc}(n \rightarrow o)} = \frac{\mathcal{N}(n)}{\mathcal{N}(o)} = \exp\{-\beta[\mathcal{U}(n) - \mathcal{U}(o)]\}. \tag{9}$$

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

$$\begin{aligned} \mathrm{acc}(o \rightarrow n) &= \mathcal{N}(n)/\mathcal{N}(o) \text{ if } \mathcal{N}(n) < \mathcal{N}(o) \\ &= 1 \qquad\qquad\quad \text{if } \mathcal{N}(n) \geq \mathcal{N}(o). \end{aligned} \tag{10}$$

Other choices for $\mathrm{acc}(o \rightarrow n)$ are possible (for a discussion, see for instance[17]), but the original choice of Metropolis *et al.* appears to result in a more efficient sampling of configuration space than most other strategies that have been proposed.

1.2 A Basic Monte Carlo Algorithm

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

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

1.3 The Algorithm

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

(and, to a lesser extent, of the Molecular Dynamics method). Let us now look at a simple Monte Carlo program.

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

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

$$\mathrm{acc}(o \to n) = \min\left(1, \exp\{-\beta[\mathcal{U}(\mathbf{r'}^N) - \mathcal{U}(\mathbf{r}^N)]\}\right). \tag{11}$$

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

Algorithm 1
[Basic Metropolis Algorithm]

```
  PROGRAM mc                          basic Metropolis algorithm

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

Comments:

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

1.4 Trial Moves

Now that we have specified the general structure of the Metropolis algorithm, we should consider its implementation. We shall not go into the problem of selecting intermolecular potentials for the model system under study. Rather, we shall simply assume that we have an atomic or molecular model system in a suitable starting configuration and that we have specified all intermolecular interactions. We must now set up the underlying Markov

Algorithm 2
[Attempt to Displace a Particle]

```
 SUBROUTINE mcmove              attempts to displace a particle

 o=int(ranf()*npart)+1          select a particle at random
 call ener(x(o),eno)            energy old configuration
 xn=x(o)+(ranf()-0.5)*delx      give particle random displacement
 call ener(xn,enn)              energy new configuration
 if (ranf().lt.exp(-beta        acceptance rule (11)
+   *(enn-eno)) x(o)=xn         accepted: replace x(o) by xn
 return
 end
```

Comments:

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

chain; that is, the matrix α. In more down-to-earth terms: we must decide how we are going to generate trial moves. We should distinguish between trial moves that involve only the molecular centers of mass, and those that change the orientation or possibly even the conformation of a molecule.

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

$$\begin{aligned} x_i' &\rightarrow x_i + \Delta\,(\mathrm{Ranf} - 0.5) \\ y_i' &\rightarrow y_i + \Delta\,(\mathrm{Ranf} - 0.5) \\ z_i' &\rightarrow z_i + \Delta\,(\mathrm{Ranf} - 0.5), \end{aligned} \tag{12}$$

where Ranf are random numbers uniformly distributed between 0 and 1. Clearly, the reverse trial move is equally probable (hence, α is symmetric).

We are now faced with two questions: how large should we choose Δ?, and should we attempt to move all particles simultaneously or one at a time? In the latter case we should pick the molecule that is to be moved at random to ensure that the underlying Markov chain remains symmetric. All other things being equal, we should choose the most efficient sampling procedure. But, to this end, we must first define what we mean by *efficient sampling*. Broadly speaking, sampling is efficient if it gives you good value for money. Good value in a simulation corresponds to high statistical accuracy, and "money" is simply *money*: the money that buys your computer time and even your own time. For the sake of the argument, we assume the average scientific programmer is poorly paid. In

that case we have to worry only about your computer budget.[b] Then we could use the following definition of an optimal sampling scheme: a Monte Carlo sampling scheme can be considered optimal if it yields the lowest statistical error in the quantity to be computed for a given expenditure of computing budget. Usually, computing budget is equivalent to CPU time.

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

Next, consider the choice of the parameter Δ which determines the size of the trial move. How large should Δ be? If it is very large, it is likely that the resulting configuration will have a high energy and the trial move will probably be rejected. If it is very small, the change in potential energy is probably small and most moves will be accepted. In the literature, one often finds the mysterious statement that an acceptance of approximately 50% should be optimal. This statement is not necessarily true. The optimum acceptance ratio is the one that leads to the most efficient sampling of configuration space. If we express efficiency as mean-square displacement per CPU time, it is easy to see that different Monte Carlo codes will have different optimal acceptance ratios. The reason is that it makes a crucial difference if the amount of computing required to test whether a trial move is accepted depends on the magnitude of the move (see Figure 2).

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

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

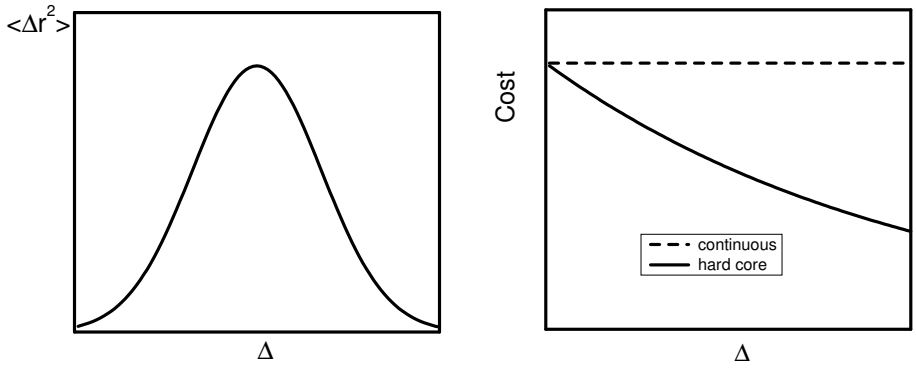

Figure 2. (left) Typical dependence of the mean-square displacement of a particle on the average size Δ of the trial move. (right) Typical dependence of the computational cost of a trial move on the step-size Δ. For continuous potentials, the cost is constant, while for hard-core potentials it decreases rapidly with the size of the trial move.

2 Canonical Ensemble

In a conventional Molecular Dynamics simulation, the total energy E and the total linear momentum P are constants of motion. Hence, Molecular Dynamics simulations measure (time) averages in an ensemble that is very similar to the microcanonical (see[18]); namely, the constant-NVE-P ensemble. In contrast, a conventional Monte Carlo simulation probes the canonical (*i.e.* constant-NVT) ensemble. The fact that these ensembles are different leads to observable differences in the statistical averages computed in Molecular Dynamics and Monte Carlo simulations. Most of these differences disappear in the thermodynamic limit and are already relatively small for systems of a few hundred particles. However, the choice of ensemble does make a difference when computing the mean-square value of fluctuations in thermodynamic quantities. Fortunately, techniques exist to relate fluctuations in different ensembles[19]. Moreover, nowadays it is common practice to carry out Molecular Dynamics simulations in ensembles other than the microcanonical. In particular, it is possible to do Molecular Dynamics at constant pressure, at constant stress, and at constant temperature. The choice of ensembles for Monte Carlo simulations is even wider: isobaric-isothermal, constant-stress-isothermal, grand-canonical (*i.e.* constant-μVT), and even microcanonical[20–25]. A more recent addition to this list is a Monte Carlo method that employs the Gibbs-ensemble technique[26], which was developed to study phase coexistence in moderately dense (multi component) fluids.

2.1 General Approach

In the following sections, we will use the following procedure to demonstrate the validity of our Monte Carlo algorithms:

1. Decide which distribution we want to sample. This distribution, denoted $\mathcal{N}$, will depend on the details of the ensemble.

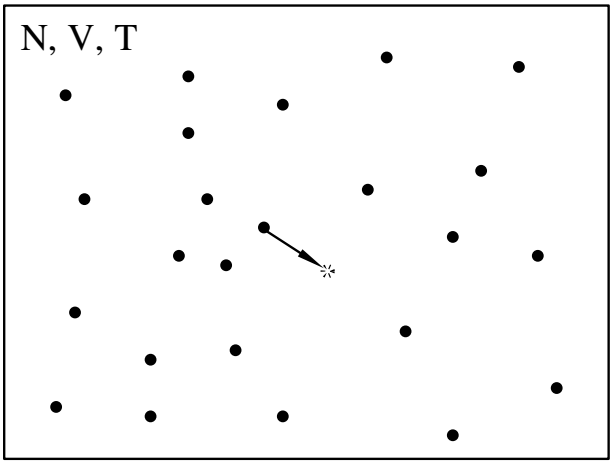

Figure 3. Canonical ensemble. The number of particles, volume, and temperature are constant. Shown is a Monte Carlo move in which a particle is displaced).

2. Impose the condition of detailed balance,

$$K(o \to n) = K(n \to o), \tag{13}$$

where $K(o \to n)$ is the flow of configuration o to n. This flow is given by the product of the probability of being in configuration o, the probability of generating configuration n, and the probability of accepting this move,

$$K(o \to n) = \mathcal{N}(o) \times \alpha(o \to n) \times \mathrm{acc}(o \to n). \tag{14}$$

3. Determine the probabilities of generating a particular configuration.

4. Derive the condition which needs to be fulfilled by the acceptance rules.

It is instructive to apply the preceding recipe to the ordinary Metropolis scheme. In the canonical ensemble, the number of particles, temperature, and volume are constant (see Figure 3).

The partition function is

$$Q(N,V,T) \equiv \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathbf{r}^N \, \exp[-\beta \mathcal{U}(\mathbf{r}^N)], \tag{15}$$

where $\Lambda = \sqrt{h^2/(2\pi m k_B T)}$ is the thermal de Broglie wavelength. From the partition function it follows that the probability of finding configuration $\mathbf{r}^N$ is given by distribution is

$$\mathcal{N}(\mathbf{r}^N) \propto \exp[-\beta \mathcal{U}(\mathbf{r}^N)]. \tag{16}$$

equations 15 and 16 are the basic equations for a simulation in the canonical ensemble.

2.2 Monte Carlo Simulations

In the canonical ensemble, we have to sample distribution (16). This can be done using the following scheme:

1. Select a particle at random and calculate the energy of this configuration $\mathcal{U}(o)$.

2. Give this particle a random displacement (see Figure 3),

$$\mathbf{r}(o) \rightarrow \mathbf{r}(o) + \Delta(\mathrm{Ranf} - 0.5),$$

   where $\Delta/2$ is the maximum displacement. The value of Δ should be chosen such that the sampling scheme is optimal (see section 1.4). The new configuration is denoted n and its energy $\mathcal{U}(n)$.

3. The move is accepted with a probability (see equation 10)

$$\mathrm{acc}(o \rightarrow n) = \min\left(1, \exp\{-\beta[\mathcal{U}(n) - \mathcal{U}(o)]\}\right). \tag{17}$$

   If rejected, the old configuration is kept.

An implementation of this basic Metropolis scheme is shown in Section 1.2 (Algorithms 1 and 2).

2.3 Justification of the Algorithm

The probability of generating a particular configuration is constant and independent of the conformation of the system

$$\alpha(o \rightarrow n) = \alpha(n \rightarrow o) = \alpha.$$

Substitution of this equation in the condition of detailed balance (13) and substitution of the desired distribution (16) gives as condition for the acceptance rules

$$\frac{\mathrm{acc}(o \rightarrow n)}{\mathrm{acc}(n \rightarrow o)} = \exp\{-\beta[\mathcal{U}(n) - \mathcal{U}(o)]\}. \tag{18}$$

It is straightforward to demonstrate that acceptance rule (17) obeys this condition.

3 Grand-Canonical Ensemble

The ensembles we have discussed so far have the total number of particles imposed. For some systems, however, one would like to obtain information on the average number of particles in a system as a function of the external conditions. For example, in adsorption studies one would like to know the amount of material adsorbed as a function of the pressure and temperature of the reservoir with which the material is in contact. A naive but theoretically valid approach would be to use the Molecular Dynamics technique (microcanonical ensemble) and simulate the experimental situation; an adsorbent in contact with a gas (see Figure 4).

Such a simulation is possible only for very simple systems. In real experiments, equilibration may take minutes or even several hours depending on the type of gas molecules.

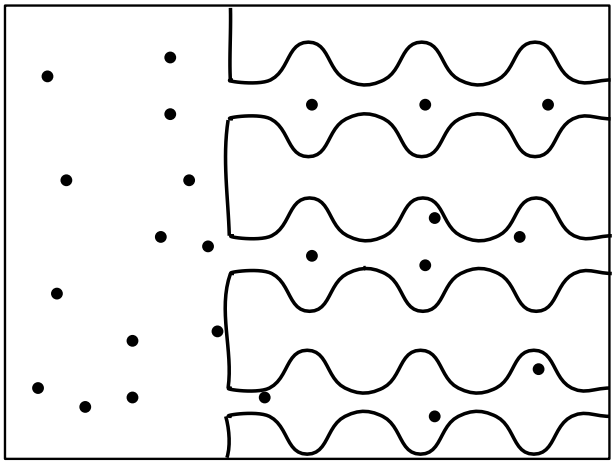

Figure 4. Adsorbent (for example a zeolite) in direct contact with a gas).

These equilibration times would be reflected in a Molecular Dynamics simulation, the difference being that a minute of experimental time takes of the order of 10^9 seconds on a computer. Furthermore, in most cases, we are not interested in the properties of the gas phase, yet a significant amount of CPU time will be spent on the simulation of this phase. Finally, in such a simulation, there is an interface between the gas phase and the adsorbent. In the interfacial region the properties of the system are different from the bulk properties in which we are interested. Since in a simulation the system is relatively small, we have to simulate a very large system to minimize the influence of this interfacial region.[c]

Most of these problems can be solved by a careful choice of ensembles. For adsorption studies, a natural ensemble to use is the grand-canonical ensemble (or μ,V,T ensemble). In this ensemble, the temperature, volume, and chemical potential are fixed. In the experimental setup, the adsorbed gas is in equilibrium with the gas in the reservoir. The equilibrium conditions are that the temperature and chemical potential of the gas inside and outside the adsorbent must be equal.[d] The gas that is in contact with the adsorbent can be considered as a reservoir that imposes a temperature and chemical potential on the adsorbed gas (see Figure 5).

We therefore have to know only the temperature and chemical potential of this reservoir to determine the equilibrium concentration inside the adsorbent. This is exactly what is mimicked in the grand-canonical ensemble: the temperature and chemical potential are imposed and the number of particles is allowed to fluctuate during the simulation. This makes these simulations different from the conventional ensembles, where the number of molecules is fixed.

[c] Such a simulation, of course, would be appropriate if the interest is in just this region.

[d] Note that the pressure is *not* defined inside the zeolite; therefore, the pressure cannot be an equilibrium quantity. However, the pressure is related to the chemical potential via an equation of state, and it is always possible to calculate the pressure of the gas that corresponds to a given chemical potential and vice versa.

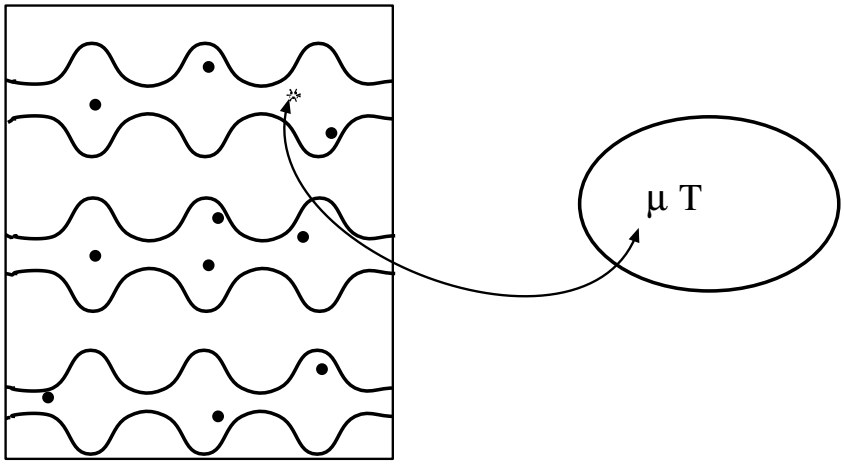

Figure 5. Adsorbent in contact with a reservoir that imposes constant chemical potential and temperature by exchanging particles and energy).

3.1 Statistical Mechanical Basis

The Metropolis sampling scheme as a method to compute thermal averages of functions $A(\mathbf{r}^N)$ that depend explicitly on the coordinates of the molecules in the N-body system under study. Examples of such mechanical properties are the potential energy or the virial contribution to the pressure. However, the Metropolis method could not be used to determine the integral $\int \mathrm{d}\mathbf{r}^N \exp[-\beta\mathcal{U}(\mathbf{r}^N)]$ itself. The latter quantity measures the effective volume in configuration space that is accessible to the system. Hence, the original Metropolis scheme could not be used to determine those thermodynamic properties of a system that depend explicitly on the configurational integral. Examples of such thermal properties are the Helmholtz free energy F, the entropy S, and the Gibbs free energy G. However, although the Metropolis method cannot be used to measure, for instance, free energies directly, it can be used to measure the difference in free energy between two possible states of an N-body system. This fact is exploited in the grand-canonical Monte Carlo method first implemented for classical fluids by Norman and Filinov[23], and later extended and improved by a number of other groups[24,27–34]. The basic idea of the grand-canonical Monte Carlo method is explained next.

To understand the statistical mechanical basis for the grand-canonical Monte Carlo technique, let us return to consider the partition fucntion of a combined system of N interacting particles in volume V and $M-N$ ideal gas molecules in volume $V_0 - V$:

$$Q(N, M, V, V_0, T) = \frac{V^N (V_0 - V)^{M-N}}{\Lambda^{3M} N!(M-N)!} \int \mathrm{d}\mathbf{s}^{M-N} \int \mathrm{d}\mathbf{s}^N \exp[-\beta\mathcal{U}(\mathbf{s}^N)].$$

Now, instead of allowing the two systems to exchange volume, let us see what happens if the systems can also exchange particles (see Figure 6).

To be more precise, we assume that the molecules in the two subvolumes are actually identical particles. The only difference is that when they find themselves in volume V, they interact and, when they are in volume $V_0 - V$, they do not. If we transfer a molecule

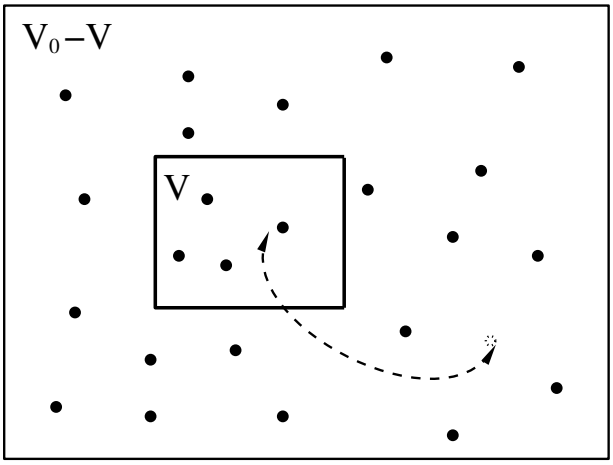

Figure 6. Ideal gas ($M-N$ particles, volume V_0-V) can exchange particles with a N-particle system (volume V).

i from a reduced coordinate $\mathbf{s}_i$ in the volume $V_0 - V$ to the same reduced coordinate in volume V, then the potential energy function $\mathcal{U}$ changes from $\mathcal{U}(\mathbf{s}^N)$ to $\mathcal{U}(\mathbf{s}^{N+1})$. The expression for the total partition function of the system, including all possible distributions of the M particles over the two subvolumes is

$$Q(M,V,V_0,T) = \sum_{N=0}^{M} \frac{V^N (V_0-V)^{M-N}}{\Lambda^{3M} N!(M-N)!} \int \mathrm{d}\mathbf{s}^{M-N} \int \mathrm{d}\mathbf{s}^N \exp[-\beta \mathcal{U}(\mathbf{s}^N)]. \quad (19)$$

We now write the probability density to find a system with $M-N$ particles at reduced coordinates $\mathbf{s}^{M-N}$ in volume $V' \equiv V_0 - V$ and N particles at reduced coordinates $\mathbf{s}^N$ in volume V:

$$\mathcal{N}(\mathbf{s}^M; N) = \frac{V^N V'^{M-N}}{Q(M,V,V',T)\Lambda^{3M} N!(M-N)!} \exp[-\beta \mathcal{U}(\mathbf{s}^N)]. \quad (20)$$

Let us now consider a trial move in which a particle is transferred from V' to the same scaled coordinate in V. First we should make sure that we construct an underlying Markov chain that is symmetric. Symmetry, in this case, implies that the a priori probability to move a particle from V' to V should be equal to the a priori probability of the reverse move. The probability of acceptance of a trial move in which we move a particle to or from volume V is determined by the ratio of the corresponding probability densities (20):

$$\alpha(N \to N+1) = \frac{V(M-N)}{V'(N+1)} \exp(-\beta[\mathcal{U}(\mathbf{s}^{N+1}) - \mathcal{U}(\mathbf{s}^N)]) \quad (21)$$

$$\alpha(N+1 \to N) = \frac{V'(N+1)}{V(M-N)} \exp(-\beta[\mathcal{U}(\mathbf{s}^{N}) - \mathcal{U}(\mathbf{s}^{N+1})]). \quad (22)$$

Now let us consider the limit that the ideal gas system is very much larger than the interacting system: $M \to \infty, V' \to \infty, (M/V') \to \rho$. Note that for an ideal gas the chemical

potential μ is related to the particle density ρ by

$$\mu = k_B T \ln \Lambda^3 \rho.$$

Therefore, in the limit $(M/N) \to \infty$, the partition function (19) becomes

$$Q(\mu, V, T) \equiv \sum_{N=0}^{\infty} \frac{\exp(\beta\mu N) V^N}{\Lambda^{3N} N!} \int \mathrm{d}\mathbf{s}^N \exp[-\beta \mathcal{U}(\mathbf{s}^N)], \tag{23}$$

and the corresponding probability density

$$\mathcal{N}_{\mu VT}(\mathbf{s}^N; N) \propto \frac{\exp(\beta\mu N) V^N}{\Lambda^{3N} N!} \exp[-\beta \mathcal{U}(\mathbf{s}^N)]. \tag{24}$$

Equations 23 and 24 are the basic equations for Monte Carlo simulations in the grand-canonical ensemble. Note that, in these equations, all explicit reference to the ideal gas system has disappeared.

3.2 Monte Carlo Simulations

In a grand-canonical simulation, we have to sample the distribution (24). Acceptable trial moves are

1. *Displacement of particles.* A particle is selected at random and given a new conformation (for example in the case of atoms a random displacement). This move is accepted with a probability

$$\mathrm{acc}(s \to s') = \min\left(1, \exp\{-\beta[\mathcal{U}(\mathbf{s}'^N) - \mathcal{U}(\mathbf{s}^N)]\}\right). \tag{25}$$

2. *Insertion and removal of particles.* A particle is inserted at a random position or a randomly selected particle is removed. The creation of a particle is accepted with a probability

$$\mathrm{acc}(N \to N+1) = \min\left[1, \frac{V}{\Lambda^3 (N+1)} \exp\{\beta[\mu - \mathcal{U}(N+1) + \mathcal{U}(N)]\}\right] \tag{26}$$

and the removal of a particle is accepted with a probability

$$\mathrm{acc}(N \to N-1) = \min\left[1, \frac{\Lambda^3 N}{V} \exp\{-\beta[\mu + \mathcal{U}(N-1) - \mathcal{U}(N)]\}\right]. \tag{27}$$

The chemical potential of the reservoir can be related to the pressure of the reservoir. Algorithm 3 shows the basic structure of a simulation in the grand-canonical ensemble.

3.3 Justification of the Algorithm

It is instructive to demonstrate that the acceptance rules (25)–(27) indeed lead to a sampling of distribution (24). Consider a move in which we start with a configuration with N particles and move to a configuration with $N+1$ particles by inserting a particle in the system. Recall than we have to demonstrate that detailed balance is obeyed:

$$K(N \to N+1) = K(N+1 \to N),$$

Algorithm 3
[Basic Grand-Canonical Ensemble Simulation]

```
 PROGRAM mc_gc                          basic μVT-ensemble
                                        simulation
 do icycl=1,ncycl                       perform ncycl MC cycles
  ran=int(ranf()*(npart+nexc))+1
  if (ran.le.npart) then
    call mcmove                         displace a particle
  else
    call mcexc                          exchange a particle
  endif                                 with the reservoir
  if (mod(icycl,nsamp).eq.0)
+   call sample                         sample averages
 enddo
 end
```

Comments:

1. This algorithm ensures that, after each MC step, detailed balance is obeyed. Per cycle we perform on average `npart` attempts to displace particles and `nexc` attempts to exchange particles with the reservoir.

2. Subroutine `mcmove` attempts to displace a particle (Algorithm 2 at page 36), subroutine `mcexc` attempts to exchange a particle with a reservoir (Algorithm 4), and subroutine `sample` samples quantities every `nsamp` cycle.

with

$$K(N \to N+1) = \mathcal{N}(N) \times \alpha(N \to N+1) \times \mathrm{acc}(N \to N+1).$$

In Algorithm 3 at each Monte Carlo step the probability that an attempt is made to remove a particle is equal to the probability of attempting to add one:

$$\alpha_{\mathrm{gen}}(N \to N+1) = \alpha_{\mathrm{gen}}(N+1 \to N),$$

where the subscript "gen" refers to the fact that α measures the probability to generate this trial move. Substitution of this equation together with equation 24 into the condition of detailed balance gives:

$$\begin{aligned}\frac{\mathrm{acc}(N \to N+1)}{\mathrm{acc}(N+1 \to N)} &= \frac{\exp[\beta\mu(N+1)]V^{N+1}\exp[-\beta\mathcal{U}(\mathbf{s}^{N+1})]}{\Lambda^{3(N+1)}(N+1)!} \\ &\quad \times \frac{\Lambda^{3N}N!\exp[\beta\mathcal{U}(\mathbf{s}^N)]}{\exp(\beta\mu N)V^N} \\ &= \frac{\exp(\beta\mu)V}{\Lambda^3(N+1)}\exp\{-\beta[\mathcal{U}(\mathbf{s}^{N+1}) - \mathcal{U}(\mathbf{s}^N)]\}.\end{aligned}$$

It is straightforward to show that acceptance rules (26) and (27) obey this condition.

Algorithm 4
[Attempt to Exchange a Particle with a Reservoir]

```
 SUBROUTINE mcexc                  attempt to exchange a particle
                                   with a reservoir
 if (ranf().lt.0.5) then           decide to remove or add a particle
   if (npart.eq.0) return          test whether there is a particle
   o=int(npart*ranf())+1           select a particle to be removed
   call ener(x(o),eno)             energy particle o
   arg=npart*exp(beta*eno)         acceptance rule (27)
+       /(zz*vol)
   if (ranf().lt.arg) then
     x(o)=x(npart)                 accepted: remove particle o
     npart=npart-1
   endif
 else
   xn=ranf()*box                   new particle at a random position
   call ener(xn,enn)               energy new particle
   arg=zz*vol*exp(-beta*enn)       acceptance rule (26)
+       /(npart+1)
   if (ranf().lt.arg) then
     x(npart+1)=xn                 accepted: add new particle
     npart=npart+1
   endif
 endif
 return
 end
```

Comment:

1. We have defined: $\mathrm{zz} = \exp(\beta\mu)/\Lambda^3$. The subroutine `ener` calculates the energy of a particle at a given position.

The most salient feature of the grand-canonical Monte Carlo technique is that in such simulations the chemical potential μ is imposed, while the number of particles N is a fluctuating quantity. During the simulation we may measure other thermodynamic quantities, such as the pressure P, the average density $\langle\rho\rangle$, or the internal energy $\langle\mathcal{U}\rangle$. As we know μ, we can derive all other thermal properties, such as the Helmholtz free energy or the entropy. This may seem surprising. After all, the Metropolis sampling cannot be used to sample absolute free energies and related quantities. Yet, with grand-canonical Monte Carlo we seem to be doing precisely that. The answer is that, in fact, we do not. What we measure is not an absolute but a relative free energy. In grand-canonical Monte Carlo, we are equating the chemical potential of a molecule in an ideal gas at density ρ (for the ideal gas case we know how to compute μ) and the chemical potential of the same species in an interacting system at density ρ'.

Grand-canonical Monte Carlo works best if the acceptance of trial moves by which

particles are added or removed is not too low. For atomic fluids, this condition effectively limits the maximum density at which the method can be used to about twice the critical density. Special tricks are needed to extend the grand-canonical Monte Carlo method to somewhat higher densities[32]. Grand-canonical Monte Carlo is easily implemented for mixtures and inhomogeneous systems, such as fluids near interfaces. In fact, some of the most useful applications of the grand-canonical Monte Carlo method are precisely in these areas of research. Although the grand canonical Monte Carlo technique can be applied to simple models of nonspherical molecules, special techniques are required since the method converges very poorly for all but the smallest polyatomic molecules.

4 Beyond Metropolis

4.1 Introduction

The original Metropolis Monte Carlo scheme[6] was designed to perform single-particle trial moves. For most simulations, such moves are perfectly adequate. However, in some cases it is more efficient to perform moves in which the coordinates of many particles are changed. A case in point is the sampling of polymer conformations. The conventional Metropolis algorithm is ill-suited for polymer simulations because the natural dynamics of polymers is dominated by topological constraints (chains cannot cross). Hence, any algorithm that mimics the real motion of macromolecules will suffer from the same problem. For this reason, many algorithms have been proposed to speed up the Monte Carlo sampling of polymer conformations (see *e.g.* ref.[35]). The Configurational-Bias Monte Carlo (CBMC) method is a dynamic MC scheme that makes it possible to achieve large conformational changes in a single trial move that affects a large number of monomeric units[36–39].

4.2 CBMC

The CBMC method is based on the Rosenbluth sampling scheme[41,36,37] for lattice systems. In this scheme, the molecular conformation is built up step-by-step, in such a way that, at every stage, the next monomeric unit is preferentially added in a direction that has a large Boltzmann weight. This increases the probability of generating a trial conformation that has no hard-core overlaps. As explained below, the probability of acceptance of the trial conformation is given by the ratio of the 'Rosenbluth weights' of the new and the old conformations. Whereas the original Rosenbluth scheme was devised for polymers on a lattice, the CBMC scheme will also work for chain molecules in continuous space. The advantage of the CBMC algorithm over many of the other, popular algorithms is that it can be used in cases where particle-insertion and particle-removal trial moves are essential. This is the case in Grand-Canonical and Gibbs-Ensemble simulations. In addition, the CBMC can be used in the simulation of grafted chains and ring polymers.

4.2.1 Detailed balance

Before explaining the CBMC scheme, it is useful to recall the general recipe to construct a Monte Carlo algorithm. It is advisable (although not strictly obligatory[42])to start from the

condition of detailed balance:

$$P_o \times P_{gen}(o \to n) \times P_{acc}(o \to n) = P_n \times P_{gen}(n \to o) \times P_{acc}(n \to o) \ , \qquad (28)$$

where P_o (P_n) is the Boltzmann weight of the old (new) conformation, P_{gen} denotes the *a priori* probability to generate the trial move from o to n, and P_{acc} is the probability that this trial move will be accepted. From Eqn. 28 it follows that

$$\frac{P_{acc}(o \to n)}{P_{acc}(n \to o)} = \exp(-\beta \Delta U) \frac{P_{gen}(n \to o)}{P_{gen}(o \to n)} \ , \qquad (29)$$

where $\exp(-\beta \Delta U)$ is the ratio of the Boltzmann weights of the new and old conformations. If we use the Metropolis rule to decide on the acceptance of MC trial moves, then Eqn 29 implies

$$P_{acc}(o \to n) = \min \left(1, \exp(-\beta \Delta U) \frac{P_{gen}(n \to o)}{P_{gen}(o \to n)} \right) \ . \qquad (30)$$

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 Eqn. 30 always equal to unity. In that case, every trial move will be accepted. This ideal situation can be reached in rare cases[40] Configurational bias Monte Carlo does not achieve this ideal situation. However it does lead to enhanced acceptance probability of trial moves that involve large conformational changes.

In CBMC, chain configurations are generated by successive insertion of the bonded segments of the chain. When the positions of the segments are chosen at random, it is very likely, that one of the segments will overlap with another particle in the fluid, which results in rejection of the trial move. The Rosenbluth sampling scheme increases the insertion probability by looking one step ahead. On lattices, the availability (i.e. the Boltzmann factor) of all sites adjacent to the previous segment can be tested. In continuous space, there are in principle an infinite number of positions that should be tested (e.g. in the case of a chain molecule with rigid bonds, all points on the surface of a sphere with a radius equal to the bond length). Of course, it is not feasible to scan an infinite number of possibilities. Surprisingly, it turns out that it is possible to construct a correct Monte Carlo scheme for off-lattice models in which only a finite number of trial segments (k), is selected either at random or, more generally, drawn from the distribution of bond-lengths and bond-angles of the 'ideal' chain molecule.

During a CBMC trial move, a polymer conformation is generated segment-by-segment. At every step, k trial segments are generated (k is, in principle arbitrary and can be chosen to optimize computational efficiency). One of these segments, say i, is selected with a probability

$$P_i = \frac{\exp(-\beta u_i)}{\sum_{j=1}^{k} \exp(-\beta u_j)}$$

where u_j is the change in the potential energy of the system that would result of this particular trial segment was added to the polymer. The probability of generating a complete conformation Γ_{new} consisting of ℓ segments is then

$$P(\Gamma)_{new} = \prod_{n=1}^{\ell} \frac{\exp(-\beta u_i(n))}{\sum_{j=1}^{k} \exp(-\beta u_j(n))}$$

To keep the equations simple, we only consider the expression for one of the ℓ segments.

$$P_{gen}(\{j\}) = d\Gamma_j P_{id}(\Gamma_j) \left[\prod_{j' \neq j}^{k} d\Gamma_{j'} P_{id}(\Gamma_{j'}) \right] \frac{\exp(-\beta u_{ext}(j))}{\sum_{j'=1}^{k} \exp(-\beta u_{ext}(j'))} \qquad (31)$$

In order to compute the acceptance probability of this move, we have to consider what happens in the reverse move. Then we start from conformation j and generate a set of k trial directions that includes i. When computing the acceptance probability of the forward move, we have to impose detailed balance. However, detailed balance in this case means not just that in equilibrium the number of moves from i to j is equal to the number of reverse move, but even that the rates are equal *for any given set of trial directions for the forward and reverse moves*. This condition we call 'super-detailed balance'. Super-detailed balance implies that we can only decide on the acceptance of the forward move if we also generate a set of $k-1$ trial directions around the old conformation i. We denote the probability to generate this set of $k-1$ trial orientations by $P_{rest}(\{i\})$, where the sub-script 'rest' indicates that this is the set of orientations around, but excluding, i. This allows us to compute the ratio $w_j^{(t)}/w_i^{(o)}$ of the Rosenbluth weights for forward and reverse moves:

$$w_j^{(t)} = \frac{\exp(-\beta u_{ext}^{(t)}(j) + \sum_{j' \neq j}^{k-1} \exp(-\beta u_{ext}^{(t)}(j'))}{k}$$

and

$$w_i^{(o)} = \frac{\exp(-\beta u_{ext}^{(o)}(i) + \sum_{i' \neq i}^{k-1} \exp(-\beta u_{ext}^{(o)}(i'))}{k}$$

The superscript (t) and (o) distinguish the **t**rial conformation from the **o**ld conformation. The acceptance probability is determined by the ratio $x \equiv w_j^{(t)}/w_i^{(o)}$ (actually, for a molecule of ℓ segments, we should compute a product of such factors). Let us assume that $w_j^{(t)} < w_i^{(o)}$. In that case, $P_{acc}(i \to j) = x$ while $P_{acc}(j \to i) = 1$. Next, let us check whether detailed balance is satisfied. To do so, we write down the explicit expressions for K_{ij} and K_{ji}.

$$K_{ij} = N_i P_{gen}(\{j\}) P_{rest}(\{i\}) w_j^{(t)}/w_i^{(o)}$$

and

$$K_{ji} = N_j P_{gen}(\{i\}) P_{rest}(\{j\}) 1$$

In addition, we use the fact that

$$P_{gen}(\{j\}) P_{rest}(\{i\}) w_j^{(t)} \sim N_j P_{rest}(\{j\}) P_{rest}(\{i\})$$

and

$$P_{gen}(\{i\}) P_{rest}(\{j\}) w_j^{(t)} \sim N_i P_{rest}(\{i\}) P_{rest}(\{j\})$$

In then follows immediately that

$$K_{ij} w_i^{(o)} = N_i P_{gen}(\{j\}) P_{rest}(\{i\}) w_j^{(t)} \tag{32}$$

$$= \text{constant} \times N_i N_j P_{rest}(\{i\}) P_{rest}(\{j\})$$

$$= N_j P_{gen}(\{i\}) P_{rest}(\{j\}) w_i^{(o)}$$

$$= K_{ji} w_i^{(o)} \tag{33}$$

Hence, K_{ij} is indeed equal to K_{ji}. Note that, in this derivation, the number of trial directions, k, was arbitrary. The procedure sketched above is valid for a complete regrowth of the chain, but it is also possible to regrow only part of a chain, i.e. to cut a chain at a (randomly chosen) point and regrow the cut part of the chain either at the same site or at the other end of the molecule. Clearly, if only one segment is regrown and only one trial direction is used, CBMC reduces to the reptation algorithm (at least, for linear homopolymers).

We still have to consider the choice for the number of trial directions at the $i-th$ regrowth step, k_i. Too many trial directions increase the cost of a simulation cycle, but too few trial directions lower the acceptance rate, and increase the simulation length. There exist simple guidelines that allow us to select k_i for every segment such that it optimizes the efficiency of the simulation[44].

4.3 Beyond CBMC

The CBMC method has several drawbacks. First of all, as the scheme looks ahead only one step at a time, it is likely to end up in "dead-alleys". Secondly, for long chain molecules, the Rosenbluth weight of the trial conformation tends to become quite small. Hence, much of the computational effort may be wasted on "doomed" configurations.

4.4 DPERM

Grassberger and co-workers[45] have suggested adding two ingredients to the Rosenbluth scheme to improve its efficiency: "pruning" and "enrichment". The basic rationale behind pruning is that it is not useful to spend much computer time on the generation of conformations that have a low Rosenbluth weight. Therefore, it is advantageous to discard ("prune") such irrelevant conformations at an early stage. The idea behind enrichment is to make multiple copies of partially grown chains that have a large statistical weight[45,46], and to continue growing these potentially relevant, chains. The algorithm that combines these two features is called the Pruned-Enriched Rosenbluth Method (PERM). The examples presented by Grassberger and co-workers[47,48] indicate that the PERM approach can be very useful to estimate the thermal equilibrium properties of long polymers. The main limitation of both the original Rosenbluth method and the PERM algorithm is that they are "static" Monte Carlo schemes. Such schemes can simulate single polymer chains very efficiently, but are less suited to simulate systems consisting of many polymer chains: at each step, one would have to simultaneously generate the conformations of all the chains in the system. On the other hand, in a dynamic scheme, one can conveniently choose a new point in phase space by only changing one chain at each step of the algorithm.

Just as CBMC is the "dynamic" version of Rosenbluth sampling. Similarly, one can construct a dynamic version of the PERM algorithm: DPERM[49].

The static PERM algorithm uses the Rosenbluth algorithm to generate the chains except that now pruning and enrichment are added. These ingredients are implemented as follows. At any step of the creation of a chain, if the partial Rosenbluth weight $W(j) = \Pi_{i=1}^{j} w_i$ of a configuration is below a lower threshold $W^{<}(j)$, there is a probability of 50% to terminate the generation of this conformation. If the conformation survives this pruning step, its Rosenbluth weight is doubled $W^{*}(j) = 2 * W(j)$. Enrichment occurs when the partial Rosenbluth weight of a conformation $W(j) = \Pi_{i=1}^{j} w_i$ exceeds an upper threshold $W^{>}(j)$. In that case, k copies of the partial chain are generated, each with a weight $W^{*}(j) = W(j)/k$. All these copies subsequently grow independently (subject to further pruning and enrichment).

The DPERM algorithm is the dynamic generalization of the PERM algorithm. As in the CBMC algorithm, we bias the acceptance of trials conformations to recover a correct Boltzmann sampling of chain conformations.
Thus, starting from an old configuration, we create a trial conformation and calculate the probability to generate it. Starting from the condition for detailed balance, we then derive the expression for the probability to accept or reject a new trial conformation. As we use the Rosenbluth method to generate chains, the probability to grow a particular conformation is :

$$P_{gen}(\text{chain}) = \Pi_{i=1}^{l} \frac{e^{-\beta u^{(i)}(n)}}{w_i} \tag{34}$$

Whenever the re-weighted Rosenbluth partial weight $W^{*}(j)$ of the chain drops below the lower threshold $W^{<}(j)$, the chain has a 50% probability of being deleted. Let us assume that this happens m times. Then, the total probability to generate a *particular* conformation is :

$$P_{gen}(\text{chain}) = \frac{1}{2^m} \Pi_{i=1}^{l} \frac{e^{-\beta u^{(i)}(n)}}{w_i} \tag{35}$$

and the re-weighted Rosenbluth weight of such a chain would be :

$$W^{*}(\text{chain\_new}) = 2^m * W(\text{chain\_new}) \tag{36}$$

$$\text{with } W(\text{chain\_new}) = \Pi_{i=1}^{l} w_i \tag{37}$$

Whenever the Rosenbluth partial weight exceeds the upper threshold, k copies of the chain are created with the Rosenbluth weight $W^{*}(j) = W(j)/k$, which leads to the creation of a set of chains : this is a deterministic procedure. At every stage during the growth of the chain, others chains will branch off. The probability to grow the entire family of chains that is generated in one DPERM move can be written as :

$$P_{gen}(\text{chain\_new}) * P_{gen}(\text{rest\_new}) \tag{38}$$

Where $P_{gen}(\text{rest\_new})$ describes the product of the probabilities involved in generating all the other pieces of chains that branch off from the main chain. If we now call p the number of times the Rosenbluth weight exceeds the upper threshold during the generation of the *given* trial configuration, the probability to generate this *particular* chain is :

$$P_{gen}(\text{chain\_new}) = k^p \Pi_{i=1}^{l} \frac{e^{-\beta u^{(i)}(n)}}{w_i} \tag{39}$$

and its re-weighted Rosenbluth weight is :

$$W^*(\text{chain\_new}) = \frac{1}{k^p} W(\text{chain\_new}) \tag{40}$$

Here k is the number of copies that are created each time the Rosenbluth weight exceeds the upper threshold. In Eqn. 39, the first term of the right hand side, describes the usual probability to generate a given chain following the Rosenbluth method. The factor k^p comes from the fact that the new chain could be any of the chains in the set so that the probability to generate a given chain is multiplied by this term. And we deduce Eqn. 40 from the fact that, each time we make some copies, the Rosenbluth weight is divided by k.

If we now also take into account the possibility that the chain can be pruned, then Eqn. 39 becomes :

$$\begin{aligned} P_{gen}(\text{chain\_new}) &= \frac{k^p}{2^m} \Pi_{i=1}^{l} \frac{e^{-\beta u^{(i)}(n)}}{w_i} \\ &= \frac{k^p}{2^m} \frac{e^{-U(n)}}{W(\text{chain\_new})} \end{aligned} \tag{41}$$

And Eqn. 40 becomes :

$$W^*(\text{chain\_new}) = \frac{2^m}{k^p} W(\text{chain\_new}) \tag{42}$$

Note that Eqn. 41 and Eqn. 42 respectively reduce to Eqn. 35 and Eqn. 36 in the absence of enrichment ($p = 0$) and to Eqn. 39 and Eqn. 40 in the absence of pruning ($m = 0$).

We now choose to select the new trial chain from the set of chains created by the DPERM move with a probability given by :

$$P_{\text{choose\_new}} = W^*(\text{chain\_new}) / W_{total}(\text{new}) \tag{43}$$

where $W^*(\text{chain\_new})$ is the re-weighted Rosenbluth weight mentioned in Eqn. 42 and W_{total} is the sum of all such weights

$$W_{total}(\text{new}) = \sum_{set} W^*_{\text{chain}} \tag{44}$$

Eqn. 43 implies that we are most likely to choose the best chain (the one with the largest re-weighted Rosenbluth weight) of the set as the next Monte Carlo trial conformation.
Assuming that we start from an old configuration denoted by the subscript *old*, we generate a new configuration following the scheme described above and, we accept this move with the following acceptance rule :

$$acc(\text{old} \rightarrow \text{new}) = min(1, \frac{W_{\text{total}}(\text{new})}{W_{total}(\text{old})}) \tag{45}$$

To calculate $W_{total}(old)$, one has to "retrace" the old chain : the chain is first clear and reconstructed following the procedure described above to determine its weight. This is exactly analogous to what is done in the configurational-bias Monte Carlo scheme.

The demonstration that this scheme satisfies detailed balance is given in Ref.[49]

This algorithm contains even more free parameters than CBMC. We choose them to optimize computational efficiency. In practice, this usually means that we wish to generate "enough" potentially successful trial chains, but not too many. Typically, the number of chains that survive at the end of a trial move should be $\mathcal{O}(1)$.

4.5 Recoil-growth

The recoil growth (RG) scheme is a dynamic Monte Carlo algorithm that was also developed with the dead-alley problem in mind[50,51]. The algorithm is related to earlier static MC schemes due to Meirovitch[52] and Alexandrowicz and Wilding[53]. 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 alley. 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. 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.

The RG algorithm is best explained by considering a totally impractical, but conceptually simple schemer that has the same effect. 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 that trial position b is open is given by

$$p_j^{\mathrm{open}}(b) = \min(1, \exp[-\beta u_j(b)]), \tag{46}$$

For hard-core interactions, the decision whether a trial direction is open or closed is unambiguous, as $p_j^{\mathrm{open}}(b)$ is either zero or one. For continuous interactions we compare $p_j^{\mathrm{open}}(b)$ with a random number between 0 and 1. If the random number is less than $p_j^{\mathrm{open}}(b)$, 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 branch will "die" within a specified number of steps (this number is denoted by $l_{\max}$). The algorithm then randomly chooses among the available open branches. As we have only looked a distance $l_{\max}$ 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 Eqn. 46. 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 $b_{\mathrm{i}+1}$ for monomer $i+1$ is generated starting from monomer i. We compute the energy of this trial monomer $u_{\mathrm{i}+1}(b)$ and, using Eqn. 46, we decide whether this position is open or closed. If this direction is closed, we try another trial position, up to a maximum 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 $l_{\max}$ steps, i.e., down to length $i-l_{\max}+1$.

   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 $i-l_{\max}$ 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 $l_{\max}$.

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 $l_{\max}$ steps. We still have to test the remaining $k_{\rm rest} \equiv k-1-k_{\rm c}$ directions. We randomly generate $k_{\rm rest}$ trial positions for monomer $i+1$ and use the recoil growth algorithm to test whether at least one "feeler" of length $l_{\max}$ can be grown in this direction grown (unless $i + l_{\max} > l$; 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 $l_{\max}$ in each of the $k_{\rm rest}$ 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

$$w_i(n) = \frac{m_i(n)}{p_i^{\rm open}(n)}$$

and $p_i^{\rm open}(n)$ is given by Eqn. 46.

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:

$$W(n) = \prod_{i=1}^{\ell} w_i(n) = \prod_{i=1}^{\ell} \frac{m_i(n)}{p_i^{\rm open}(n)}. \tag{47}$$

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 $l_{\max}$:

$$W(o) = \prod_{i=1}^{\ell} w_i(o) = \prod_{i=1}^{\ell} \frac{m_i(o)}{p_i^{\rm open}(o)}.$$

Finally, the new conformation is accepted with a probability:

$$\mathrm{acc}(o \to n) = \min(1, \exp[-\beta U(n)]W(n)/\exp[-\beta U(o)]W(o)), \tag{48}$$

where $U(n)$ and $U(o)$ are the energies of the new and old conformations, respectively. It can easily be demonstrated (see [50,51]) that this scheme generates a Boltzmann distribution of conformations.

5 Relative Merits

A comparison between CBMC and the RG algorithm was made by Consta *et al.*[51], 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. A similar comparison has been made between CBMC and DPERM[49]. For the cases studied, DPERM was no worse than CBMC, but also not much better. However, the fact that pruning can be performed on any chain property (i.e. not necessarily the Rosenbluth weight), may make the method attractive in special cases.

The basic idea behind both DPERM and RG is that these algorithms aim to avoid investing computational effort in the generation of trial moves that are, in the end, rejected. Houdayer[54] has proposed an algorithm that aims to achieve the same. In this algorithm, the trial move is a so-called "wormhole" move, where a polymer grows in one part of the simulation box while it shrinks at its original location. The growth-shrinkage process is carried our using a reptation-like algorithm. This algorithm has the advantage that, even if trial moves to the new state are rejected, the "old" state has also changed. This speeds up relaxation. In addition, the computing effort for the wormhole scheme appears to scale favorably with polymer size for long polymers (namely as N^{n}[55], rather than as $\exp(cN)$). However, for short chains, the existing schemes are almost certainly more efficient (in ref.[54], the comparison with CBMC is made for a particularly inefficient CBMC-parameter choice). For longer chains, the wormhole scheme really should win. However, in that regime, all schemes are extremely costly. Nevertheless, as pointed out in ref.[54], a combination of the various algorithms would probably be more efficient than any one of them alone.

Acknowledgments

The work of the FOM Institute is part of the research program of the "Stichting voor Fundamenteel Onderzoek der Materie" (FOM), which is financially supported by the "Nederlandse Organisatie voor Wetenschappelijk Onderzoek" (NWO).

References

1. D. Frenkel and B. Smit *Understanding Molecular Simulations: from Algorithms to Applications*, Academic Press, San Diego, 2nd edition (2002).
2. G. Ciccotti, D. Frenkel, and I.R. McDonald. *Simulation of Liquids and Solids*. North-Holland, Amsterdam, 1987.
3. J.A. Prins, Onze voorstelling omtrent de bouw van de stof. Inaugural Address, 11-10-1928, Rijksuniversiteit Groningen *Physica*, 8:257, 1928.
4. O.K. Rice. *J. Chem. Phys.*, 12:1, 1944.
5. J.D. Bernal. Bakerian lecture. *Proc. Roy. Soc.*, 280:299, 1964.
6. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.N. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.
7. E. Fermi, J.G. Pasta, and S.M. Ulam. Studies of non-linear problems. *LASL Report*, LA-1940, 1955.

8. B.J. Alder and T.A. Wainwright. In I. Prigogine, editor, *Proceedings of the International Symposium on Statistical Mechanical Theory of Transport Processes (Brussels, 1956)*. Interscience, New York, 1958.
9. J.B. Gibson, A.N. Goland, M. Milgram, and G.H. Vineyard. Dynamics of radiation damage. *Phys. Rev.*, 120:1229–1253, 1960.
10. G.H. Vineyard. Autobiographical remarks of G.H. Vineyard. In P.C. Gehlen, J.R. Beeler, and R.I. Jaffe, editors, *Interatomic Potentials and Simulation of Lattice Defects*, pages xiii–xvi. Plenum, New York, 1972.
11. A. Rahman. Correlations in the motion of atoms in liquid argon. *Phys. Rev.*, 136:A405–A411, 1964.
12. L. Verlet. Computer 'experiments' on classical fluids. i. thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159:98–103, 1967.
13. J.A. Barker and R.O. Watts. Structure of water: A Monte Carlo calculation. *Chem. Phys. Lett.*, 3:144–145, 1969.
14. I.R. McDonald and K. Singer. Calculation of the thermodynamic properties of liquid argon from Lennard-Jones parameters by a Monte Carlo method, *Discuss. Faraday Soc.*, 43:40, 1967.
15. W.W. Wood. Early history of computer simulation in statistical mechanics. In G. Ciccotti and W.G. Hoover, editors, *Molecular Dynamics Simulations of Statistical Mechanics Systems*, pages 2–14. Proceedings of the 97th Int. "Enrico Fermi" School of Physics, North Holland, Amsterdam, 1986.
16. N.G. van Kampen. *Stochastic Processes in Physics and Chemistry*. North-Holland, Amsterdam, 1981.
17. M.P. Allen and D.J. Tildesley. *Computer Simulation of Liquids*. Clarendon Press, Oxford, 1987.
18. J.J. Erpenbeck and W.W. Wood. Molecular dynamics techniques for hard-core systems. In B.J. Berne, editor, *Statistical Mechanics, Part B*, pages 1–40. Plenum, New York, 1976.
19. J.L. Lebowitz, J.K. Percus, and L. Verlet. Ensemble dependence of fluctuations with application to machine computations. *Phys. Rev.*, 153:250–254, 1967.
20. W.W. Wood. Monte Carlo calculations for hard disks in the isothermal-isobaric ensemble. *J. Chem. Phys.*, 48:415–434, 1968.
21. I.R. McDonald. NpT-ensemble Monte Carlo calculations for binary liquid mixtures. *Mol. Phys.*, 23:41–58, 1972.
22. R. Najafabadi and S. Yip. Observation of finite-temperature strain transformation (f.c.c. $\leftrightarrow$ b.c.c.) in Monte Carlo simulation of iron. *Scripta Metall.*, 17:1199–1204, 1983.
23. G.E. Norman and V.S Filinov. Investigation of phase transitions by a Monte-Carlo method. *High Temp. (USSR)*, 7:216–222, 1969.
24. D.J. Adams. Chemical potential of hard-sphere fluids by Monte Carlo methods. *Mol. Phys.*, 28:1241–1252, 1974.
25. M. Creutz. Microcanonical Monte Carlo simulation. *Phys. Rev. Lett.*, 50:1411–1414, 1983.
26. A.Z. Panagiotopoulos. Direct determination of phase coexistence properties of fluids by Monte Carlo simulation in a new ensemble. *Mol. Phys.*, 61:813–826, 1987.
27. D.J. Adams. Grand canonical ensemble Monte Carlo for a Lennard-Jones fluid. *Mol.*

Phys., 29:307–311, 1975.
28. D.J. Adams. Calculating the low temperature vapour line by Monte Carlo. *Mol. Phys.*, 32:647–657, 1976.
29. D.J. Adams. Calculating the high-temperature vapour line by Monte Carlo. *Mol. Phys.*, 37:211–221, 1979.
30. L.A. Rowley, D. Nicholson, and N.G. Parsonage. Monte Carlo grand canonical ensemble calculation in a gas-liquid transition region for 12-6 argon. *J. Comp. Phys.*, 17:401–414, 1975.
31. J. Yao, R.A. Greenkorn, and K.C. Chao. Monte Carlo simulation of the grand canonical ensemble. *Mol. Phys.*, 46:587–594, 1982.
32. M. Mezei. A cavity-biased (T,V,μ) Monte Carlo method for the computer simulation of fluids. *Mol. Phys.*, 40:901–906, 1980.
33. J.P. Valleau and L.K. Cohen. Primitive model electrolytes. I. Grand canonical Monte Carlo computations. *J. Chem. Phys.*, 72:5935–5941, 1980.
34. W. van Megen and I.K. Snook. The grand canonical ensemble Monte Carlo method applied to the electrical double layer. *J. Chem. Phys.*, 73:4656–4662, 1980.
35. Monte Carlo and Molecular Dynamics Simulations in Polymer Science Edited by K. Binder, OUP, Oxford, 1995.
36. J. Harris and S.A. Rice, A lattice model of a supported monolayer of amphiphilic molecules: Monte Carlo simulations *J. Chem. Phys.* **88**, 1298–1306 (1988).
37. J.I. Siepmann and D. Frenkel, Configurational-bias Monte Carlo: a new sampling scheme for flexible chains *Mol.Phys.* **75**, 59–70 (1992).
38. D. Frenkel, G.C.A.M. Mooij, and B. Smit, Novel scheme to study structural and thermal properties of continuously deformable molecules *J.Phys. Condensed Matter* **4**,3053–3076 (1992).
39. J.J. de Pablo, M. Laso, and Suter U.W, Simulation of polyethylene above and below the melting point *J. Chem. Phys.* **96**, 2395–2403 (1992).
40. R.H Swendsen and J.S. Wang, Nonuniversal critical dynamics in Monte Carlo simulations, *Phys. Rev. Lett.* **58**, 86–88 (1987).
41. M.N. Rosenbluth and A.W. Rosenbluth, Monte Carlo simulations of the average extension of molecular chains *J.Chem. Phys.* **23**, 356–359 (1955).
42. V.I. Manousiouthakis and M.W. Deem, Strict Detailed Balance is Unnecessary in Monte Carlo Simulation *J. Chem. Phys.*, **110**,2753–2758(1999).
43. G.C.A.M. Mooij, D. Frenkel, and B. Smit, Direct simulation of phase equilibria of chain molecules *J. Phys. Condensed Matter* **4**, L255–L259 (1992).
44. G.C.A.M. Mooij and D.Frenkel, A systematic optimization scheme for configurational bias Monte Carlo *Molecular Simulation* **17**,41–55(1996).
45. P. Grassberger, Pruned-enriched Rosenbluth method: Simulations of theta polymers of chain length up to 1,000,000 *Phys. Rev. E* **56**, 3682–3693 (1997).
46. U. Bastolla, H. Frauenkron, E. Gerstner, P. Grassberger, and W. Nadler, Testing a new Monte Carlo algorithm for protein folding *Proteins: Struc. Func. Gen.* **32**,52–66 (1998).
47. H. Frauenkron, U. Bastolla, E. Gerstner, P. Grassberger, and W. Nadler, New Monte Carlo Algorithm for Protein Folding *Phys. Rev. Lett.* **80**,3149–3152 (1998).
48. U. Bastolla and P. Grassberger, **xxx** *J. Stat. Phys.* **89**,1061 (1997).
49. N. Combe, T.J.H.Vlugt. P.R. ten Wolde and D. Frenkel, Dynamic pruned-enriched

Rosenbluth method *Mol. Phys.* **101**, 16751682 (2003).
50. S.Consta, T.J.H. Vlugt, J.Wichers Hoeth, B.Smit, and D.Frenkel, Recoil growth algorithm for chain molecules with continuous interactions *Mol. Phys.* **97**,1243–1254(1999).
51. S. Consta, N.B. Wilding, D. Frenkel, and Z. Alexandrowicz, Recoil growth: An efficient simulation method for multi-polymer systems *J. Chem. Phys.* **110**,3220–3228(1999).
52. H. Meirovitch, Statistical Properties Of The Scanning Simulation Method For Polymer-Chains *J. Chem. Phys.* **89**,2514–2522,(1988).
53. Z. Alexandrowicz and N.B. Wilding, Simulation of polymers with rebound selection *J. Chem. Phys.* **109**,5622–5626(1998).
54. J. Houdayer, The wormhole move: A new algorithm for polymer simulations *J. Chem. Phys.* **116**, 1783–1787(2002).
55. Houdayer[54] suggests that the computing time per accepted move should scale as N^2. However, as the problem is asymptotically that of a one-dimensional random walk with absorbing boundaries, it is more plausible that, for very long chains, the computing time per accepted move should scale as N^3.