Preface

Although two decades ago molecular modeling and simulation of biomolecules were in the realm of specialists with access to supercomputers, ongoing improvements in force fields and powerful software readily available to the academic community have stimulated a great interest among bioscientists who are primarily interested in investigating biological or chemical problems. This development has been accompanied by a decrease in the price/performance ratio of hardware that enables us to carry out meaningful simulations on desktop workstations or small clusters of workstations. For example, all-atom models of a protein in a lipid membrane with water molecules included can be simulated for a few nanoseconds.

The purpose of *Molecular Modeling of Proteins* is to enable nonspecialists, first, to grasp the scope of methods available and, second, to apply methods easily to their own problems. Although software packages in molecular modeling are accompanied by good manuals, the first-time user may easily be frustrated over a problem that requires only a small tweak of an input file to solve. Thus, most chapters contain, apart from a thorough introduction, step-by-step instructions and notes on troubleshooting and hints about how to avoid pitfalls.

The first part of the book describes the methodologies of molecular modeling including a chapter about normal modes and essential dynamics. This part contains, apart from practical hints and tips, a thorough treatment of the underlying theories. The next part focuses on free energy calculations, followed by various chapters about the molecular modeling of membrane proteins. A later part contains chapters about protein structure determination by comparative protein modeling as well as modeling based on experimental data. A further part is devoted to the conformational changes of proteins, and protein folding and unfolding and misfolding in prion diseases. The last part contains several chapters about applications to drug design. The topics have been chosen to represent the latest developments in the field, albeit highly relevant to biochemical and biomedical problems. Although this book is directed at the modeling of proteins, the techniques described are equally applicable to other biomolecules, such as DNA or carbohydrates, provided the adequate force

fields are used. The chapters are written by internationally well-established investigators; they include leading developers of popular simulation packages or force fields.

Molecular Modeling of Proteins is directed to scientists in chemistry, biochemistry, biology, biophysics, and bioinformatics working in industry and academia, who are interested in applying the techniques described to their own research. Additionally, the book forms a valuable resource for educators who wish to teach courses to university students or professionals about molecular modeling.

Andreas Kukol

Chapter 2
Monte Carlo Simulations

David J. Earl and Michael W. Deem

Summary A description of Monte Carlo methods for simulation of proteins is given. Advantages and disadvantages of the Monte Carlo approach are presented. The theoretical basis for calculating equilibrium properties of biological molecules by the Monte Carlo method is presented. Some of the standard and some of the more recent ways of performing Monte Carlo on proteins are presented. A discussion of the estimation of errors in properties calculated by Monte Carlo is given.

Keywords: Markov chain · Metropolis algorithm · Monte Carlo · Protein simulation · Stochastic methods

1 Introduction

The term Monte Carlo generally applies to all simulations that use stochastic methods to generate new configurations of a system of interest. In the context of molecular simulation, specifically, the simulation of proteins, Monte Carlo refers to importance sampling, which we describe in Sect. 2, of systems at equilibrium. In general, a Monte Carlo simulation will proceed as follows: starting from an initial configuration of particles in a system, a Monte Carlo move is attempted that changes the configuration of the particles. This move is accepted or rejected based on an *acceptance criterion* that guarantees that configurations are sampled in the simulation from a statistical mechanics ensemble distribution, and that the configurations are sampled with the correct weight. After the acceptance or rejection of a move, one calculates the value of a property of interest, and, after many such moves, an accurate average value of this property can be obtained. With the application of statistical mechanics, it is possible to calculate the equilibrium thermodynamic properties of the system of interest in this way. One important necessary condition for Monte Carlo simulations is that the scheme used must be *ergodic*, namely, every point that is accessible in configuration space should be able to be reached from any other point in configuration space in a finite number of Monte Carlo moves.

From: Methods in Molecular Biology, vol. 443, Molecular Modeling of Proteins
Edited by Andreas Kukol

1.1 Advantages of Monte Carlo

Unlike molecular dynamics simulations, Monte Carlo simulations are free from the restrictions of solving Newton's equations of motion. This freedom allows for cleverness in the proposal of moves that generate trial configurations within the statistical mechanics ensemble of choice. Although these moves may be nontrivial, they can lead to huge speedups of up to 10^{10} or more in the sampling of equilibrium properties. Specific Monte Carlo moves can also be combined in a simulation allowing the modeler great flexibility in the approach to a specific problem. In addition, Monte Carlo methods are generally easily parallelizable, with some techniques being ideal for use with large CPU clusters.

1.2 Disadvantages of Monte Carlo

Because one does not solve Newton's equations of motion, no dynamical information can be gathered from a traditional Monte Carlo simulation. One of the main difficulties of Monte Carlo simulations of proteins in an explicit solvent is the difficulty of conducting large-scale moves. Any move that significantly alters the internal coordinates of the protein without also moving the solvent particles will likely result in a large overlap of atoms and, thus, the rejection of the trial configuration. Simulations using an implicit solvent do not suffer from these drawbacks, and, therefore, coarse-grained protein models are the most popular systems where Monte Carlo methods are used. There is also no general, good, freely available program for the Monte Carlo simulation of proteins because the choice of which Monte Carlo moves to use, and the rates at which they are attempted, vary for the specific problem one is interested in, although we note that a Monte Carlo module has recently been added to CHARMM [1].

2 Theoretical Basis for Monte Carlo

The aim of a Monte Carlo simulation is the accurate calculation of equilibrium thermodynamic and physical properties of a system of interest. Let us consider calculating the average value of some property, $\langle A \rangle$. This could be calculated by evaluating the following:

$$\langle A \rangle = \frac{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta U\left(\mathbf{r}^N\right)\right] A\left(\mathbf{r}^N\right)}{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta U\left(\mathbf{r}^N\right)\right]}, \tag{1}$$

where $\beta = 1/k_{\mathrm{B}}T$, U is the potential energy, and $\mathbf{r}^N$ denotes the configuration of an N particle system (i.e., the positions of all N particles). Now, the probability density of finding the system in configuration $\mathbf{r}^N$ is:

$$\rho\left(\mathbf{r}^N\right) = \frac{\exp\left[-\beta U\left(\mathbf{r}^N\right)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta U\left(\mathbf{r}^N\right)\right]}, \tag{2}$$

where the denominator in Eq. 2 is the configurational integral. If one can randomly generate N_{MC} points in configuration space according to Eq. 2, then Eq. 1 can be expressed as:

$$\langle A \rangle \approx \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} A\left(\mathbf{r}_i^N\right). \tag{3}$$

After equilibration of our system of interest, errors in $\langle A \rangle$ scale as $1/\sqrt{N_{MC}}$, as discussed in Sect. 4. The Monte Carlo methods that we outline in Sect. 3 are responsible for generating the N_{MC} points in configuration space in a simulation.

A Monte Carlo algorithm consists of a group of Monte Carlo moves that generate a Markov chain of states. This Markov process has no history dependence, in the sense that new configurations are generated with a probability that depends only on the current configuration and not on any previous configurations. If our system is currently in state m, then the probability of moving to a state n is defined as π_{mn}, where π is the transition matrix. Let us introduce a probability vector, ρ, that defines the probability that the system is in a particular state; ρ_i is the probability of being in state i. The initial probability vector, for a randomly chosen starting configuration, is $\rho^{(0)}$ and the probability vector for subsequent points in the simulation is $\rho^{(j)} = \rho^{(j-1)}\pi$. The equilibrium, limiting distribution of the Markov chain, ρ^*, results from applying the transition matrix an infinite number of times, $\rho^* = \lim_{N_{MC}\to\infty} \rho^{(0)}\pi^{N_{MC}}$. We note:

$$\rho^* = \rho^*\pi. \tag{4}$$

If we are simulating in the canonical ensemble, then ρ^* is reached when ρ_i is equal to the Boltzmann factor (Eq. 4) for all states. Thus, once a system has reached equilibrium, any subsequent Monte Carlo moves leave the system in equilibrium. The initial starting configuration of the system should not matter as long as the system is simulated for a sufficient number of Monte Carlo steps, N_{MC}. For the simulation to converge to the limiting distribution, the Monte Carlo moves used must satisfy the balance condition and they must result in ergodic sampling [2]. If the transition matrix satisfies Eq. 4, then the balance condition is met. For the stricter condition of detailed balance to be satisfied, the net flux between two states must be zero at equilibrium, i.e.:

$$\rho_m \pi_{mn} = \rho_n \pi_{nm}. \tag{5}$$

For all of the Monte Carlo moves we present in Sect. 3, the balance or detailed balance conditions hold.

Now, how is the transition matrix chosen, such that balance or detailed balance is satisfied? In other words, how do we propose a Monte Carlo move and correctly choose whether to accept or reject it? As an example, let us consider the Metropolis

acceptance criterion. The transition matrix between states m and n can be written as:

$$\pi_{mn} = \alpha_{mn} p_{mn}, \tag{6}$$

where α_{mn} is the probability of proposing a move between the two states and p_{mn} is the probability of accepting the move. Assuming that $\alpha_{mn} = \alpha_{nm}$, as is the case for many but not all Monte Carlo moves, then Metropolis proposed the following scheme: if the new state (n) is lower in energy than the old state (m), then accept the move (i.e., $p_{mn} = 1$ and $\pi_{mn} = \alpha_{mn}$ for $\rho_n \geq \rho_m$), or, if the new state is higher in energy than the old state, then accept the move with $p_{mn} = \exp(-\beta [U(n) - U(m)]) < 1$; i.e., $\pi_{mn} = \alpha_{mn} (\rho_n / \rho_m)$ for $\rho_n < \rho_m$, where the energy of state i is $U(i)$. If the old and new states are the same, then the transition matrix is given by $\pi_{mm} = 1 - \sum_{n \neq m} \pi_{mn}$. The Metropolis *acceptance criterion* [3,4] can be summarized as:

$$p_{mn} = \min \{1, \exp(-\beta [U(n) - U(m)])\}. \tag{7}$$

In a computer simulation, for the case where $U(n)$ is greater than $U(m)$, one generates a pseudorandom number with a value between 0 and 1, and if this number is less than p_{mn}, then the trial move is accepted.

Monte Carlo simulations can be conducted in several different statistical mechanics ensembles, and the distribution that we sample from depends on the ensemble. These ensembles include, but are not limited to, the canonical ensemble (constant number of particles N, volume V, and temperature T), the isobaric–isothermal ensemble (constant N, pressure P, and T), the grand canonical ensemble (constant chemical potential μ, V, T), and the semigrand canonical ensemble. For simulations of proteins, the modeler is unlikely to stray too often from the canonical ensemble where the partition function is:

$$Q(N, V, T) \equiv \prod_{i=1}^{N} \frac{1}{\Lambda_i^3 N!} \int \mathrm{d}\mathbf{r}^N \exp\left[-\beta U\left(\mathbf{r}^N\right)\right], \tag{8}$$

where Λ_i is the thermal de Broglie wavelength, $\Lambda_i = \sqrt{h^2 / (2\pi m_i k_B T)}$ and m_i is the mass of particle i. A thorough description of Monte Carlo simulations in other ensembles can be found elsewhere [5,6].

3 Monte Carlo Methods for Protein Simulation and Analysis

3.1 Standard Monte Carlo Moves

For a molecular system, there are several standard Monte Carlo moves that one can use to explore conformational degrees of freedom. The simplest trial move is to

select at random an atom, i, in the system, and propose a change in its Cartesian coordinates:

$$x_i^{new} = x_i^{old} + \Delta(\chi - 0.5), \tag{9a}$$

$$y_i^{new} = y_i^{old} + \Delta(\chi - 0.5), \tag{9b}$$

$$z_i^{new} = z_i^{old} + \Delta(\chi - 0.5), \tag{9c}$$

where χ is a pseudorandom number between 0 and 1 that is different for each axis at each attempted move, and Δ sets the maximum displacement. After moving the atom, one calculates the energy of the new trial structure, and the new structure is accepted or rejected based on the Metropolis criterion, Eq. 7. For molecules such as proteins, these moves are often not very efficient, because a change in the coordinates of one atom changes the bond lengths between the selected atom and all the atoms to which it is connected. This move will also change any bond angles and torsional angles that the atom is involved in, and, therefore, acceptance of the move is unlikely for all but the very smallest of moves, especially because proteins often have a number of stiff or rigid bonds (or bond angles or torsional angles). As an alternative scheme, one can select individual bond lengths, bond angles, or torsional angles in the molecule to change at random, while keeping all other bond lengths, bond angles, and torsional angles fixed. For bond lengths, bond angles, and torsional angles that are effectively rigid, one can impose constraints on their movement. Torsional angle (or pivot) moves can result in significant changes in the structure of the molecule, and, thus, can be problematic when the protein is surrounded by a solvent. This is because any significant change in the conformation of the protein will result in an overlap with the solvent with a high energetic penalty, and, thus, the rejection of the move. Pivot moves can be very effective in implicit solvent simulations, however.

When considering multiple molecules, one can also use Monte Carlo moves that translate or rotate entire molecules. For flexible, nonlinear molecules such as proteins, the method of quarternions should be used for rotation moves. For translation moves, the center of mass of the molecule in question is simply changed, in the same way as for a single atom move (Eqs. 9a–9c).

3.2 Configurational-Bias Monte Carlo

Configurational-bias Monte Carlo moves fall under the more general category of biased Monte Carlo moves. In these moves, the Monte Carlo trial move is biased toward the proposal of reasonable configurations and, to satisfy detailed balance, the acceptance rules used must be altered because α, in Eq. 6, will no longer be symmetric. Configurational-bias Monte Carlo was originally proposed for the simulation of chain molecules [7] and was later adapted for the simulation of biomolecules [8]. The idea behind the approach is to delete a section of a molecule and to then "regrow" the section. The regrowth of the molecule is biased such that energetically

reasonable conformations are likely to be produced by the method used. Biasing during the regrowth can occur by, for example, restricting torsional angles to reasonable bounds and, if one is regrowing a central section of a molecule, biasing the regrowth toward an area of the simulation box. A configurational-bias Monte Carlo algorithm is composed of the following main steps. First, a section of a molecule is deleted. Second, a trial conformation for the regrowth is generated using a Rosenbluth scheme, and the Rosenbluth weight for the new configuration, $W(n)$, is calculated. Third, the Rosenbluth weight for the old configuration, $W(m)$, is calculated by "retracing" the original conformation. Finally, the trial configuration is accepted with the probability:

$$p_{mn} = \min\left\{1, \frac{W(n)}{W(m)}\right\}. \tag{10}$$

We do not provide details here for determining the Rosenbluth weight, or the generation of trial orientations for the regrowth, of peptide and protein molecules within configurational-bias Monte Carlo methods, because they are lengthy and nontrivial. We instead refer the reader to the following excellent references in which the theory is fully described [6, 7].

3.3 Rebridging and Fixed End Moves

Rebridging, concerted rotation, or end-bridging Monte Carlo moves have been developed to explore the conformational degrees of freedom of internal, backbone portions of polymers, peptides, and proteins [9–12]. An example of a rebridging move for a peptide molecule that causes a local change in conformation but leaves the positions of the rest of the molecule fixed, is shown in Fig. 1. Torsional angles ϕ_0 and

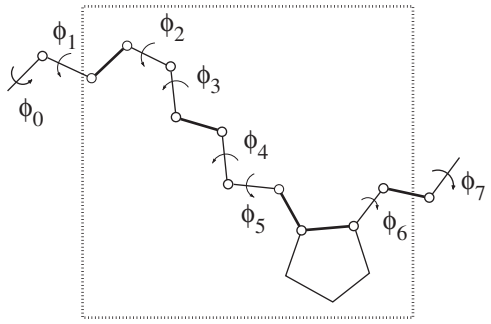

Fig. 1 Driver angles ϕ_0 and ϕ_7 are changed, breaking the connectivity of the molecule in the enclosed region, in an analytic rebridging move. The backbone segment is rebridged in the enclosed region, and the *solid lines* represent pi bonds or rigid molecular fragments within which no rotation is possible. Reprinted from [12] with permission. Copyright American Institute of Physics (1999)

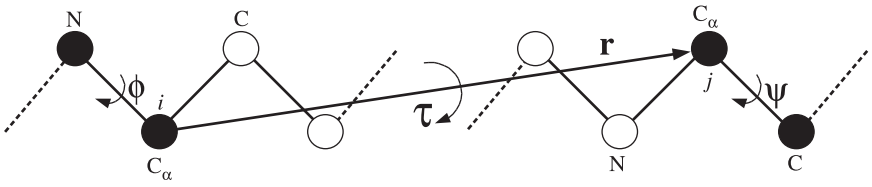

Fig. 2 Fixed end move for biomolecules. The atoms between i and j are rotated about **r** by an angle τ

ϕ_7 are rotated by $\Delta\phi_0$ and $\Delta\phi_7$, causing a change in the units between 0 and 6, and breaking the connectivity of the molecule. In this move, rigid units only, where bond lengths and angles are fixed at a prescribed or equilibrium value, are considered to reduce the situation to a geometric problem, and it is possible to determine all of the solutions that reinsert the backbone in a valid, connected way. Where there are side chains attached to the backbone, they are considered to be rigidly rotated. To satisfy detailed balance, solutions for new and old configurations are determined, and the maximum number of solutions can be shown to be limited to 16. One of these solutions is randomly selected, and the new conformation is accepted or rejected based on a modified acceptance criterion.

An alternative to the rebridging move is a fixed end move, shown in Fig. 2, in which a rigid rotation, τ, along the C_α ends of an arbitrary length of the backbone is attempted. This move changes two torsional angles, ϕ and ψ, as well as the bond angle at each end of the segment, but keeps the other internal angles and torsional angles in the molecule fixed. The rigid rotation, τ, is chosen such that the bond angles at the end of the segment do not vary by more than 10° from their natural range. The relations between the torsional angles, the bond angles, and τ are nonlinear, but it has been shown that trial moves can be proposed that satisfy detailed balance and are effective in equilibrating peptide molecules [13].

3.4 Hybrid Monte Carlo

A hybrid Monte Carlo move involves running a molecular dynamics simulation for a fixed length of time [14]. The configuration at the end of this molecular dynamics run is accepted or rejected based on the Metropolis criterion (Eq. 7), where $U(m)$ is the potential energy before the molecular dynamics run, and $U(n)$ is the potential energy at the end of the run [15]. Thus, molecular dynamics can be used to propose a Monte Carlo move that involves the collective motion of many particles. An advantage that this method has over conventional molecular dynamics simulations is that large time steps may be used in the hybrid Monte Carlo move, because there is no need to conserve energy during the molecular dynamics run. As long as the molecular dynamics algorithm that is used is time reversible and area preserving, then it is acceptable for use in a hybrid Monte Carlo move. Indeed, several very suitable

multiple-time-step molecular dynamics algorithms now exist that are ideal for use in hybrid Monte Carlo moves, including those proposed by Martyna, Tuckerman, and coworkers [16].

At the start of each new hybrid MC move, particle velocities can be assigned at random from a Maxwell distribution, or alternatively they can be biased toward particle velocities from previous, successful hybrid Monte Carlo steps, with, of course, a modified acceptance criterion. The total length of time one uses for the molecular dynamics run is at the discretion of the simulator. However, the general rule of not using too long or short a run applies, because long runs that are not accepted are wasteful of CPU time, and short runs do not advance the configuration through phase space quickly.

3.5 Parallel Tempering

In the parallel tempering method [17], one simulates M replicas of an original system of interest. These replicas are typically each at a different temperature. The general idea of the method is that the higher-temperature replicas are able to access large volumes of phase space, whereas lower-temperature systems may become trapped in local energy minima during the timescales of a typical computer simulation. By allowing configuration swaps between different (typically adjacent in temperature) replicas, the lower-temperature systems can access a representative set of low-energy regions of phase space. The partition function for the M replica system in the canonical ensemble is:

$$Q \equiv \prod_{i=1}^{M} \frac{q_i}{N!} \int \mathrm{d}\mathbf{r}_i^N \exp\left[-\beta_i U\left(\mathbf{r}_i^N\right)\right], \tag{11}$$

where $q_i = \prod_{j=1}^{N} \left(2\pi m_j k_B T_i / h^2\right)^{3/2}$ comes from integrating out the momenta, and $\beta_i = 1/k_B T_i$ is the reciprocal temperature of the replica. If the probability of performing a swap move is equal for all conditions, then exchanges of configurations between replicas i and j are accepted with the probability:

$$p_{ij} = \min\left\langle 1, \exp\left\{+\left[\beta_i - \beta_j\right]\left[U\left(\mathbf{r}_i^N\right) - U\left(\mathbf{r}_j^N\right)\right]\right\}\right\rangle \tag{12}$$

A rather complex issue when using parallel tempering is how many replicas one should use and what temperatures each replica should be set at. As with most Monte Carlo moves, there are no hard and set rules. The highest temperature must be high enough for the simulation to pass over all of the energy barriers in phase space in a manageable computational time, and there must be a sufficient number of replicas to allow a reasonable probability of acceptance for parallel tempering moves, allowing each system to sample both high and low temperatures. To maximize the efficiency of parallel tempering simulations, recent theoretical [18] and numerical

studies [19] have recommended "tuning" the temperature intervals such that moves between replicas are accepted with a probability of 20 to 25%. An adaptive approach based on the total "round-trip" time between high and low temperature replicas seems to be well founded for tuning the temperature intervals [20].

With appropriate scheduling, parallel tempering is an ideal approach to use with large CPU clusters [21], and its effectiveness in the simulation of biomolecules, as well as in solid state and material modeling, is well proven [22].

3.6 Density of States Methods

The canonical partition function can also be written as:

$$Q\,(N, V, T) \equiv \sum_{E_i} g(N, V, E_i) \exp\left[-\beta E_i\right], \tag{13}$$

where g is the density of states. Wang and Landau [23] proposed a Monte Carlo scheme whereby one performs a random walk in energy space, while keeping a running estimate of the inverse of the density of states as a function of the energy, or, alternatively, collecting the configurational temperature as a function of the energy and determining the density of states by integrating the inverse temperature. Although density of states methods do not satisfy detailed balance, it has been proven that they asymptotically satisfy balance [24]. During the simulation, the density of states is continuously modified to produce a flat histogram in energy space, preventing the simulation from becoming stuck in low-energy minima. de Pablo and coworkers have used the density of states method, combined with hybrid Monte Carlo and pivot moves, to fold small protein molecules successfully [25].

3.7 Monte Carlo as an Optimization Tool

Monte Carlo methods that satisfy the balance condition are essential if one is interested in calculating properties at equilibrium. If one is, instead, only interested in optimizing a property of interest, for example, determining a minimum energy structure, then there is no such restriction, and one has greater flexibility in the choice of moves one can use. Simulated annealing, in which a simulation begins at a high temperature and is sequentially reduced, is one such method that does not satisfy balance but can be highly effective in overcoming energetic barriers.

In protein structure prediction, Monte Carlo methods can be particularly effective as an optimization tool. For example, Baker and coworkers have had great success in the high-resolution de novo structure prediction of small proteins [26]. In their method, they use a combination of multiscale force fields, Monte Carlo torsional angle moves, side-chain optimization using rotamer representations, and

gradient-based minimization of their all-atom energy function. Similar methods can be used in the refinement of structures from nuclear magnetic resonance (NMR) and x-ray data [27]. Saven and coworkers have also shown that biased Monte Carlo moves and parallel tempering-based simulations can be particularly useful in protein sequence design efforts, in which one attempts to define amino acid sequences that fold to a particular tertiary structure [28].

4 Statistical Errors

Although we may wish that the results from our computer simulations could provide exact values for a property that we are interested in, this is never the case, because we are unable to simulate our system for an infinite length of time. One must, therefore, estimate the error in the calculated value of the property. One way to do this is to use the method of block averaging. Suppose that we have n samples of a fluctuating property of interest, A, that were taken when our system was in equilibrium. The average value of A is estimated as:

$$\langle A \rangle \approx \bar{A} \equiv \frac{1}{n} \sum_{i=1}^{n} A_i . \tag{14}$$

If all of our samples were uncorrelated, then our estimate of the variance would be:

$$\sigma^2 (A) = \left\langle A^2 \right\rangle - \langle A \rangle^2 \approx \frac{1}{n} \sum_{i=1}^{n} \left[A_i - \bar{A} \right]^2 . \tag{15}$$

However, in a simulation, our samples are correlated. To account for this fact, one may compute averages for blocks of data, and use these block averages to estimate the variance. Where we have n' blocks of data, the variance of our new set of data is:

$$\sigma^2 \left(A' \right) = \left\langle A'^2 \right\rangle - \left\langle A' \right\rangle^2 = \frac{1}{n'} \sum_{i=1}^{n'} \left[A'_i - \bar{A} \right]^2 , \tag{16}$$

where A'_i is the block average of set i, and the average for the whole set of data remains the same. As the size of the block used increases, the correlation between the blocks will decrease. Thus, as the block size becomes sufficiently large, $\sigma^2 \left(A' \right) / \left(n' - 1 \right)$ should be constant and provide an estimate of the error of our ensemble average of A:

$$\varepsilon^2 (A) \approx \frac{\sigma^2 \left(A' \right)}{n' - 1} , \tag{17}$$

and an accurate measure of our statistical error ε, is now available.

Another way of calculating errors in a quantity is through the bootstrap method. From our original n samples, one produces N new sets of data, each containing n

samples. The samples that comprise each new set of data are picked at random from the original data set, and, therefore, it is probable that there will be repeat elements in the new data sets. These new data sets provide an estimate of the probability distribution of our property of interest, and, thus, provide another estimate of the variance and error in the simulation results.

5 Conclusions

Having presented a large number of Monte Carlo moves in Sect. 3, we conclude by discussing how one can combine these moves in an algorithm that satisfies balance. Most Monte Carlo algorithms are comprised of several different Monte Carlo moves that are effective at relaxing different degrees of freedom in a molecule or system. How to choose which Monte Carlo moves to include in a combined algorithm, and the probability of attempting each of them, is problem specific.

To satisfy detailed balance, the choice of what type of Monte Carlo move to attempt at a particular time is chosen on a probabilistic basis, and each of the specific Monte Carlo moves satisfies detailed balance individually. This scheme presents a problem when using methods such as parallel tempering, in which it is convenient to synchronize replicas in Monte Carlo move number and attempt swaps after a set number of Monte Carlo steps. This latter approach satisfies the balance condition, however, and therefore, it is perfectly acceptable. Also acceptable, because it satisfies balance, is sequential updating of a system, in which the moves are performed in a defined sequence, rather than the moves chosen at random.

Most Monte Carlo simulations to date are performed by software custom written by research groups. The most typical language for these Monte Carlo simulations is C. Both the Allen and Tildesley book [6] and the Frenkel and Smit book [5] offer web sites with useful routines [29, 30]. C language code for analytical rebridging [12] is available [31]. A web site by Mihaly Mezei lists several useful links related to Monte Carlo simulation of proteins [32].

References

1. Hu, J., Ma, A., and Dinner, A. R. (2006) Monte Carlo simulations of biomolecules: The MC module in CHARMM. *J. Comput. Chem.* **27**, 203–216.
2. Manousiouthakis, V. I. and Deem, M. W. (1999) Strict detailed balance is unnecessary in Monte Carlo simulation. *J. Chem. Phys.* **110**, 2753–2756.
3. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. N., and Teller, E. (1953) Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**, 1087–1092.
4. Metropolis, N. (1987) The beginning of the Monte Carlo method. *Los Alamos Science*, **12**, 125–130.
5. Frenkel, D. and Smit, B. (2002) *Understanding Molecular Simulation: From Algorithms to Applications.* Academic Press, San Diego, CA.

6. Allen, M. P. and Tildesley, D. J. (1987) *Computer Simulation of Liquids*. Clarendon Press, Oxford.
7. Siepmann, J. I. and Frenkel, D. (1992) Configurational-bias Monte Carlo: A new sampling scheme for flexible chains. *Mol. Phys.* **75**, 59–70.
8. Deem, M. W. and Bader, J. S. (1996) A configurational bias Monte Carlo method for linear and cyclic peptides. *Mol. Phys.* **87**, 1245–1260.
9. Dodd, L. R., Boone, T. D., and Theodorou, D. N. (1993) A concerted rotation algorithm for atomistic Monte Carlo simulation of polymer melts and glasses. *Mol. Phys.* **78**, 961–996.
10. Mavrantzas, V. G., Boone, T. D., Zevropoulou, E., and Theodorou, D. N. (1999) End-bridging Monte Carlo: A fast algorithm for atomistic simulation of condensed phases of long polymer chains. *Macromolecules* **32**, 5072–5096.
11. Wu, M. G. and Deem, M. W. (1999) Efficient Monte Carlo methods for cyclic peptides. *Mol. Phys.* **97**, 559–580.
12. Wu, M. G. and Deem, M. W. (1999) Analytical rebridging Monte Carlo: Application to cis/trans isomerization in proline-containing, cyclic peptides. *J. Chem. Phys.* **111**, 6625–6632.
13. Betancourt, M. R. (2005) Efficient Monte Carlo moves for polypeptide simulations. *J. Chem. Phys.* **123**, 174905.
14. Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. (1987) Hybrid Monte Carlo. *Phys. Rev. Lett.* **195**, 216–222.
15. Mehlig, B., Heermann, D. W., and Forrest, B. M. (1992) Hybrid Monte Carlo method for condensed matter systems. *Phys. Rev. B* **45**, 679–685.
16. Tuckerman, M. E., Berne, B. J., and Martyna, G. J. (1992) Reversible multiple time scale molecular dynamics. *J. Chem. Phys.* **97**, 1990–2001.
17. Geyer, C. J. and Thompson, E. A. (1995) Annealing Markov-Chain Monte Carlo with applications to ancestral inference, *J. Am. Stat. Assn.* **90**, 909–920.
18. Kone, A. and Kofke, D. A. (2005) Selection of temperature intervals for parallel tempering simulations, *J. Chem. Phys.* **122**, 206101.
19. Rathore, N., Chopra, M., and de Pablo, J. J. (2005) Optimal allocation of replicas in parallel tempering simulations, *J. Chem. Phys.* **122**, 024111.
20. Katzgraber, H. G., Trebst, S., Huse, D. A., and Troyer, M. (2006) Feedback-optimized parallel tempering Monte Carlo, *J. Stat. Mech.: Exp. & Theory* P03018.
21. Earl, D. J. and Deem, M. W. (2004) Optimal allocation of replicas to processors in parallel tempering simulations. *J. Phys. Chem. B* **108**, 6844–6849.
22. Earl, D. J. and Deem, M. W. (2005) Parallel tempering: theory, applications, and new perspectives, *Phys. Chem. Chem. Phys.* **7**, 3910–3916.
23. Wang, F. and Landau, D. P. (2001) Efficient, multiple-range random walk algorithm to calculate the density of states, *Phys. Rev. Lett.* **86**, 2050–2053.
24. Earl, D. J. and Deem, M. W. (2005) Markov chains of infinite order and asymptotic satisfaction of balance: Application to the adaptive integration method, *J. Phys. Chem. B* **109**, 6701–6704.
25. Rathore, N., Knotts IV, T. A., and de Pablo, J. J. (2003) Configurational temperature density of states simulations of proteins, *Biophys. J.* **85**, 3963–3968.
26. Bradley, P., Misura, K. M. S., and Baker, D. (2005) Toward high-resolution de novo structure prediction for small proteins, *Science* **309**, 1868–1871.
27. Meiler, J. and Baker, D. (2003) Rapid protein fold determination using unassigned NMR data, *Proc. Natl. Acad. Sci. USA* **100**, 15404–15409.
28. Yang, X. and Saven, J. G. (2005) Computational methods for protein design sequence variability: biased Monte Carlo and replica exchange, *Chem. Phys. Lett.* **401**, 205–210.
29. http://www.ccl.net/cca/software/SOURCES/FORTRAN/allen-tildesley-book/index.shtml
30. http://molsim.chem.uva.nl/frenkel_smit/index.html
31. http://www.mwdeem.rice.edu/rebridge/
32. http://fulcrum.physbio.mssm.edu/∼mezei/