3
Dynamics Methods

Oren M. Becker
Tel Aviv University, Tel Aviv, Israel

Masakatsu Watanabe*
Moldyn, Inc., Cambridge, Massachusetts

I. INTRODUCTION

Molecular dynamics simulation, which provides the methodology for detailed microscopical modeling on the atomic scale, is a powerful and widely used tool in chemistry, physics, and materials science. This technique is a scheme for the study of the natural time evolution of the system that allows prediction of the static and dynamic properties of substances directly from the underlying interactions between the molecules.

Dynamical simulations monitor time-dependent processes in molecular systems in order to study their structural, dynamic, and thermodynamic properties by numerically solving an equation of motion, which is the formulation of the rules that govern the motion executed by the molecule. That is, molecular dynamics (MD) provides information about the time dependence and magnitude of fluctuations in both positions and velocities, whereas the Monte Carlo approach provides mainly positional information and gives only little information on time dependence.

Depending on the desired level of accuracy, the equation of motion to be numerically solved may be the classical equation of motion (Newton's), a stochastic equation of motion (Langevin's), a Brownian equation of motion, or even a combination of quantum and classical mechanics (QM/MM, see Chapter 11).

Good reviews of the application of dynamic simulation methods to biomolecules can be found in the books by Brooks et al. [1] and McCammon and Harvey [2]. Good short reviews on this topic can also be found in Refs. 3–5. More detailed discussions of dynamic simulation methodologies can be found in books by Allen and Tildesley [6], Frenkel and Smit [7], and Rapaport [8] and in the review by van Gunsteren [9].

II. TYPES OF MOTIONS

Macromolecules in general, and proteins in particular, display a broad range of characteristic motions. These range from the fast and localized motions characteristic of atomic fluctuations to the slow large-scale motions involved in the folding transition. Many of these motions, on each and every time scale, have an important role in the biological function of proteins. For example, localized side-chain motion controls the diffusion of oxygen into and out of myoglobin and hemoglobin [1]. A more extensive "medium-scale" structural transition is involved, for example, in the hemoglobin R to T allosteric transition, which makes it such an efficient transport agent [1]. Finally, prion proteins exhibit a global structural transition of biological importance. These proteins undergo a global transition from an α-helical structure to a predominantly β-sheet structure, which is implicated in the onset of Creutzfeldt-Jacob disease (CJD) in humans and the "mad cow" disease in cattle (bovine spongiform encephalopathy; BSE) [10].

Table 1 gives a crude overview of the different types of motion executed by a protein and their characteristic time scales and amplitudes. These should be regarded as rough guidelines, because individual motions in specific systems may differ significantly from these estimates. Note that the motions executed by a protein span almost 20 orders of magnitude in characteristic time scales, from femtoseconds (10^{-15} s) to hours (10^4–10^5 s). They also cover a wide range of amplitudes (0.01–100 Å) and energies (0.1–100 kcal/mol).

An important characteristic of biomolecular motion is that the different types of motion are interdependent and coupled to one another. For example, a large-scale dynamic transition cannot occur without involving several medium-scale motions, such as helix rearrangements. Medium-scale motions cannot occur without involving small-scale motions, such as side-chain movement. Finally, even side-chain motions cannot occur without the presence of the very fast atomic fluctuations, which can be viewed as the "lubricant" that enables the whole molecular construction to move. From the point of view of dynamic

Table 1 An Overview of Characteristic Motions in Proteins

Type of motion	Functionality examples	Time and amplitude scales
Local motions	Ligand docking flexibility	Femtoseconds (fs) to picoseconds (ps) (10^{-15}–10^{-12} s); less than 1 Å
Atomic fluctuation	Temporal diffusion pathways	
Side chain motion		
Medium-scale motions		
Loop motion	Active site conformation adaptation	Nanoseconds (ns) to microseconds (μs) (10^{-9}–10^{-6} s); 1–5 Å
Terminal-arm motion		
Rigid-body motion	Binding specificity	
Large-scale motions		
Domain motion	Hinge-bending motion	Microseconds (μs) to milliseconds (ms) (10^{-6}–10^{-3} s); 5–10 Å
Subunit motion	Allosteric transitions	
Global motions		
Helix-coil transition	Hormone activation	Milleseconds (ms) to hours (10^{-3}–10^4 s); more than 10 Å
Folding/unfolding	Protein functionality	
Subunit association		

simulations, this has serious implications. It indicates that even in the study of slow large-scale motions (of biological importance) it is not possible to ignore the fast small-scale motions, which eventually are the ones that impose limitations on the simulation time step and length.

III. THE STATISTICAL MECHANICS BASIS OF MOLECULAR DYNAMICS

A classical system is described by a classical Hamiltonian, H, which is a function of both coordinates $\mathbf{r}$ and momenta $\mathbf{p}$. For regular molecular systems, where the potential energy function is independent of time and velocity, the Hamiltonian is equal to the total energy,

$$H = H(\mathbf{r}, \mathbf{p}) = K(\mathbf{p}) + U(\mathbf{r}) = \sum_i \frac{p_i}{2m_i} + U(\mathbf{r}) \tag{1}$$

where $K(\mathbf{p})$ is the kinetic energy, $U(\mathbf{r})$ is the potential energy, p_i is the momentum of particle i, and m_i the mass of particle i. A microscopic state of the system is therefore characterized by the set of values $\{\mathbf{r}, \mathbf{p}\}$, which corresponds to a point in the space defined by both coordinates $\mathbf{r}$ and momenta $\mathbf{p}$ (known as ''phase space'').

To obtain thermodynamic averages over a ''canonical'' ensemble, which is characterized by the macroscopic variables (N, V, T), it is necessary to know the probability of finding the system at each and every point (= state) in phase space. This probability distribution, $\rho(\mathbf{r}, \mathbf{p})$, is given by the Boltzmann distribution function,

$$\rho(\mathbf{r}, \mathbf{p}) = \frac{\exp\,[-H(\mathbf{r}, \mathbf{p})/k_BT}{Z} \tag{2}$$

where the canonical partition function, Z, is an integral over all phase space of the Boltzmann factors $\exp\,[-H(\mathbf{r}, \mathbf{p})/k_BT]$, and k_B is the Boltzmann factor. Once this distribution function is known it can be used to calculate phase space averages of any dynamic variable $A(\mathbf{r}, \mathbf{p})$ of interest. Examples for dynamic variables are the position, the total energy, the kinetic energy, fluctuations, and any other function of $\mathbf{r}$ and/or $\mathbf{p}$. These averages,

$$\langle A(\mathbf{r}, \mathbf{p})\rangle_Z = \int_V d\mathbf{r} \int_{-\infty}^{\infty} d\mathbf{p}\; \rho(\mathbf{r}, \mathbf{p})A(\mathbf{r}, \mathbf{p}) \tag{3}$$

are called ''thermodynamic averages'' or ''ensemble averages'' because they take into account every possible state of the system. However, in order to calculate these thermodynamic averages, it is necessary to simultaneously know the Boltzmann probability [Eq. (2)] for each and every state $\{\mathbf{r}, \mathbf{p}\}$, which is an extremely difficult computational task.

An alternative strategy for calculating systemwide averages is to follow the motion of a single point through phase space instead of averaging over the whole phase space all at once. That is, in this approach the motion of a single point (a single molecular state) through phase space is followed as a function of time, and the averages are calculated only over those points that were visited during the excursion. Averages calculated in this way are called ''dynamic averages.'' The motion of a single point through phase space is obtained by integrating the system's equation of motion. Starting from a point $\{\mathbf{r}(0), \mathbf{p}(0)\}$, the integration procedure yields a trajectory that is the set of points $\{\mathbf{r}(t), \mathbf{p}(t)\}$

describing the state of the system at any successive time t. Dynamic averages of any dynamical variable $A(\mathbf{r}, \mathbf{p})$ can now be calculated along this trajectory as follows:

$$\langle A(\mathbf{r}, \mathbf{p})\rangle_\tau = \frac{1}{\tau}\int_0^\tau A(\mathbf{r}(t), \mathbf{p}(t))dt \tag{4}$$

where τ is the duration of the simulation. Compared to the previous approach, dynamic averaging is easier to perform. The two averaging strategies can be summarized as follows:

Thermodynamic average. An average over *all* points in phase space at a *single* time.
Dynamic average. An average over a *single* point in phase space at *all* times.

It is hoped that the point that is being dynamically followed will eventually cover all of phase space and that the dynamic average will converge to the desired thermodynamic average. A key concept that ties the two averaging strategies together is the *ergodic hypothesis.* This hypothesis states that for an infinitely long trajectory the thermodynamic ensemble average and the dynamic average become equivalent to each other,

$$\lim_{\tau\to\infty} \langle A(\mathbf{r}, \mathbf{p})\rangle_\tau = \langle A(\mathbf{r}, \mathbf{p})\rangle_Z \tag{5}$$

In other words, the ergodic hypothesis claims that when the trajectory becomes long enough, the point that generates it will eventually cover all of phase space, so the two averages become identical. For this hypothesis to hold, the system has to be at equilibrium (technically, at a stationary state). Also, there must not be any obstacle, such as a fragmented topology, that will prevent an infinitely long trajectory from covering all of phase space. A system that obeys these two conditions is said to be *ergodic*, and its hypothesis is the theoretical justification for using molecular dynamic simulations as a means for calculating thermodynamic averages of molecular systems. It is tacitly assumed that finite molecular dynamics trajectories are "long enough" in the ergodic sense.

IV. NEWTONIAN MOLECULAR DYNAMICS

A. Newton's Equation of Motion

The temporal behavior of molecules, which are quantum mechanical entities, is best described by the quantum mechanical equation of motion, i.e., the time-dependent Schrödinger equation. However, because this equation is extremely difficult to solve for large systems, a simpler classical mechanical description is often used to approximate the motion executed by the molecule's heavy atoms. Thus, in most computational studies of biomolecules, it is the classical mechanics Newtonian equation of motion that is being solved rather than the quantum mechanical equation.

In its most simplistic form, Newton's equation of motion (also known as Newton's second law of motion) states that

$$F_i = m_i a_i = m_i \ddot{r}_i \tag{6}$$

where F_i is the force acting on particle i, m_i is the mass of particle i, a_i is its acceleration, and $\ddot{r}_i$ is the second derivative of the particle position $\mathbf{r}$ with respect to time. The force F_i is determined by the gradient of the potential energy function, $U(\mathbf{r})$, discussed in Chapter 2, which is a function of all the atomic coordinates $\mathbf{r}$,

$$F_i = -\nabla_i U(\mathbf{r}) \tag{7}$$

Equation (7) is a second-order differential equation. A more general formulation of Newton's equation of motion is given in terms of the system's Hamiltonian, H [Eq. (1)]. Put in these terms, the classical equation of motion is written as a pair of coupled first-order differential equations:

$$\dot{r}_k = \frac{\partial H(\mathbf{r}, \mathbf{p})}{\partial p_k}; \qquad \dot{p}_k = -\frac{\partial H(\mathbf{r}, \mathbf{p})}{\partial r_k} \tag{8}$$

By substituting the definition of H [Eq. (1)] into Eq. (8), we regain Eq. (6). The first first-order differential equation in Eq. (8) becomes the standard definition of momentum, i.e., $p_i = m_i \dot{r}_i = m_i v_i$, while the second turns into Eq. (6). A set of two first-order differential equations is often easier to solve than a single second-order differential equation.

B. Properties of Newton's Equation of Motion

Newton's equation of motion has several characteristic properties, which will later serve as "handles" to ensure that the numerical solution is correct (Section V.C). These properties are

Conservation of energy. Assuming that U and H do not depend explicitly on time or velocity (so that $\partial H/\partial t = 0$), it is easy to show from Eq. (8) that the total derivative dH/dt is zero; i.e., the Hamiltonian is a constant of motion for Newton's equation. In other words, there is conservation of total energy under Newton's equation of motion.

Conservation of linear and angular momentum. If the potential function U depends only on particle separation (as is usual) and there is no external field applied, then Newton's equation of motion conserves the total linear momentum of the system, $\boldsymbol{P}$,

$$\boldsymbol{P} = \sum_i p_i \tag{9}$$

and the total angular momentum, $\boldsymbol{L}$,

$$\boldsymbol{L} = \sum_i r_i \times p_i = \sum_i m_i r_i \times \dot{r}_i \tag{10}$$

Time reversibility. The third property of Newton's equation of motion is that it is reversible in time. Changing the signs of all velocities (or momenta) will cause the molecule to retrace its trajectory. If the equations of motion are solved correctly, then the numerical trajectory should also have this property. Note, however, that in practice this time reversibility can be reproduced by numerical trajectories only over very short periods of time because of the chaotic nature of large molecular systems.

C. Molecular Dynamics: Computational Algorithms

Solving Newton's equation of motion requires a numerical procedure for integrating the differential equation. A standard method for solving ordinary differential equations, such as Newton's equation of motion, is the finite-difference approach. In this approach, the molecular coordinates and velocities at a time $t + \Delta t$ are obtained (to a sufficient degree of accuracy) from the molecular coordinates and velocities at an earlier time t. The equations are solved on a step-by-step basis. The choice of time interval Δt depends on the properties of the molecular system simulated, and Δt must be significantly smaller than the characteristic time of the motion studied (Section V.B).

A good starting point for understanding finite-difference methods is the Taylor expansion about time t of the position at time $t + \Delta t$,

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \dot{\mathbf{r}}(t)\Delta t + \frac{1}{2}\ddot{\mathbf{r}}(t)\Delta t^2 + \cdots \tag{11}$$

Alternatively, this can be written as

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 + \cdots \tag{12}$$

where $\mathbf{v}(t)$ is the velocity vector and $\mathbf{a}(t)$ is the acceleration. Because the integration proceeds in a stepwise fashion, and recalling Eq. (6), it is convenient to rewrite the above expansion in a discrete form. Using $\mathbf{r}_n$ to indicate the position at step n (at time t) and $\mathbf{r}_{n+1}$ to indicate the position at the next step, $n + 1$ (at time $t + \Delta t$), Eq. (12) can be written as

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_n\Delta t + \frac{1}{2}\left(\frac{\mathbf{F}_n}{\mathbf{m}}\right)\Delta t^2 + O(\Delta t^3) \tag{13}$$

where $O(\Delta t^n)$ is the terms of order Δt^n or smaller. With this information the velocity $\mathbf{v}_{n+1}$ at time $n + 1$ can be crudely estimated, for example, as

$$\mathbf{v}_{n+1} = (\mathbf{r}_{n+1} - \mathbf{r}_n)/2 \tag{14}$$

Together, Eqs. (13) and (14) form an integration algorithm. Given the position $\mathbf{r}_n$, the velocity $\mathbf{v}_n$, and the force $\mathbf{F}_n$ at step n, these equations allow one to calculate (actually, estimate) the position $\mathbf{r}_{n+1}$ and velocity $\mathbf{v}_{n+1}$ at step $n + 1$. The formulation is highly trivial and results in a low quality integration algorithm (large errors). Other, more accurate, algorithms have been developed using the same kind of reasoning. In the following subsections we survey some of the more commonly used finite-difference integration algorithms, highlighting their advantages and disadvantages.

1. Verlet Integrator

The most common integration algorithm used in the study of biomolecules is due to Verlet [11]. The Verlet integrator is based on two Taylor expansions, a forward expansion ($t + \Delta t$) and a backward expansion ($t - \Delta t$),

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_n\Delta t + \frac{1}{2}\left(\frac{\mathbf{F}_n}{\mathbf{m}}\right)\Delta t^2 + O(\Delta t^3) \tag{15a}$$

$$\mathbf{r}_{n-1} = \mathbf{r}_n + \mathbf{v}_n\Delta t + \frac{1}{2}\left(\frac{\mathbf{F}_n}{\mathbf{m}}\right)\Delta t^2 - O(\Delta t^3) \tag{15b}$$

The sum of the two expansions yields an algorithm for propagating the position,

$$\mathbf{r}_{n+1} = 2\mathbf{r}_n - \mathbf{r}_{n-1} + \frac{\mathbf{F}_n}{\mathbf{m}}\Delta t^2 + O(\Delta t^4) \tag{16}$$

Translated into a stream of commands, this algorithm is executed in two steps:

1. Use the current position $\mathbf{r}_n$ to calculate the current force $\mathbf{F}_n$.
2. Use the current and previous positions $\mathbf{r}_n$ and $\mathbf{r}_{n-1}$ together with the current force $\mathbf{F}_n$ (calculated in step 1) to calculate the position in the next step, $\mathbf{r}_{n+1}$, according to Eq. (16).

These two steps are repeated for every time step for each atom in the molecule. Subtracting Eq. (15b) from Eq (15a) yields a complementary algorithm for propagating the velocities,

$$\mathbf{v}_n = \frac{\mathbf{r}_{n+1} - \mathbf{r}_{n+1}}{2\Delta t} + O(\Delta t^2) \tag{17}$$

Figure 1a gives a graphical representation of the steps involved in a Verlet propagation. The algorithm embodied in Eqs. (16) and (17) provides a stable numerical method for solving Newton's equation of motion for systems ranging from simple fluids to biopolymers. Like any algorithm, the Verlet algorithm has advantages as well as disadvantages.

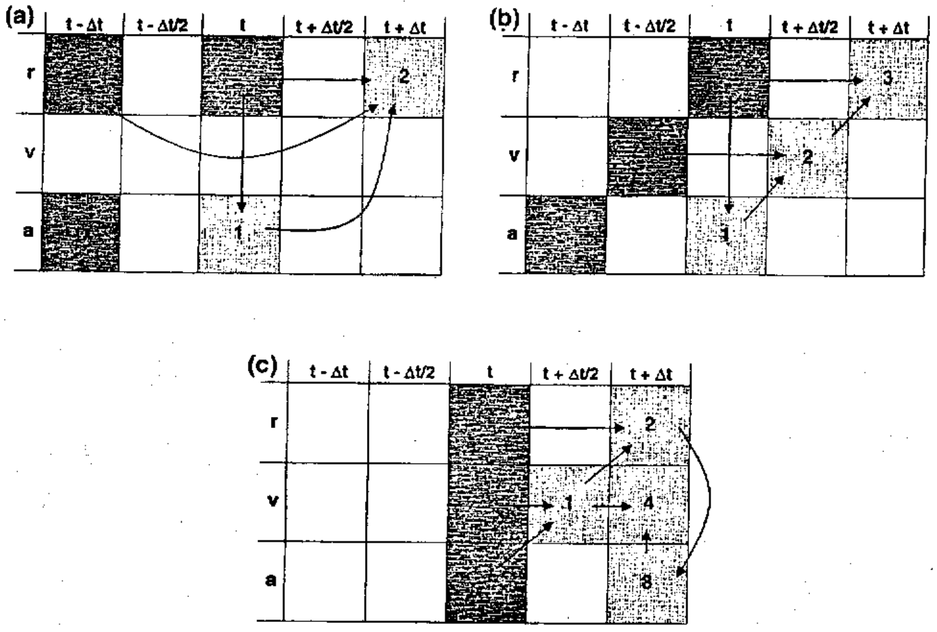

Figure 1 A stepwise view of the Verlet integration algorithm and its variants. (a) The basic Verlet method. (b) Leap-frog integration. (c) Velocity Verlet integration. At each algorithm dark and light gray cells indicate the initial and calculating variables, respectively. The numbers in the cells represent the orders in the calculation procedures. The arrows point from the data that are used in the calculation of the variable that is being calculated at each step.

Advantages of the Verlet algorithm are

1. The position integration is quite accurate [errors on the order of $O(\Delta t^4)$] and is independent of the velocity propagation, a fact that simplifies the position integration and reduces memory requirements.
2. The algorithm requires only a single force evaluation per integration cycle (computationally, force evaluations are the most ''expensive'' part of the simulation).
3. This formulation, which is based on forward and backward expansions, guarantees time reversibility (a property of the equation of motion).

Disadvantages of the Verlet algorithm are

1. The velocity propagation is subject to relatively large errors, on the order $O(\Delta t^2)$. Recall that an accurate estimate of the velocity is required for the kinetic energy evaluations. An added inconvenience is that $\mathbf{v}_n$ can be computed only if $\mathbf{r}_{n+1}$ is already known.
2. Further numerical errors are introduced when an $O(\Delta t^2)$ term is added to an $O(\Delta t^0)$ term.
3. The Verlet algorithm is not ''self-starting.'' A lower order Taylor expansion [e.g., Eq. (13)] is often used to initiate the propagation.
4. It must be modified to incorporate velocity-dependent forces or temperature scaling.

2. Leap-Frog Integrator

Modifications to the basic Verlet scheme have been proposed to tackle the above deficiencies, particularly to improve the velocity evaluation. One of these modifications is the leap-frog algorithm, so called for its half-step scheme: Velocities are evaluated at the midpoint of the position evaluation and vice versa [12,13]. The algorithm can be written as

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_{n+1/2}\Delta t \tag{18a}$$

$$\mathbf{v}_{n+1/2} = \mathbf{v}_{n-1/2} + \frac{\mathbf{F}_n}{\mathbf{m}}\Delta t \tag{18b}$$

where $\mathbf{v}_{n\pm 1/2}$ stands for the velocity at the mid-step time $[t \pm (1/2)\Delta t]$. Elimination of the velocities from these equations shows that the method is algebraically equivalent to the Verlet algorithm. Cast in the form of execution instructions, the leap-frog algorithm involves three steps:

1. Use the current position $\mathbf{r}_n$ to calculate the current force $\mathbf{F}_n$.
2. Use the current force $\mathbf{F}_n$ and previous mid-step velocity $\mathbf{v}_{n-1/2}$ to calculate the next mid-step velocity $\mathbf{v}_{n+1/2}$.
3. Use the current position $\mathbf{r}_n$ and the next mid-step velocity $\mathbf{v}_{n+1/2}$ (from step 2) to calculate the position in the next step, $\mathbf{r}_{n+1}$.

Figure 1b gives a graphical representation of the steps involved in the leap-frog propagation. The current velocity $\mathbf{v}_n$, which is necessary for calculating the kinetic energy, can be calculated as

$$\mathbf{v}_n = (\mathbf{v}_{n+1/2} + \mathbf{v}_{n-1/2})/2 \tag{19}$$

Advantages of the leap-frog algorithm are

1. It improves velocity evaluation.
2. The direct evaluation of velocities gives a useful handle for controlling the simulation temperature (via velocity scaling).
3. It reduces the numerical errors, since here $O(\Delta t^1)$ terms are added to $O(\Delta t^0)$ terms.

Disadvantages of the leap-frog algorithm are

1. It still does not handle the velocities in a completely satisfactory manner, because the velocities at time t are only approximated by Eq. (19).
2. This algorithm is computationally a little more expensive than the Verlet algorithm.

3. Velocity Verlet Integrator

An even better handling of the velocities is obtained by another variant of the basic Verlet integrator, known as the "velocity Verlet" algorithm. This is a Verlet-type algorithm that stores positions, velocities, and accelerations all at the same time t and minimizes round-off errors [14]. The velocity Verlet algorithm is written

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_n \Delta t + \frac{1}{2}\left(\frac{\mathbf{F}_n}{\mathbf{m}}\right)\Delta t^2 \tag{20a}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{1}{2}\left[\frac{\mathbf{F}_n}{\mathbf{m}} + \frac{\mathbf{F}_{n+1}}{\mathbf{m}}\right]\Delta t \tag{20b}$$

Again, elimination of the velocities from these equations recovers the Verlet algorithm. In practice, the velocity Verlet algorithm consists of the following steps:

1. Calculate the position $\mathbf{r}_{n+1}$ at time $t + \Delta t$ from Eq. (20a).
2. Calculate the velocity at mid-step $\mathbf{v}_{n+1/2}$ using the equation

$$\mathbf{v}_{n+1/2} = \mathbf{v}_n + \frac{1}{2}\left(\frac{\mathbf{F}_n}{\mathbf{m}}\right)\Delta t \tag{21}$$

3. Calculate the force $\mathbf{F}_{n+1}$ at time $t + \Delta t$.
4. Finally, complete the velocity move to $\mathbf{v}_n$ by using

$$\mathbf{v}_{n+1} = \mathbf{v}_{n+1/2} + \frac{1}{2}\left(\frac{\mathbf{F}_{n+1}}{\mathbf{m}}\right)\Delta t \tag{22}$$

At this point, the kinetic energy at time $t + \Delta t$ is available. Figure 1c gives a graphical representation of the steps involved in the velocity Verlet propagation.

Advantages of the velocity Verlet algorithm are

1. It is numerically very stable.
2. It is convenient and simple [the code of this method is a straightforward transcription of Eqs. (20)–(22)].
3. It provides an accurate evaluation of velocities and hence of the kinetic energy.

The main disadvantage of this algorithm is that it is computationally a little more expensive than the simpler Verlet or leap-frog algorithms (though the added accuracy often outweighs this slight overhead).

V. MOLECULAR DYNAMICS: SIMULATION PRACTICE

A. Assigning Initial Values

Newton's equation of motion is a second-order differential equation that requires two initial values for each degree of freedom in order to initiate the integration. These two initial values are typically a set of initial coordinates $\{\mathbf{r}(0)\}$ and a set of initial velocities $\{\mathbf{v}(0)\}$.

1. Initial Coordinates

The initial coordinates $\{\mathbf{r}(0)\}$ are usually obtained from experimentally determined molecular structures, mainly from X-ray crystallography and NMR experiments. Alternatively, the initial coordinates can be based on computer models generated by a variety of modeling techniques (see Chapters 14 and 15). Note, however, that even the experimentally determined structures must often undergo some preparation steps before they can be used as initial structures in a dynamic simulation.

First, it is not possible to determine hydrogen atom positions by X-ray crystallography. Thus the coordinates for the many hydrogen atoms in the molecule are missing from X-ray coordinate files. These coordinates must be added to the initial structure before the simulation is started. Several algorithms are available for ensuring reasonable placement of hydrogens.

In some cases, whole parts of the protein are missing from the experimentally determined structure. At times, these omissions reflect flexible parts of the molecule that do not have a well-defined structure (such as loops). At other times, they reflect parts of the molecule (e.g., terminal sequences) that were intentionally removed to facilitate the crystallization process. In both cases, structural models may be used to fill in the gaps.

After all the coordinates are accounted for, it is good practice to refine the initial structure by submitting it to energy minimization (see Chapter 4). The role of this minimization is to relieve local stresses due to nonbonded overlaps, as well as to relax bond length and bond angle distortions in the experimental structure. The origin of these stresses is due both to the empirical nature of the energy function (Chapter 2) and to the average nature of the experimentally determined structures.

2. Initial Velocities

Unlike the initial coordinates, which can be obtained experimentally, the only relevant information available about atomic velocities is the system's temperature T, which determines the velocity distribution. In the absence of a better guideline, initial velocities (v_x, v_y, v_z) are usually randomly assigned from the standard Maxwellian velocity distribution at a temperature T,

$$P(v)dv = \left(\frac{m}{2\pi k_B T}\right)^{1/2} \exp\left[\frac{-mv^2}{2k_B T}\right] dv \tag{23}$$

This initial assignment is, of course, not at equilibrium. In particular, the expected velocity correlation between neighboring atoms is not guaranteed, and most likely it is nonexistent (i.e., in general, neighboring atoms, such as bonded pairs, are expected to move at similar velocities). Furthermore, the random assignment process may accidentally assign high velocities to a localized cluster of atoms, creating a "hot spot" that makes the simulation unstable. To overcome this problem, it is common practice to start a simulation with a "heat-up" phase. Velocities are initially assigned at a low temperature, which is then increased gradually allowing for dynamic relaxation. This slow heating continues until the simulation reaches the desired temperature.

In practice, heating is performed by increasing atomic velocities, either by reassigning new velocities from a Maxwellian distribution [Eq. (23)] at an elevated temperature or by scaling the velocities by a uniform factor. This heating process, as well as the dynamic simulation that follows, requires a measurable definition of the system's temperature T at time t. According to the equipartition theorem, the temperature, $T(t)$, at any given time t is defined in terms of the mean kinetic energy by

$$T(t) = \frac{1}{k_B N_{\text{dof}}} \sum_{i=1}^{N_{\text{dof}}} m_i |v_i|^2 \tag{24}$$

where N_{dof} is the number of unconstrained degrees of freedom in the system ($N_{\text{dof}} = 3N - n$, where N is the number of atoms and n is the number of constraints). It is clear from this expression that scaling the velocities by a factor of $[T_0/T(t)]^{1/2}$ will result in a mean kinetic energy corresponding to a desired temperature T_0.

Another problem related to the initial velocity assignment is the large total linear momentum $\boldsymbol{P}$ and total angular momentum $\boldsymbol{L}$ formed [Eqs. (9) and (10)]. These momenta cause a drift of the molecule's center of mass relative to the reference frame. Numerically, this drift hampers the computational efficiency of the simulation, overshadowing the smaller internal motions by the physically irrelevant translational drift and global rotation. Because Newton's equation of motion conserves linear and angular momenta, these momenta will not go away unless they are actively taken out. The ideal zero-drift situation is reached by periodically zeroing these momenta during the equilibration phase of the simulation.

B. Selecting the Integration Time Step

The size of the time step Δt is an important parameter that determines the magnitude of the error associated with each of the foregoing integration algorithms. On the one hand, a small time step means better integration quality. But on the other hand it also means that more integration steps are required for the same length of simulation. Thus, every simulation involves a trade-off between economy and accuracy. The time step in molecular dynamics simulations is one of the most important factors that balance this trade-off. In general, one would like to choose the largest possible time step that will still ensure an accurate simulation.

An appropriate time step should be small by comparison to the period of the fastest motion (highest frequency motion) in the system being simulated. If τ is the period of the fastest motion, a good rule of thumb for selecting Δt is

$$\tau/\Delta t \approx 20 \tag{25}$$

For biomolecules, such as proteins, the fastest motions are the stretching vibrations of the bonds connecting hydrogen atoms to heavy atoms (X—H stretching). The frequency of these motions is in the vicinity of 3000 cm^{-1}, which means periods of about 10 fs (1×10^{-14} s). Thus, an appropriate time step for simulating biomolecules would be $\Delta t \approx$

0.5 fs. This extremely small time step is the main reason that, with only a few exceptions, molecular dynamics simulations are limited today to the nanosecond (10^{-9} s) time scale.

Naturally, much effort is invested in developing advanced algorithms that allow for larger time steps that enable longer simulations. A basic rationale, common to many such approaches, is to remove the fast (high frequency) motions from the numerical integration and account for them in some other way. Because of the characteristic coupling between types of motion (Section III.A), this task is far from simple. A first step in this direction is to take the X—H stretching motions out of the numerical integration. If these stretching motions were accounted for in some other way, then the time step would be determined by the next fastest molecular motion, i.e., the X—X stretching modes with frequencies around 1500 cm^{-1}. According to relation (25), this elimination will increase the time step by a factor of 2 (to $\Delta t \approx 1.0$ fs), extending the length of the simulation by a similar factor at only a slight additional computational cost.

The algorithm that is usually employed to account for the hydrogen positions is SHAKE [15,16] (and its variant RATTLE [17]). Stated in a simplistic way, the SHAKE algorithm assumes that the length of the X—H bond can be considered constant. Because in a numerical simulation there are always fluctuations, this means that the deviation of the current length $d_k(t)$ of the kth bond from its ideal (constant) bond length d_k^0 must be smaller than some tolerance value ε,

$$s_k = [d_k(t)^2 - d_k^{o2}]/d_k^{o2} < \varepsilon \tag{26}$$

SHAKE is an iterative procedure that adjusts the atomic positions (of the hydrogen atoms in this case) after each integration step (of the heavy atoms) in order to simultaneously satisfy all the constraints. It iterates until s_k is smaller than ε for all values of k, (A more detailed description of the algorithm can be found in Refs. 14 and 16 and in appendix A1.4 of Ref. 2.) SHAKE can be applied not only to X—H type bonds but also to all bond-stretching motions in the system, allowing for time steps as large as 2 or 3 fs (depending on the details of the system). Although, SHAKE can in principle be applied to bending motions too, it was found that such constraints result in low quality simulations. This is due to the fact that the important coupling between bond angle motion and torsional motion is neglected [18].

More advanced methods for extending the length of molecular dynamics simulations are discussed in Section VIII.

C. Stability of Integration

An important issue in any numerical study is that of the accuracy and stability of the simulation. A simulation become unstable when it has picked up errors along the way and has become essentially meaningless. In general, it is unrealistic to expect that any approximate method of solution will follow the exact classical trajectory indefinitely. It is our goal, however, to maintain a stable simulation for at least the duration of interest for the specific study. Thus, the stability of the simulation must be gauged at all times. If one is lucky, an unstable simulation will also crash and not reach its designated termination. It may happen, however, that even though the unstable simulation reaches its designated termination, its content carries little merit.

The best gauges for the stability of any simulation are the constants of motion of the physical equation that is numerically solved, i.e., quantities that are expected to be conserved during the simulation. Since numerical fluctuations cannot be avoided, a dynamic variable $A(\mathbf{r},\mathbf{p})$ is considered numerically "conserved" if the ratio of its fluctuations to its value is below an acceptable tolerance ε, $\Delta A/A < \varepsilon$. The constants of motion for Newton's equation of motion were specified in Section IV.B.

Conservation of energy. Newton's equation of motion conserves the total energy of the system, E (the Hamiltonian), which is the sum of potential and kinetic energies [Eq. (1)]. A fluctuation ratio that is considered adequate for a numerical solution of Newton's equation of motion is

$$\frac{\Delta E}{E} < 10^{-4} \qquad \text{or} \qquad \log_{10}\left(\frac{\Delta E}{E}\right) < -4 \tag{29}$$

Conservation of linear and angular momenta. After equilibrium is reached, the total linear momentum P [Eq. (9)] and total angular momentum L [Eq. (10)] also become constants of motion for Newton's equation and should be conserved. In advanced simulation schemes, where velocities are constantly manipulated, momentum conservation can no longer be used for gauging the stability of the simulation.

Time reversibility. Newton's equation is reversible in time. For a numerical simulation to retain this property it should be able to retrace its path back to the initial configuration (when the sign of the time step Δt is changed to $-\Delta t$). However, because of chaos (which is part of most complex systems), even modest numerical errors make this backtracking possible only for short periods of time. Any two classical trajectories that are initially very close will eventually exponentially diverge from one another. In the same way, any small perturbation, even the tiny error associated with finite precision on the computer, will cause the computer trajectories to diverge from each other and from the exact classical trajectory (for examples, see pp. 76–77 in Ref. 6). Nonetheless, for short periods of time a stable integration should exhibit temporal reversibility.

D. Simulation Protocol and Some Tricks for Better Simulations

Every molecular dynamics simulation consists of several steps. Put together, these steps are called a "simulation protocol." The main steps common to most dynamic simulation protocols are the following.

1. *Preparation of the data.* Preparation of the initial coordinates (adding hydrogen atoms, minimization) and assignment of initial velocities.
2. *Heating up.* Gradual scaling of the velocities to the desired temperature, accompanied by short equilibration dynamics.
3. *Equilibration.* A relatively long dynamic simulation, whose goal is to ensure that the simulation is stable and free of erratic fluctuations. This step may take from tens of picoseconds to several hundred picoseconds.
4. *Production.* When the simulation is "equilibrated," the dynamic simulation is considered reliable. From this point on, the trajectory generated is stored for further analysis. Typical "production runs" take from several hundred picoseconds up to tens of nanoseconds (depending on the size of the system and the available computer power).

5. *Analysis.* The resulting trajectory is submitted to careful analysis. (Refer to Section VI.)

We have presented a simple protocol to run MD simulations for systems of interest. There are, however, some ''tricks'' to improve the efficiency and accuracy of molecular dynamics simulations. Some of these techniques, which are discussed later in the book, are today considered standard practice. These methods address diverse issues ranging from efficient force field evaluation to simplified solvent representations.

One widely used trick is to apply bookkeeping to atom–atom interactions, commonly referred to as the neighbor list [11], which is illustrated in Figure 2. If we simulate a large N-particle system and use a cutoff that is smaller than the simulation box, then many particles do not contribute significantly to the energy of a particle i. It is advantageous, therefore, to exclude from the expensive energy calculation particle pairs that do not interact. This technique increases the efficiency of the simulations. Details of programming for this approach can be found in the books by Allen and Tildesley [6] and Frenkel and Smit [7].

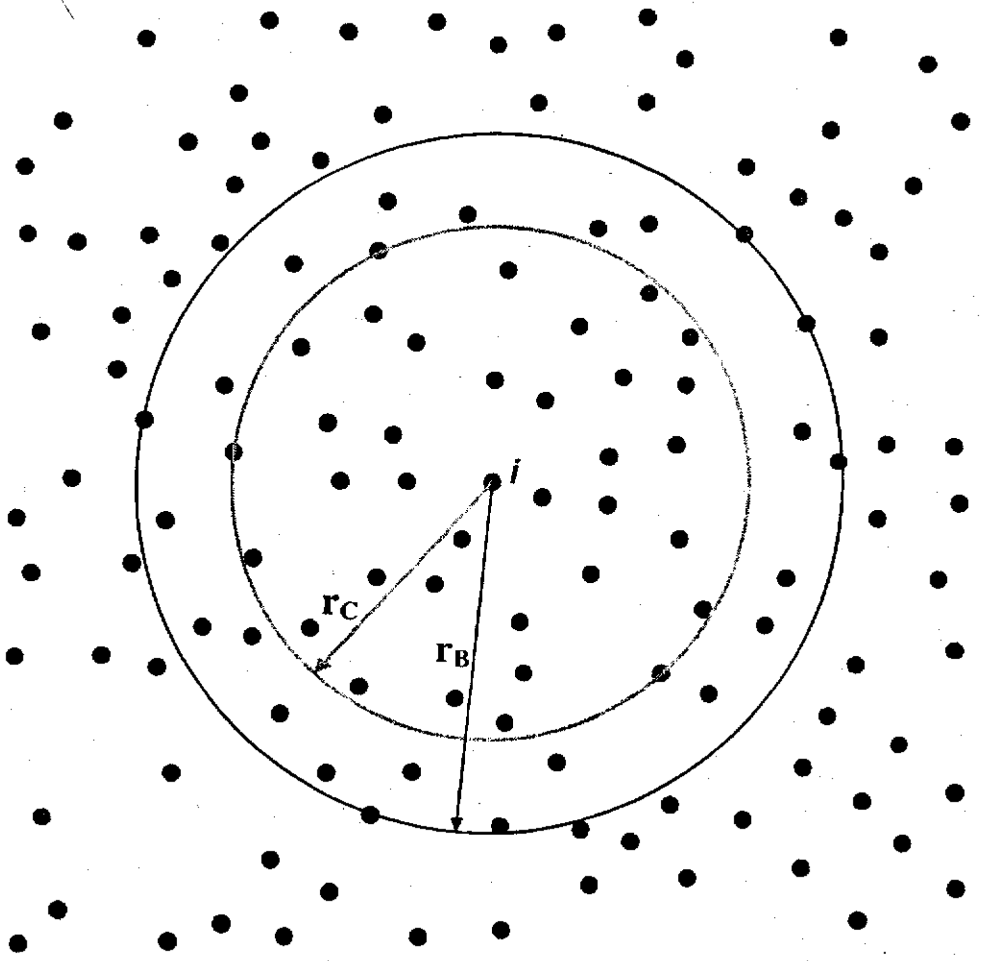

Figure 2 A particle i interacts mainly with particles that are within the cutoff radius r_C. The ''neighbor list'' contains only those particles that are within a sphere of radius $r_B > r_C$. Particles outside this sphere will not contribute to the force or energy affecting particle i. The use of a neighbor list that is periodically updated during the simulation reduces the computer time required in calculating pairwise interactions.

Other techniques can be found elsewhere in this volume. The following list includes pointers to several of these techniques:

1. Constant-temperature and constant-pressure simulations—Section VII.C, this chapter.
2. Constraint and multiple time step methods—Section VIII, this chapter.
3. Periodic boundary conditions—Chapter 5.
4. Long-range interactions and extended electrostatics—Chapter 5.
5. Solvation models—Chapter 7.

VI. ANALYSIS OF DYNAMIC TRAJECTORIES

An important issue, the significance of which is sometime underestimated, is the analysis of the resulting molecular dynamics trajectories. Clearly, the value of any computer simulation lies in the quality of the information extracted from it. In fact, it is good practice to plan the analysis procedure before starting the simulation, as the goals of the analysis will often determine the character of the simulation to be performed.

A. Time Series

The direct output of a dynamic simulation is a numerical trajectory, i.e., a series of system "snapshots" (coordinates and/or velocities) taken at equal time intervals $\Delta\tau$ from the full trajectory (the sampling interval $\Delta\tau$ is typically much larger than Δt). The size of the trajectory sampling interval, $\Delta\tau$, should be determined according to the time scale of the phenomenon that is being studied. For example, a 1 ps sampling interval may be a good choice for studying phenomena that take many tens of picoseconds but is clearly inappropriate for studying fast processes on the subpicosecond time scale.

Calculating any dynamic variable, $A(t)$, along the trajectory results in a "time series." Dynamic variables can be any function of coordinates and/or velocities. They may be relatively straightforward, such as total energy, specific bond lengths, or torsion angles of interest, or quite complex. Examples of the latter are the end-to-end distance in a protein (a quantity useful for studying protein folding), distances between a hydrogen bond donor and an acceptor, an angle formed between two helices, and so forth. The most straightforward analytical procedure is to plot the time series as a function of time. Such plots give a quick and easy overview of the simulation and are especially useful for picking up trends (e.g., drifts) or sudden transitions from one state to another.

Because a time series consists of instantaneous values taken at the trajectory sampling points, they tend to be very noisy. The level of noise can be reduced by simple smoothing procedures. A common smoothing procedure is to slide an "N-point window" along the data points and plot the "window averages" $\langle A(t)\rangle_N$ as a function of time [instead of plotting $A(t)$ itself]. The width of the window, N, and the suitability of the smoothing approach depend on the property that is being studied and on the ratio between the sampling interval $\Delta\tau$ and the characteristic time of the noise. For example, noise due to bond-stretching motions (time scale of 10–50 fs) can be smoothed by 0.5–1 ps windows. Alternatively, simply increasing the size of the sampling interval $\Delta\tau$ beyond the characteristic time scale of the "noise," to 0.5–1 ps in this case, will also reduce the effect of the noise.

B. Averages and Fluctuations

Time series plots give a useful overview of the processes studied. However, in order to compare different simulations to one another or to compare the simulation to experimental results it is necessary to calculate average values and measure fluctuations. The most common average is the root-mean-square (rms) average, which is given by the second moment of the distribution,

$$\langle A\rangle_{\text{rms}} = (\langle A^2\rangle)^{1/2} = \left[\frac{1}{N_S}\sum_{i=1}^{N_S}(A_i)^2\right]^{1/2} \tag{28}$$

where A is any dynamic variable and N_s is the number of "snapshots" in the trajectory. Root-mean-square fluctuations are calculated in the same way, with the fluctuation ΔA, which is described as a difference with respect to an average A, replacing the values A in Eq. (28).

Higher moments of the distribution are often of interest too, especially when nonisotropic or anharmonic processes are being studied. The third moment of the distribution reflects the skewness α_3 defined as

$$\alpha_3 = \langle A^3\rangle/\langle A^2\rangle^{3/2} \tag{29}$$

while the fourth moment reflects the excess kurtosis α_4 defined as

$$\alpha_4 = \langle A^4\rangle/\langle A^2\rangle^2 - 3 \tag{30}$$

Both α_3 and α_4 are zero for a Gaussian distribution.

C. Correlation Functions

A powerful analytical tool is the time correlation function. For any dynamic variable $A(t)$, such as bond lengths or dihedral angles, the time autocorrelation function $C_A(t)$ is defined as

$$C_A(t) = \langle A(t)\, A(0)\rangle \tag{31}$$

This function measures the correlation of the property $A(t)$ to itself at two different times separated by the time interval t, averaged over the whole trajectory. The auto-correlation function is *reversible* in time [i.e., $C_A(t) = C_A(-t)$], and it is *stationary* (i.e., $\langle A(t + t)\, A(t)\rangle = \langle A(t)\, A(0)\rangle$). In practical terms, the autocorrelation function is obtained by averaging the terms $\langle A(s + t)\, A(s)\rangle$ while sliding s along the trajectory.

A time "cross-correlation" function between dynamic variables $A(t)$ and $B(t)$ is defined in a similar way:

$$C_{AB}(t) = \langle A(t)\, B(0)\rangle \tag{32}$$

An important property of the time autocorrelation function $C_A(t)$ is that by taking its Fourier transform, $F\,\{C_A(t)\}_\omega$, one gets a spectral decomposition of all the frequencies that contribute to the motion. For example, consider the motion of a single particle in a harmonic potential (harmonic oscillator). The "time series" describing the position of the

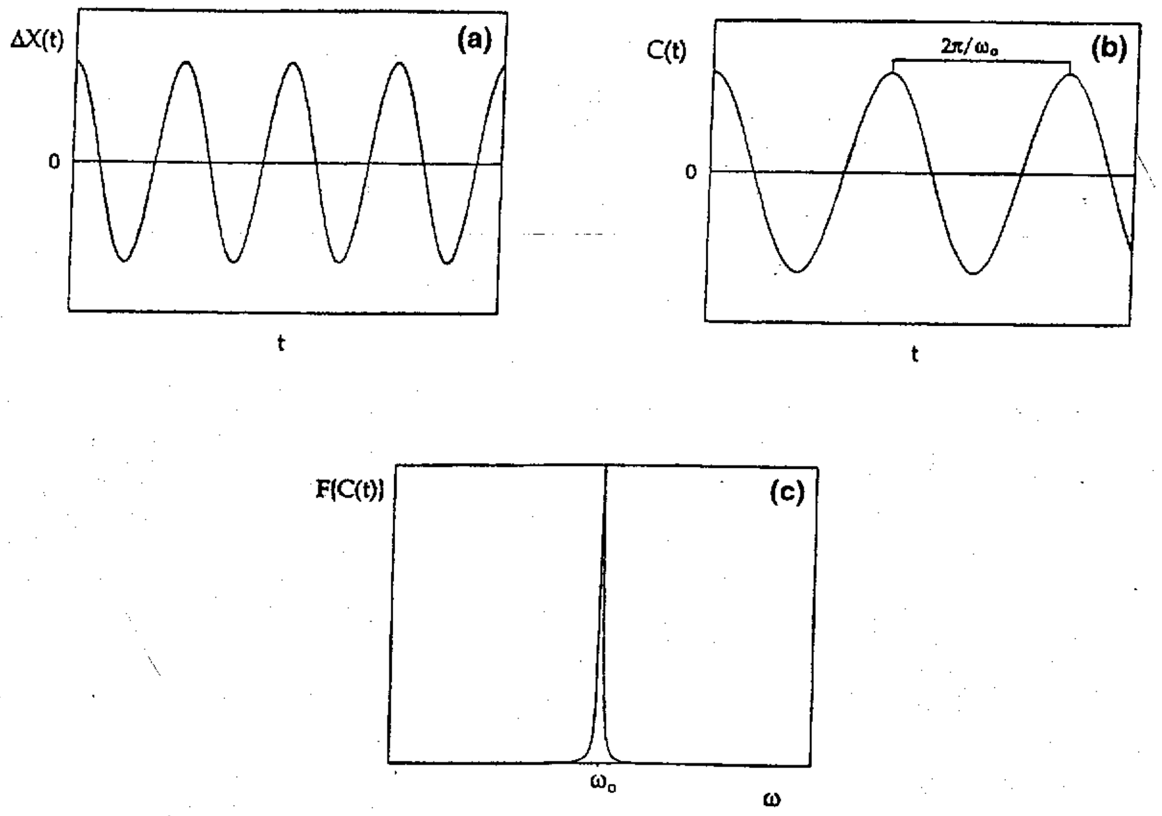

Figure 3 (a) A "time series" describing the position as a function of time of a particle moving in a one-dimensional harmonic well (harmonic oscillator). (b) The autocorrelation function of that motion; (c) its Fourier transform spectrum.

particle as a function of time is given by the $\cos(\omega_0 t)$ function (Fig. 3a). The autocorrelation function is given by a cosine function with a period $2\pi/\omega_0$ (Fig. 3b), and its Fourier transform gives a spectrum with a single sharp peak at ω_0 (Fig. 3c). The resulting frequency can be used to extract the (real or effective) local force constant $K_0 = m\omega_0^2$, where m is the mass of the system.

D. Potential of Mean Force

The potential of mean force is a useful analytical tool that results in an effective potential that reflects the average effect of all the other degrees of freedom on the dynamic variable of interest. Equation (2) indicates that given a potential function it is possible to calculate the probability for all states of the system (the Boltzmann relationship). The potential of mean force procedure works in the reverse direction. Given an observed distribution of values (from the trajectory), the corresponding effective potential function can be derived. The first step in this procedure is to organize the observed values of the dynamic variable, A, into a distribution function $\rho(A)$. From this distribution the "effective potential" or "potential of mean force," $W(A)$, is calculated from the Boltzmann relation:

$$W(A) = -RT \ln [\rho(A)] \tag{33}$$

The advantage of a potential of mean force is that it reflects the effect of environmental factors on the behavior of the dynamic variable of interest. Such an effect may be the damping of the solvent or a correlated fluctuation, which reduces effective transition barriers.

E. Estimating Standard Errors

Computer simulation is an experimental science to the extent that calculated dynamic properties are subject to systematic and statistical errors. Sources of systematic error consist of size dependence, poor equilibration, non-bond interaction cutoff, etc. These should, of course, be estimated and eliminated where possible. It is also essential to obtain an estimate of the statistical significance of the results. Simulation averages are taken over runs of finite length, and this is the main cause of statistical imprecision in the mean values so obtained.

Statistical errors of dynamic properties could be expressed by breaking a simulation up into multiple blocks, taking the average from each block, and using those values for statistical analysis. In principle, a block analysis of dynamic properties could be carried out in much the same way as that applied to a static average. However, the block lengths would have to be substantial to make a reasonably accurate estimate of the errors. This approach is based on the assumption that each block is an independent sample.

Another approach is to run multiple MD simulations with different initial conditions. The recent work by Karplus and coworkers [19] observes that multiple trajectories with different initial conditions improve conformational sampling of proteins. They suggest that multiple trajectories should be used rather than a single long trajectory.

VII. OTHER MD SIMULATION APPROACHES

A. Stochastic Dynamics

There are cases in which one is interested in the motion of a biomolecule but wishes also to study the effect of different solvent environments on this motion. In other cases, one may be interested in studying the motion of one part of the protein (e.g., a side chain or a loop) as moving in a ''solvent bath'' provided by the remainder of the protein. One way to deal with these issues is, of course, to explicitly include all the additional components in the simulation (explicit water molecules, the whole protein, etc.). This solution is computationally very expensive, because much work is done on parts of the system that are of no direct interest to the study.

Another way is to reduce the magnitude of the problem by eliminating the explicit solvent degrees of freedom from the calculation and representing them in another way. Methods of this nature, which retain the framework of molecular dynamics but replace the solvent by a variety of simplified models, are discussed in Chapters 7 and 19 of this book. An alternative approach is to move away from Newtonian molecular dynamics toward stochastic dynamics.

The basic equation of motion for stochastic dynamics is the Langevin equation,

$$m_i \ddot{r}_i = -\nabla_i U(\mathbf{r}) - m_i \beta_i v_i(t) + R_i(t) \tag{34}$$

which has three terms on the right-hand side. The first term accounts for molecular interactions and is the same as that used in Newtonian molecular dynamics [Eqs. (6) and (7). The other two terms are unique to stochastic dynamics and represent solvent effects. The second term is a dissipative drag force reflecting friction due to the solvent. This term is proportional to the velocity v_i and the friction coefficient β_i (which is related to the diffusion constant D_i). The third term is a random force $R_i(t)$ that represents stochastic collisions between solvent molecules and the solute. The stochastic force introduces energy into the system, and the friction force removes energy from the system (for further discussion see Ref. 20). The Langevin random force $R_i(t)$ on atom i is obtained from a random Gaussian distribution of zero mean and a variance related to the friction coefficient,

$$\langle R_i(t)\rangle = 0 \tag{35a}$$

$$\langle R_i(t)R_i(0)\rangle = 6m_i k_B T\delta(t) = 2D_i\,\delta(t) \tag{35b}$$

Sometimes an additional term, F_i^{mean}, is added to the right-hand side of Eq. (34). This "mean force" represents an average effect of degrees of freedom not explicitly treated in the simulation.

Because of the additional velocity-dependent forces (the dissipative forces), the straightforward finite-difference algorithms of the Verlet type cannot be used to integrate Langevin's equation. An algorithm may be derived that reduces to the Verlet algorithm in the limit of vanishing friction ($\beta_i \rightarrow 0$). This algorithm is obtained by adding the Langevin terms to the Verlet equation described in Eqs. (16) and (17). The resulting algorithm is of the order $O(\Delta t^3)$ and is valid for $\beta_i\,\Delta t < 1$.

B. Brownian Dynamics

Biomolecular motions that involve substantial displacements of the molecular surface, such as the motion of heavy particles in aqueous solution, are usually damped owing to the high viscosity of the surrounding solvent. In many such cases the damping effects are sufficiently great that internal forces are negligible and the motion has the character of a random walk. In other cases, such as the diffusion pairing of a substrate with its enzyme, the specific details of the internal dynamics are of little interest to the intermolecular motion. In such cases a further level of simplification can be introduced into the equation of motion. The relevant approximation applicable to such cases is the Brownian equation of motion, which is a diffusional analog of the molecular dynamics method. The Brownian equation can be easily derived from the stochastic Langevin equation presented in Eq. (34). If the inertial term on the left-hand side of the equation is small compared to the force terms on the right-hand side, it can be neglected, resulting in the diffusional Brownian equation of motion,

$$v_i(t) = \dot{r}_i = \frac{-\nabla V_i(\mathbf{r}) + F_i^{\text{mean}} + R_i(t)}{m_i\beta_i} \tag{36}$$

where the properties of the stochastic force $R_i(t)$ and its dependence on the friction coefficient D_i are given by Eqs. (35a) and (35b).

As with Newtonian molecular dynamics, a number of different algorithms have been developed to calculate the diffusional trajectories. An efficient algorithm for solving the Brownian equation of motion was introduced by Ermak and McCammon [21]. A detailed survey of this and other algorithms as well as their application can be found in Ref. 2.

C. Molecular Dynamics in Alternative Ensembles

The original molecular dynamics (MD) technique was used only to study the natural time evolution of a classical system of N particles in a volume V. In such simulations, the total energy, E, is a constant of motion, so that the time averages obtained by such a conventional MD simulation are equivalent to microcanonical ensemble averages. It is often scientifically more relevant to perform simulations in other ensembles, such as the canonical (NVT) that is associated with a Helmholtz free energy or the isothermal-isobaric (NPT) ensemble that is associated with a Gibbs free energy. Two rather different types of solutions to the problem of simulating alternative ensembles with the MD method have been proposed.

The first approach is based on introducing simple velocity or position rescaling into the standard Newtonian MD. The second approach has a dynamic origin and is based on a reformulation of the Lagrangian equations of motion for the system (so-called extended Lagrangian formulation.) In this section, we discuss several of the most widely used constant-temperature or constant-pressure schemes.

1. Constant-Temperature MD

The simplest method that keeps the temperature of a system constant during an MD simulation is to rescale the velocities at each time step by a factor of $(T_0/T)^{1/2}$, where T is the current instantaneous temperature [defined in Eq. (24)] and T_0 is the desired temperature. This method is commonly used in the equilibration phase of many MD simulations and has also been suggested as a means of performing "constant temperature molecular dynamics" [22]. A further refinement of the velocity-rescaling approach was proposed by Berendsen et al. [24], who used velocity rescaling to couple the system to a heat bath at a temperature T_0. Since heat coupling has a characteristic relaxation time, each velocity v is scaled by a factor λ, defined as

$$\lambda = \left[1 + \frac{\Delta t}{2\tau_T}\left(\frac{T_0}{T} - 1\right)\right]^{1/2} \tag{37}$$

In this expression, Δt is the size of the integration time step, τ_T is a characteristic relaxation time, and T is the instantaneous temperature. In the simulation of water, they found a relaxation time of $\tau_T = 0.4$ ps to be appropriate. However, this method does not correspond exactly to the canonical ensemble.

An alternative method, proposed by Andersen [23], shows that the coupling to the heat bath is represented by stochastic impulsive forces that act occasionally on randomly selected particles. Between stochastic collisions, the system evolves at constant energy according to the normal Newtonian laws of motion. The stochastic collisions ensure that all accessible constant-energy shells are visited according to their Boltzmann weight and therefore yield a canonical ensemble.

To carry out this method, values are chosen for T_0, the desired temperature, and ν, the mean frequency with which each particle experiences a stochastic collision. If successive collisions are uncorrected, then the distribution of time intervals between two successive stochastic collisions, $P(\nu, t)$, is of the Poisson form,

$$P(\nu, t) = \nu \exp(-\nu t) \tag{38}$$

A constant-temperature simulation now consists of the following steps:

1. Start with an initial set of positions and momenta and integrate the equation of motion.

2. The probability that any particular particle suffers a stochastic collision in a time interval of Δt is $\nu\ \Delta t$.
3. If particle i has been selected to undergo a collision, obtain its new velocity from a Maxwellian velocity distribution, defined in Eq. (23), corresponding to the desired temperature T_0. All other particles are unaffected by this collision.

Another popular approach to the isothermal (canonical) MD method was shown by Nosé [25]. This method for treating the dynamics of a system in contact with a thermal reservoir is to include a degree of freedom that represents that reservoir, so that one can perform deterministic MD at constant temperature by reformulating the Lagrangian equations of motion for this extended system. We can describe the Nosé approach as an illustration of an extended Lagrangian method. Energy is allowed to flow dynamically from the reservoir to the system and back; the reservoir has a certain thermal inertia associated with it. However, it is now more common to use the Nosé scheme in the implementation of Hoover [26].

To construct Nosé–Hoover constant-temperature molecular dynamics, an additional coordinate, s, and its conjugate momentum p_s are introduced. The Hamiltonian of the extended system of the N particles plus extended degrees of freedom can be expressed as

$$H_{\text{Nosé}} = \sum_{i=1}^{N} \frac{P_i^2}{2m_i s^2} + U(q) + \frac{P_s^2}{2Q} + \frac{g}{\beta} \ln s \tag{39}$$

where β is $1/k_BT$, Q is an effective "mass" associated with s, and g is a parameter related to the degrees of freedom in the system. The microcanonical distribution in the augmented set of variables in Eq. (39) is equivalent to a canonical distribution of the variables r_i and p_i/s. One of the disadvantages of the original Nosé approach, however, is that s can be interpreted as a scaling factor of the time step. This implies that the real time-step fluctuations occur during a simulation.

In a simulation it is not convenient to work with fluctuating time intervals. The real-variable formulation is therefore recommended. Hoover [26] showed that the equations derived by Nosé can be further simplified. He derived a slightly different set of equations that dispense with the time-scaling parameter s. To simplify the equations, we can introduce the thermodynamic friction coefficient, $\xi = p_s/Q$. The equations of motion then become

$$\dot{r}_i = \frac{p_i}{m_i} \tag{40a}$$

$$\dot{p}_i = -\frac{\partial U}{\partial r_i} - \xi p_i \tag{40b}$$

$$\dot{\xi} = \left(\sum_i \left(\frac{p_i^2}{m_i}\right) - \frac{g}{\beta}\right) Q^{-1} \tag{40c}$$

$$\frac{\dot{s}}{s} = \frac{d \ln s}{dt} = \xi \tag{40d}$$

Note that Eq. (40d), in fact, is redundant, because the other three equations form a closed set. Nonetheless, if we solve the equations of motion for s as well, we can use the following, $\overline{H}_{\text{Nosé}}$, as a diagnostic tool, because this quantity has to be conserved during the simulation even though $\overline{H}_{\text{Nosé}}$ is no longer a Hamiltonian:

$$\overline{H}_{\text{Nosé}} = \sum_{i=1}^{N} \frac{p_i^2}{2m_i} + U(q) + \frac{\xi 2Q}{2} + \frac{gs}{\beta} \tag{41}$$

where $g = 3N$ in this real-variable formulation.

Theoretically, static quantities are independent of the value chosen for the parameter Q. In practice, however, we observe that quantities are Q-dependent because of the finite number of MD simulation steps. Too high a value of Q results in slow energy flow between the system and reservoir, and in the limit $Q \to \infty$ we regain conventional MD. On the other hand, if Q is too low, the energy undergoes long-lived, weakly damped oscillations, resulting in poor equilibration. Nosé suggested the choice of the adjustable parameter Q for the physical system. It may, however, be necessary to choose Q by trial and error in order to achieve satisfactory damping.

Both Andersen and Nosé–Hoover methods, indeed, have generated canonical distributions. However, sometimes the Nosé–Hoover thermostat runs into ergodicity problems and the desired distribution cannot be achieved. The Andersen method does not suffer from such problems, but its dynamics are less realistic than Nosé–Hoover. To alleviate the ergodicity problems, Martyna et al. [27] proposed a scheme in which the Nosé–Hoover thermostat is coupled to another thermostat or, if necessary, to a whole chain of thermostats. The coupling ensures that the thermostats are allowed to fluctuate. In the original Nosé–Hoover method, the thermostat variable does not fluctuate. This generalization of the original Nosé–Hoover method is also shown to generate a canonical distribution, but this approach no longer faces the ergodicity problems. Details of the programming for this approach may be obtained from the book by Frenkel and Smit [7].

2. Constant-Pressure MD

In a normal molecular dynamics simulation with repeating boundary conditions (i.e., periodic boundary condition), the volume is held fixed, whereas at constant pressure the volume of the system must fluctuate. In some simulation cases, such as simulations dealing with membranes, it is more advantageous to use the constant-pressure MD than the regular MD. Various schemes for prescribing the pressure of a molecular dynamics simulation have also been proposed and applied [23,24,28,29]. In all of these approaches it is inevitable that the system box must change its volume.

To include the volume as a dynamic variable, the equations of motion are determined in the analysis of a system in which the positions and momenta of all particles are scaled by a factor proportional to the cube root of the volume of the system. Andersen [23] originally proposed a method for constant-pressure MD that involves coupling the system to an external variable, V, the volume of the simulation box. This coupling mimics the action of a piston on a real system. The piston has a "mass" M_v [which has units of (mass)(length)$^{-4}$]. From the Lagrangian for this extended system, the equations of motion for the particles and the volume of the cube are

$$\dot{r}_i = \frac{P_i}{m_i} + \frac{1}{3}\left(\frac{\dot{V}}{V}\right) r_i \tag{42a}$$

$$\dot{p}_i = F_i - \frac{1}{3}\left(\frac{\dot{V}}{V}\right)p_i \tag{42b}$$

$$\ddot{V} = \frac{1}{M_v}[P(t) - P_0] \tag{42c}$$

where V is the volume, $P(t)$ is the instantaneous pressure, P_0 is the desired pressure, and r_i, p_i, m_i, and F_i are the position, momentum, mass, and force, respectively, for each particle i. Andersen proved that the solution to these equations produces trajectories in the isobaric-isoenthalpic (NPH) ensemble where the particle number, pressure, and enthalpy of the system are constant. Here the choice of piston mass determines the decay time of the volume fluctuations. It has been proven that equilibrium quantities are independent of M_v, but in practice M_v influences the dynamic properties in the simulations. Even though there is no precise formulation for choosing M_v, Andersen suggests that the piston mass may be chosen by trial and error to satisfy the length of time required for a sound wave to travel through the simulation cell.

An alternative procedure rescales the coordinates of each atom at each time step [24]. The atomic coordinate x and the characteristic distance for repeating boundary conditions, d, are rescaled to values μx and μd, respectively, where

$$\mu = \left[1 - \frac{\Delta t}{\tau_P}(P_0 - P)\right]^{1/3} \tag{43}$$

Here, Δt is the size of the time step, τ_p is a characteristic relaxation time, and P_0 is the pressure of the external constant-pressure bath. The instantaneous pressure can be calculated as follows:

$$P = \frac{2}{3V}\left[E_k + \frac{1}{2}\sum_{i<j} r_{ij} \cdot F_{ij}\right] \tag{44}$$

where V is the volume and E_k is the kinetic energy, r_{ij} is the vector from particle i to particle j, and F_{ij} is the force on particle j due to particle i. In simulations of water, values of $\tau_p = 0.01–0.1$ ps were found suitable. This method does not drastically alter the dynamic trajectories and is easy to program, but the appropriate ensemble has not been identified. Therefore, the meaning of fluctuations in any observed quantity cannot be determined.

An algorithm for performing a constant-pressure molecular dynamics simulation that resolves some unphysical observations in the extended system (Andersen's) method and Berendsen's methods was developed by Feller et al. [29]. This approach replaces the deterministic equations of motion with the piston degree of freedom added to the Langevin equations of motion. This eliminates the unphysical fluctuation of the volume associated with the piston mass. In addition, Klein and coworkers [30] present an advanced constant-pressure method to overcome an unphysical dependence of the choice of lattice in generated trajectories.

In the foregoing treatments of pressure feedback, the simulation volume retains its cubic form, so changes consist of uniform contractions and expansions. The method is readily extended to the case of a simulation region in which the lengths and directions of the edges are allowed to vary independently. Parrinello and Rahman [31] and Nosé and Klein [32] extended the Andersen method to the case of noncubic simulation cells and derived a new Lagrangian for the extended system. Though their equations of motion are different from Andersen's original equations, they produce an identical ensemble. This technique is particularly helpful in the study of crystal structures of proteins, because it allows for phase changes in the simulation, which may involve changes in the unit cell dimensions and angles.

These constant-pressure MD methods can be combined with a suitable temperature control mechanism, as discussed in the previous section, to produce a more useful method to control both pressure and temperature simultaneously during the simulation. There are several approaches. The simplest approach is to use the scaling factors. The details of the algorithm are given by Berendsen et al. [24]. Another approach [25,26] is to define the appropriate extended Lagrangian for simultaneously coupling pressure and temperature to produce the isothermal-isobaric (*NPT*) ensemble. Hoover [26] presents a detailed description of the Nosé–Hoover constant-temperature method with Andersen's constant-pressure method. Even though this extended system method is slightly more complicated to program, this is the best candidate for conducting *NPT* ensemble MD. Details on programming for this approach are available in the book by Rapaport [8]. In addition, the new Langevin piston method [29] for constant pressure can be easily extended to couple a Nosé–Hoover thermostat to obtain a constant-pressure and constant-temperature method.

VIII. ADVANCED SIMULATION TECHNIQUES

Computer simulations have become a valuable tool for the theoretical investigation of biomolecular systems. Unfortunately, these simulations are often computationally demanding tasks owing to the large number of particles as well as the complex nature of their associated interactions. A longstanding problem, however, is that molecular dynamics is typically limited to a time scale of 10^{-6} s (1 μs) or less. In an MD simulation, the most rapidly varying quantities, such as the bond lengths, limit the integration time step, while the more slowly varying molecular processes are of primary interest (see Table 1) and determine the simulation length required. This would make the simulation of molecular substances very expensive.

A variety of techniques have been introduced to increase the time step in molecular dynamics simulations in an attempt to surmount the strict time step limits in MD simulations so that long time scale simulations can be routinely undertaken. One such technique is to solve the equations of motion in the internal degree of freedom, so that bond stretching and angle bending can be treated as rigid. This technique is discussed in Chapter 6 of this book. Herein, a brief overview is presented of two approaches, constrained dynamics and multiple time step dynamics.

A. Constrained Dynamics

To avoid the situation in which high frequency motions, such as bond stretching and bond angle bending which limits the integration time step, it is customary to eliminate such degrees of freedom entirely by the simple expedient of replacing them with constraints. In general, the dynamics could satisfy many constraints simultaneously, e.g., many bond lengths and bond angles. Assuming that a total of n distance constraints are imposed on a particular molecule, the constraint σ_k for a fixed distance d_{ij} between atom i and j can be expressed as

$$\sigma_k = r_{ij}^2 - d_{ij}^2 = 0, \qquad k = 1, \ldots, n \tag{45}$$

The equations of motion follow directly from the Lagrangian formulation containing all constraints. The result is

$$m_i \ddot{r}_i = F_i + G_i \tag{46}$$

where F_i is the usual force term, m_i the mass of the ith atom, and the additional term G_i expresses the effect of the constraints on atom i, G_i can be written

$$G_i = -\sum_{\alpha} \lambda_\alpha \frac{\partial \sigma_\alpha}{\partial r_i} \tag{47}$$

Here α denotes the set of constraints that directly involve r_i and the $\{\lambda_\alpha\}$ are the Lagrange multipliers introduced into the problem.

There are various ways to obtain the solutions to this problem. The most straightforward method is to solve the full problem by first computing the Lagrange multipliers from the time-differentiated constraint equations and then using the values obtained to solve the equations of motion [7,8,37]. This method, however, is not computationally cheap because it requires a matrix inversion at every iteration. In practice, therefore, the problem is solved by a simple iterative scheme to satisfy the constraints. This scheme is called SHAKE [6,14] (see Section V.B). Note that the computational advantage has to be balanced against the additional work required to solve the constraint equations. This approach allows a modest increase in speed by a factor of 2 or 3 if all bonds are constrained.

Although constrained dynamics is usually discussed in the context of the geometrically constrained system described above, the same techniques can have many other applications. For instance, constant-pressure and constant-temperature dynamics can be imposed by using constraint methods [33,34]. Car and Parrinello [35] describe the use of the extended Lagrangian to maintain constraints in the context of their ab initio MD method. (For more details on the Car–Parrinello method, refer to the excellent review by Galli and Pasquarrello [36].)

B. Multiple Time Step Methods

According to the nature of the empirical potential energy function, described in Chapter 2, different motions can take place on different time scales, e.g., bond stretching and bond angle bending vs. dihedral angle librations and non-bond interactions. Multiple time step (MTS) methods [38–40,42] allow one to use different integration time steps in the same simulation so as to treat the time development of the slow and fast movements most effectively.

Tuckerman et al. [38] showed how to systematically derive time-reversible, area-preserving MD algorithms from the Liouville formulation of classical mechanics. Here, we briefly introduce the Liouville approach to the MTS method. The Liouville operator for a system of N degrees of freedom in Cartesian coordinates is defined as

$$iL = [\ldots, H] = \sum_{i=1}^{N} \left[\dot{r}_i \frac{\partial}{\partial r_i} + F_i(r) \frac{\partial}{\partial p_i} \right] = \dot{r} \frac{\partial}{\partial r} + F(r) \frac{\partial}{\partial p} \tag{48}$$

where $[\ldots, \ldots]$ is the Poisson bracket, H is the Hamiltonian of the system, r_i and p_i are the position and conjugate momentum at coordinate i, $\dot{r}_i$ is the time derivative of r_i, and F_i is the force acting on the ith degree of freedom. The state of the system, Γ, at time Δt is then given by

$$\Gamma[r(\Delta t), p(\Delta t)] = U(\Delta t) \cdot \Gamma[r(0), p(0)] \tag{49}$$

where $U(t)$ is composed of a classical time evolution operator, $e^{iL\Delta t}$, and Γ could be any arbitrary function that depends on all coordinates and momenta of the system.

We decompose the Liouville operator into two parts,

$$iL = iL_1 + iL_2 \tag{50}$$

Unfortunately, we cannot replace $e^{iL\Delta t}$ by $e^{iL_1\Delta t}\, e^{iL_2\Delta t}$, because iL_1 and iL_2 are noncommutative operators. Applying Trotter's theorem [41], however, we can decompose the propagator, $U(\Delta t)$:

$$U(\Delta \mathrm{t}) = e^{iL\Delta t} = e^{iL_1\Delta t/2}\, e^{iL_2\Delta t}\, e^{iL_1\Delta t/2} \tag{51}$$

The idea is now to replace the formal solution of the Liouville equation by the discretized version. The middle term $e^{iL_2\Delta t}$ of the propagator in Eq. (51) can be further decomposed by an additional Trotter factorization to obtain

$$e^{iL_2\Delta t} = (e^{iL_2\Delta\tau})^n + O(n\Delta\tau^3) \tag{52}$$

where $\Delta t = n\ \Delta\tau$. Here the smaller time interval $\Delta\tau$ and the integer n determining the number of steps are chosen to guarantee stable dynamics for the system. Now Eq. (51) becomes

$$U(\Delta t) \approx e^{iL_1\Delta t/2}\, (e^{iL_2\Delta\tau})^n\, e^{iL_1\Delta t/2} \tag{53}$$

With the propagator written in this way, the equation of motion can be integrated by a multiple time step algorithm in Cartesian coordinates because Δt and $\Delta\tau$ are different integration time steps ($\Delta t > \Delta\tau$ when $n > 1$). As an example, the force terms are separated into two components

$$F(\mathbf{r}) = F_f(\mathbf{r}) + F_s(\mathbf{r}) \tag{54}$$

where F_f associates with "stiff" degrees of freedom or fast-varying forces, such as forces from bond-stretching, angle-bending motions, and F_s is associated with the rest of the contributions (i.e., slowly varying forces), such as forces from torsion motions and non-bond interaction. By introducing this decomposition into the Liouville operator we obtain

$$iL = \dot{r}\frac{\partial}{\partial r} + F_f(r)\frac{\partial}{\partial p} + F_s(r)\frac{\partial}{\partial p} \tag{55}$$

In this separation, the two Liouville operators, iL_1 and iL_2 of Eq. (50) can now be defined:

$$iL_2 = \dot{r}\frac{\partial}{\partial r} + F_f(r)\frac{\partial}{\partial p}; \qquad iL_1 = F_s(r)\frac{\partial}{\partial p} \tag{56}$$

The propagator $U(\Delta t)$ defined in Eq. (53) can now be implemented algorithmically as follows:

1. Starting with the initial state $[r(0), p(0)]$, generate the motion by using the propagator $e^{iL_1\Delta t/2}$.

2. Using the final state of step 1 as the initial state, generate the motion using the middle propagator $e^{iL_2\Delta\tau}$. Repeat this step n times.
3. Finally, starting with the state generated in step 2 as the initial state, generate the motion using the propagator $e^{iL_1\Delta t/2}$.

Corresponding implementations of the velocity Verlet operator can be easily derived for this Liouville propagator [38]. It should also be realized that the decomposition of iL into a sum of iL_1 and iL_2 is arbitrary. Other decompositions are possible and may lead to algorithms that are more convenient. One example is that in a typical MD simulation, a large portion of the computer processing time is spent in examining the non-bond pair interactions. These non-bond forces, therefore, can be divided into fast and slow parts based on distance by using a continuous switching function [42]. Applications of this MTS method to protein simulations have been shown to reduce the CPU time by a factor of 4–5 without altering dynamical properties [39,40,42]. In addition, this MTS approach shows significantly better performance enhancement in systems where the separation of fast and slow motions is pronounced [43].

C. Other Approaches and Future Direction

There are other approaches to obtaining efficiency in MD simulations. Examples include eigenvector-based schemes [44,45], implicit integration schemes [46], path optimization schemes [47], and a transition state theory approach [48]. Recently, a unique approach to overcome the time scale problems in MD was developed. Multibody order (N) dynamics [MBO(N)D] [49] is based on aggregating atoms of a macromolecule into groupings of interacting flexible and rigid bodies. Body flexibility is modeled by a truncated set of body-based normal modes. This approach allows for the unimportant high frequency modes of vibration, such as bond and angle motions, to be eliminated, leaving only the important lower frequency motions. This results in the use of a larger integration time step size, substantially reducing the computational time required for a given dynamic simulation. By coupling MBO(N)D with MTS described in the previous section, speed increases of up to 30-fold over conventional simulation methods have been realized in various MD simulations [49]. In addition to increasing computational efficiency, the approach also allows for a simplified analysis of dynamics simulations, as there are fewer degrees of freedom to consider.

Additionally, continuous developments of computer architectures, such as the clock speed of CPU chips and massive parallel computers, also help to carry out simulations of large biomolecules that require enormous computing power. In recent years, distributed memory parallel computers have been offering cost-effective computational power to researchers. This approach shows a great advantage in the size of the system (it is possible to run a million atoms in the system), although the simulation length is not scaled as well as the size because of the nature of solving equations of motion sequentially in time.

Finally, molecular modeling based on low resolution (coarse-grain) models has gained some attention in the field of biomolecular simulations [50]. This approach dramatically reduces the number of interaction sites by adapting a simple approach (e.g., a single site per residue) [51,52] or a multiple sites per residue approach (e.g., one backbone and one side chain interaction site per residue) [53,54]. These coarse-grain potentials are described by two categories: those based on statistical contact information derived from high resolution protein structures [51,52,54], and those base on established molecular mechanics force fields [53]. Coarse grain approaches are another way to gain a significant increase in speed and therefore begin to address large systems, such as protein–protein complexes, routinely.

Despite recent developments in algorithms and computer hardware, to bridge the gap between the time and size scales accessible by computer simulations and those required by experimental observations we still need to develop noble approaches.

REFERENCES

1. CL Brooks III, M Karplus, BM Pettitt, Proteins: A Theoretical Perspective of Dynamics, Structure and Thermodynamics. New York: Wiley, 1988.
2. JA McCammon, SC Harvey. Dynamics of Proteins and Nucleic Acids. Cambridge, UK: Cambridge Univ Press, 1987.
3. M Karplus, GA Petsko. Nature 347:631–639, 1990.
4. CL Brooks III, DA Case. Chem Rev 93:2487–2502, 1993.
5. WF van Gunsteren, HJC Berendsen. Angew Chem Int Ed Engl 29:992–1023, 1990.
6. MP Allen, DJ Tildesley. Computer Simulations of Liquids. New York: Oxford Univ Press, 1989.
7. D Frenkel, B Smit. Understanding Molecular Simulation from Algorithms to Applications. New York: Academic Press, 1996.
8. DC Rapaport. The Art of Molecular Dynamics Simulation. Cambridge, UK: Cambridge Univ Press, 1995.
9. WF van Gunsteren. Molecular dynamics and stochastic dynamics simulations: A primer. In: WF van Gunsteren, PK Weiner, AJ Wilkinson, eds. Computer Simulations of Biomolecular Systems. Leiden: ESCOM, 1993, pp 3–36.
10. SB Prusiner. Cell 93:337, 1998.
11. L Verlet. Phys Rev 159:98, 1967.
12. RW Hockney. Methods Comput Phys 9:136, 1970.
13. D Potter. Computational Physics. New York: Wiley, 1972, Chap 2.
14. W Swope, HC Andersen, H Berens, KR Wilson. J Chem Phys 76:637, 1992.
15. JP Ryckaert, G Ciccotti, HJC Berendsen. J Comput Phys 23:327, 1977.
16. WF van Gunsteren, HJC Berendsen. Mol Phys 34:1311, 1977.
17. HC Andersen. J Comput Phys 52:24, 1983.
18. WF van Gunsteren, M Karplus. Macromolecules 15:1528, 1982.
19. LSD Caves, JD Evanseck, M Karplus. Protein Sci 7:649, 1998.
20. F Reif. Fundamentals of Statistical and Thermal Physics. London: McGraw-Hill, 1965, Chap 15.5.
21. DL Ermak, JA McCammon. J Chem Phys 69:1352, 1978.
22. LV Woodcock. Chem Phys Lett 10:257, 1971.
23. HC Andersen. J Chem Phys 72:2384, 1980.
24. HJC Berendsen, JPM Postma, WF van Gunsteren, J Hermans. J Chem Phys 81:3684, 1984.
25. S Nosé. J Chem Phys 81:511, 1984.
26. WG Hoover. Phys Rev A 31:1695, 1985.
27. GJ Martyna, ML Klein, M Tuckerman. J Chem Phys 97:2635, 1992.
28. DJ Evans, GP Morriss. Comp Phys Repts 1:297, 1984.
29. SE Feller, Y Zhang, RW Pastor, BR Brooks. J Chem Phys 103:4613, 1995.
30. GJ Martyna, DJ Tobias, ML Klein. J Chem Phys 101:4177, 1994.
31. M Parrinello, A Rahman. Phys Rev Lett 45:1196, 1980.

32. S Nosé, ML Klein. Mol Phys 50:1055, 1983.
33. DJ Evans, GP Morriss. Phys Lett 98A:433, 1983.
34. DJ Evans, GP Morriss. Chem Phys 77:63, 1983.
35. R Car, M Parrinello. Phys Rev Lett 55:2471, 1985.
36. G Galli, A Pasquarello. First-principle molecular dynamics. In: MP Allen, DJ Tildesley, eds. Proceedings of the NATO ASI on Computer Simulation in Chemical Physics. Dordrecht: Kluwer, 1993, pp 261–313.
37. G Ciccotti, JP Ryckaert. Comp Phys Rep 4:345, 1986.
38. ME Tuckerman, BJ Berne, GJ Martyna. J Chem Phys 97:1990, 1992.
39. M Watanabe, M Karplus. J Chem Phys 99:8063, 1993, and references cited therein.
40. M Watanabe, M Karplus. J Phys Chem 99:5680, 1995.
41. HF Trotter. Proc Am Math Soc 10:545, 1959.
42. DD Humphreys, RA Friesner, BJ Berne. J Phys Chem 98:6885, 1994.
43. P Procacci, BJ Berne. J Chem Phys 101:2421, 1994.
44. A Amadei, ABM Linssen, BL deGroot, DMF vanAlten, HJC Berendsen. Biomol Struct Dynam 13:615, 1996.
45. BL deGroot, A Amadei, DMF vanAlten, HJC Berendsen. Biomol Struct Dynam 13:741, 1996.
46. CS Peskin, T Schlick. Commun Pure Appl Math 42:1001, 1989.
47. R Olender, R Elber. J Chem Phys 105:9299, 1996.
48. AF Voter. Phys Rev Lett 78:3908, 1997.
49. HM Chun, CE Padilla, DN Chin, M Watanabe, VI Karlov, HE Alper, K Soosaar, K Blair, O Becker, LSD Caves, R Nagle, DN Haney, BL Farmer. J Comput Chem 21:159, 2000.
50. T Haliloglu, I Bahar. Proteins 31:271, 1998.
51. M Hendlich, P Lackner, S Weitckus, H Floeckner, R Froschauer, K Gottsbacher, G Casari, MJ Sippl. J Mol Biol 216:167, 1990.
52. BA Reva, AV Finkelstein, MF Scanner, AJ Olson. Protein Eng 10:856, 1997.
53. T Head-Gordon, CL Brooks III. Biopolymers 31:77, 1991.
54. I Bahar, RL Jernian. J Mol Biol 266:195, 1997.