An analysis of the accuracy of Langevin and molecular dynamics algorithms

Richard W. Pastor , Bernard R. Brooks & Attila Szabo

Molecular Physics, 1988, Vol. 65, No. 6, 1409–1419

An analysis of the accuracy of Langevin and molecular dynamics algorithms

by RICHARD W. PASTOR

Biophysics Laboratory, Center for Biologics Evaluation and Research, Food and Drug Administration, 8800 Rockville Pike, Bethesda, Maryland 20892, U.S.A.

BERNARD R. BROOKS

Division of Computer Research and Technology, National Institutes of Health, Bethesda, Maryland 20892, U.S.A.

and ATTILA SZABO

Laboratory of Chemical Physics, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, Maryland 20892, U.S.A.

(*Received 8 March 1988; accepted 10 June 1988*)

Analytic expressions for mean squared positions and velocities of a harmonic oscillator are derived for Langevin dynamics algorithms valid in the high and low friction limits, and for the Verlet algorithm. For typical values of the parameters, errors in the positions are small. However, if the velocity is defined by the usual Verlet form, kinetic energies (and therefore calculated temperatures) can be in error by several per cent for the Langevin algorithms. If the Bunger–Brooks–Karplus algorithm is used to calculate positions, a simple redefinition of the velocity results greatly in improved kinetic energies. In addition, due to cancellation of errors in the velocities and the positions, the correct virial is obtained. The effect of including the force derivative in diffusive algorithms is examined. Positional and velocity averages are calculated for the Verlet algorithm for arbitrary initial conditions, and errors in the total energy and virial are analysed. Connection is made with the Langevin algorithms, and it is shown for harmonic oscillators that different definitions of the velocity are required to optimally calculate the temperature, pressure, and total energy, respectively.

1. Introduction

Computer simulations based on the Langevin equation have found a wide range of applications in both physical [1] and biopolymer [2] studies. As this is still an emerging field of research, however, a number of different Langevin dynamics algorithms are currently in use. These fall into three general classes depending on the product of the collision frequency γ (the friction constant divided by the mass) and the timestep Δ. The first class is valid when $\gamma\Delta \ll 1$. The second applies to the diffusive regime, where γ is large. A third class which is valid for arbitrary $\gamma\Delta$ has been developed [1, 4, 5] using the analytic solution of the Langevin equation for a particle in a constant force [6]. This makes it possible to increase the timestep to where it is limited by the systematic force; however random sampling from a bivariate distribution and considerable storage is required for the requisite coefficients and other terms. Unfortunately, since little error or stability analysis is available, it is often difficult to make a reasoned choice between algorithms, even within a particular class.

In the next two sections of this paper we present an analysis of the systematic errors that arise for the numerical solution of the Langevin equation, as opposed to statistical errors which are determined by the length of a simulation. For the most part the algorithms considered are accurate at low values of $\gamma\Delta$. For biopolymers such as proteins this limit does not pose a severe restriction. Here, due to the relatively stiff potential, a timestep of 0.001 ps or less is generally required, and from hydrodynamic arguments [7, 8] a collision frequency on the order of $50\,\mathrm{ps}^{-1}$ is appropriate for aqueous solutions at room temperature; hence $\gamma\Delta$ is typically 0·05. Based on an analysis of a finite difference approximation to the Langevin equation as applied to a free particle and particle in a constant field of force, three different algorithms are derived in § 2, each of which optimizes a different calculated quantity. These algorithms are applied to the harmonic oscillator in § 3. Since they were derived to accurately handle various features of the zero and constant force case, the harmonic potential is the simplest system that can be used to test them. It is found that one of the three, proposed originally by Brunger, Brooks and Karplus [9], is preferable provided that a Verlet velocity is not used. The dynamics of neat hydrocarbon chains generally takes place at high collision frequency [7], requiring intermediate to high $\gamma\Delta$ algorithms for accurate simulation. To this end, two diffusive or Brownian algorithms are briefly compared.

The Verlet algorithm is considered in § 4. Including its variants such as the leapfrog [10, 11], velocity Verlet [12] or Beeman [13], this algorithm is widely used in molecular dynamics simulations [14]. Based on the explicit solution of the finite difference approximation to the harmonic oscillator, the extent of non-conservation of energy and error in the virial is demonstrated for arbitrary initial conditions. As for the Langevin algorithms, optimal definitions of the velocity differ with the calculated quantity (temperature, pressure and total energy). Section 5 summarizes the work. An Appendix reviews the solution of the relevant difference equations.

2. Theory

2.1. *Defining an Algorithm*

The Langevin equation for a particle of mass m and friction constant ζ is [6]

$$m\, d^2x/dt^2 = F(t) - \zeta\, dx/dt + R(t), \tag{2.1}$$

where x is the position and F is the systematic force. The influence of the solvent or heat bath is modelled by the random force $R(t)$, which is assumed to be uncorrelated with the positions and velocities of the particles, and gaussian with a mean of zero and variance given by

$$\langle R(t)R(t')\rangle = 2m\gamma kT\delta(t - t'), \tag{2.2}$$

where k is Boltzmann's constant, T is the absolute temperature, and $\delta(t - t')$ is the Dirac delta function. The collision frequency $\gamma = \zeta/\mathrm{m}$.

Since $\gamma\Delta$ is small for many applications in biopolymers, it is of interest to explore algorithms valid in this regime. Further, because storage is a serious consideration for large systems such as proteins, we restrict this study to algorithms where only the current value of the force is required; thus, corrections using the derivative of the force are not included, though the methods presented in this paper can easily be extended to these cases [15]. Finally, we write the algorithm in a Verlet-like form, i.e. the positions are calculated directly and reduce to the Verlet algorithm in the limit of zero collision frequency. The choice has a number of advantages, including low requirements of computer resources and ease of implementation, particularly when dealing with systems where not all particles are explicitly stochastic. Thus, we consider algorithms that have the form†

$$x_{n+1} = x_n + a(x_n - x_{n-1}) + bR_n + cF_n, \tag{2.3}$$

where x_{n-1}, x_n and x_{n+1} are the positions at successive timesteps, and F_n are R_n are the systematic and random forces applied at step n, respectively. R_n is taken from a gaussian distribution of mean zero and variance

$$\langle R_n^2 \rangle = 2m\gamma kT/\Delta \tag{2.4}$$

and a, b and c are coefficients that depend only on γ, Δ and m. Given that we are seeking a numerical solution to (2.1) which will only be exact in the limit $\Delta \to 0$, it is permissible to choose these coefficients to optimize certain observables recognizing that others therefore might not be as well described. The quantities that we consider are

for the free particle (f.p.):
- (*a*) mean squared displacement at time t;
- (*b*) mean squared velocity;
- (*c*) velocity autocorrelation function;

for the particle in a constant field of force:
- (*d*) terminal velocity;

for the harmonic oscillator (h.o.):
- (*e*) mean squared position;
- (*f*) mean squared velocity;
- (*g*) the virial.

The unknown coefficients a, b and c in (2.3) are then determined by requiring a subset of these observables to match the analytic results. In § 2.2 we choose properties (*a*) and (*d*) to be satisfied. Then, by defining the velocity in two different ways and including (*c*), three different sets of coefficients for an algorithm of the form (2.3) are specified. Based on the results for (*e*)–(*g*), the optimal algorithm is determined in § 3.

2.2. *Defining the coefficients*

Setting $F = 0$ in (2.3), $\langle x_n^2 \rangle$, the mean squared displacement for the f.p. at step n, is calculated by solving the second order difference equation using standard methods as described in the appendix. Taking the appropriate limits leads to

$$\lim_{n\to\infty} \langle x_n^2 \rangle = nb^2 \langle R^2 \rangle/(1 - a)^2, \tag{2.5}$$

where $\langle \ldots \rangle$ signifies an ensemble average. Substitution of $t = n\Delta$ allows comparison with the exact long-time f.p. result:

$$\lim_{t\to\infty} \langle x(t)^2 \rangle = 2Dt = 2(kT/m\gamma)t. \tag{2.6}$$

† The most general three term finite approximation to $m\, d^2x/dt^2 + \zeta\, dx/dt$ is $cx_{n+1} + bx_n + ax_{n-1}$. Without loss of generality we set $c = 1$. If this approximation is required to be exact for $x =$ constant, then $1 + b + a = 0$, and hence a is the only free parameter.

Therefore, to calculate the mean squared displacement with no systematic error, we set

$$b = (1 - a)(\Delta/m\gamma). \tag{2.7}$$

We now turn to the velocity in order to determine the coefficient a. Here we define the velocity in two ways

$$u_n = (x_n - x_{n-1})/\Delta, \tag{2.8 a}$$

$$v_n = (x_{n+1} - x_{n-1})/2\Delta = (u_{n+1} + u_n)/2. \tag{2.8 b}$$

Equation (2.8 *b*) is equivalent to the commonly used Verlet velocity, however, as shown in §§ 3 and 4, (2.8 *a*) is preferable for applications where an accurate virial is required. (In specifying the Verlet leap-frog algorithm u_n has been denoted $v_{n-1/2}$ [14] or z_{n-1} [12].) To obtain $\langle u^2\rangle$ we first rewrite (2.3) as

$$u_{n+1} = au_n + bR_n/\Delta. \tag{2.9}$$

Squaring both sides, averaging, and using the relations

$$\langle u_{n+1}u_{n+1}\rangle = \langle u_n u_n\rangle = \langle u^2\rangle, \tag{2.10}$$

$$\langle u_n R_n\rangle = 0, \tag{2.11}$$

we find

$$\langle u^2\rangle_{\text{f.p.}} = \frac{b^2\langle R^2\rangle}{(1-a^2)\Delta^2}. \tag{2.12}$$

The discrete form of the velocity autocorrelation function is

$$C(n\Delta) = \langle u(0)u(n\Delta)\rangle = \langle u_1 u_{n+1}\rangle. \tag{2.13}$$

Multiplying (2.9) by u_1 and averaging

$$\langle u(0)u(n\Delta)\rangle = a\langle u(0)u((n-1)\Delta)\rangle. \tag{2.14}$$

The solution to this equation yields the velocity autocorrelation function

$$\langle u(0)u(n\Delta)\rangle_{\text{f.p.}} = \langle u^2\rangle a^n \tag{2.15}$$

Proceeding to $\langle v^2\rangle$, squaring both sides of (2.8 *b*) and using (2.14) we find

$$\langle v^2\rangle = (\langle u^2\rangle + \langle u(0)u(\Delta)\rangle)/2 \tag{2.16}$$

Then substituting (2.12) and (2.15) into (2.16)

$$\langle v^2\rangle_{\text{f.p.}} = \frac{b^2\langle R^2\rangle}{2\Delta^2(1-a)}. \tag{2.17}$$

It is then straightforward to show that

$$\langle v(0)v(n\Delta)\rangle_{\text{f.p.}} = \langle v^2\rangle a^n. \tag{2.18}$$

The coefficient a can now be defined in three ways, each of which leads to an exact result for one of the quantities indicated below when (2.4) is substituted for $\langle R^2\rangle$:

(i) $a = \dfrac{1 - \frac{1}{2}\gamma\Delta}{1 + \frac{1}{2}\gamma\Delta}$ (exact mean squared velocity using (2.8 *a*)),

(ii) $a = 1 - \gamma\Delta$ (exact mean squared velocity using (2.8 *b*)),

(iii) $a = \exp(-\gamma\Delta)$ (exact decay in (2.14) or (2.18)).

The remaining parameter c is determined by analysing the motion of a particle in a constant force. From (2.3)

$$\langle v \rangle_{\text{terminal}} = cF/(1-a)\Delta. \tag{2.19}$$

Noting from the Langevin equation (2.1) that the terminal velocity equals $F/m\gamma$, and using (2.7), we arrive at $b = c$. In summary, the three algorithms are

$$x_{n+1} = x_n + (x_n - x_{n-1})\frac{1-\frac{1}{2}\gamma\Delta}{1+\frac{1}{2}\gamma\Delta} + (\Delta^2/m)(F_n + R_n)(1+\tfrac{1}{2}\gamma\Delta)^{-1}, \tag{2.20}$$

$$x_{n+1} = x_n + (x_n - x_{n-1})(1-\gamma\Delta) + (\Delta^2/m)(F_n + R_n), \tag{2.21}$$

$$x_{n+1} = x_n + (x_n - x_{n-1})\exp(-\gamma\Delta) + (\Delta^2/m)(F_n + R_n)(1-\exp(-\gamma\Delta))/\gamma\Delta. \tag{2.22}$$

An equation having the form similar to (2.22) was proposed by Schneider and Stoll [16]; (2.20) was recently given by Bunger, Brooks and Karplus [9]. Equation (2.20) corresponds to the finite difference approximation to (2.3) in which

$$d^2x/dt^2 \approx (x_{n+1} - 2x_n + x_{n-1})/\Delta^2 \tag{2.23 a}$$

and

$$dx/dt \approx (x_{n+1} - x_{n-1})/2\Delta, \tag{2.23 b}$$

while (2.21) can be obtained using

$$dx/dt \approx (x_n - x_{n-1})/\Delta. \tag{2.24}$$

It is interesting to note that although (2.20) can be derived using the finite difference approximation (2.23 *b*) for the velocity, it gives the correct value of $\langle u^2 \rangle$ (see(2.8 *a*)) rather than $\langle v^2 \rangle$(see (2.8 *b*)) for a free particle. A similar reversal is found for (2.21).

3. The harmonic oscillator

3.1. *Langevin dynamics*

The preceding algorithms are now analysed for the harmonic oscillator. Setting $F_n = -m\omega^2 x_n$, (2.3) becomes

$$x_{n+1} = (1 + a - bm\omega^2)x_n - ax_{n-1} + bR_n. \tag{3.1}$$

It is convenient to define

$$\langle y^2 \rangle = \langle x_n x_{n-1} \rangle, \tag{3.2 a}$$

$$\langle z^2 \rangle = \langle x_{n+1} x_{n-1} \rangle. \tag{3.2 b}$$

These quantities result upon multiplying both sides of (3.1) by x_n or x_{n-1} and averaging. The mean squared displacement, $\langle x^2 \rangle$, is then obtained by squaring both sides, solving for $\langle y^2 \rangle$, and regrouping

$$\langle x^2 \rangle_{\text{h.o.}} = \frac{b(1+a)\langle R^2 \rangle}{2(1-a)(1+a-\frac{1}{2}m\omega^2 b)(m\omega^2)}. \tag{3.3}$$

The mean squared velocities can be calculated by squaring both sides of (2.8):

$$\langle u^2\rangle = (2/\Delta^2)(\langle x^2\rangle - \langle y^2\rangle), \tag{3.3 a}$$

$$\langle v^2\rangle = (1/2\Delta^2)(\langle x^2\rangle - \langle z^2\rangle). \tag{3.3 b}$$

Then, from (3.1)

$$\langle u^2\rangle_{\text{h.o.}} = \frac{b^2\langle R^2\rangle}{\Delta^2(1-a)(1+a-\frac{1}{2}m\omega^2 b)}, \tag{3.4 a}$$

$$\langle v^2\rangle_{\text{h.o.}} = \frac{b^2\langle R^2\rangle}{2\Delta^2(1-a)}. \tag{3.4 b}$$

Note that $\langle v^2\rangle$ is the same as for a f.p., while $\langle u^2\rangle$ depends on the frequency of the oscillator. Defining $\langle \hat{x}^2\rangle$ and $\langle \hat{v}^2\rangle$ to be the calculated values divided by the exact values for the h.o., $kT/m\omega^2$ and kT/m, respectively, the analytic results for (2.20) are

$$\langle \hat{x}^2\rangle_{\text{h.o.}} = (1 - \tfrac{1}{4}\omega^2\Delta^2)^{-1}, \tag{3.5 a}$$

$$\langle \hat{u}^2\rangle_{\text{h.o.}} = (1 - \tfrac{1}{4}\omega^2\Delta^2)^{-1} \tag{3.5 b}$$

$$\langle \hat{v}^2\rangle_{\text{h.o.}} = (1 + \tfrac{1}{2}\gamma\Delta)^{-1} \tag{3.5 c}$$

For (2.21)

$$\langle \hat{x}^2\rangle_{\text{h.o.}} = \left[1 - \frac{\omega^2\Delta^2}{4-2\gamma\Delta}\right]^{-1}, \tag{3.6 a}$$

$$\langle \hat{u}^2\rangle_{\text{h.o.}} = (1 - \tfrac{1}{2}\gamma\Delta - \tfrac{1}{4}\omega^2\Delta^2)^{-1}, \tag{3.6 b}$$

$$\langle \hat{v}^2\rangle_{\text{h.o.}} = 1 \tag{3.6 c}$$

and, finally, for (2.22)

$$\langle \hat{x}^2\rangle_{\text{h.o.}} = \left[1 - \frac{\omega^2\Delta}{2\gamma}\frac{1-\exp(-\gamma\Delta)}{1+\exp(-\gamma\Delta)}\right]^{-1}, \tag{3.7 a}$$

$$\langle \hat{u}^2\rangle_{\text{h.o.}} = \left[\frac{\gamma\Delta}{2}\frac{1+\exp(-\gamma\Delta)}{1-\exp(-\gamma\Delta)} - \frac{\omega^2\Delta^2}{4}\right]^{-1}, \tag{3.7 b}$$

$$\langle \hat{v}^2\rangle_{\text{h.o.}} = (1-\exp(-\gamma\Delta))/\Delta\gamma. \tag{3.7 c}$$

Assuming a h.o. frequency of $100\,\text{ps}^{-1}$, collision frequency of $50\,\text{ps}^{-1}$ and timestep of 0·001 ps, errors in $\langle x^2\rangle$ for all three algorithms are small, about 0·25 per cent. Errors in $\langle v^2\rangle$ are significantly larger for (2.20) and (2.22), about 2·5 per cent. Thus, for these algorithms the apparent temperature calculated from the kinetic energy would underestimate the bath or reference temperature used in (2.4); if a new temperature is defined with appropriate scaling for $\gamma\Delta$, the correct bath temperature is recovered. For (2.21) $\langle v^2\rangle$ is exact, although the error in $\langle x^2\rangle$ is slightly larger than for the other two algorithms.

Given that the errors in $\langle x^2\rangle$ are comparable in all three algorithms, and they are of equal simplicity, what criteria can be used to choose one of them? We note that (2.20) alone has the physically correct result that $\langle x^2\rangle$ is independent of γ. Further, $\langle \hat{x}^2\rangle - \langle \hat{u}^2\rangle$ equals zero independent of γ and ω; i.e. the average kinetic and potential energies are exactly equal for the h.o. if the velocity is calculated by (2.8 *a*). Hence the virial is exact. The virial, which is equal to twice the difference in the kinetic and potential energies for the h.o., is required for calculations of pressure. Thus, we recommend (2.20) with a velocity of the form (2.8 *a*).

3.2. *Diffusive dynamics*

For large γ (2.22) reduces to a diffusive or brownian (as opposed to Langevin) dynamics algorithm:

$$x_{n+1} = x_n + (\Delta/m\gamma)F_n + X_n. \tag{3.8}$$

The stochastic term, X_n, which represents a random displacement in the position, is taken from a gaussian distribution with zero mean and variance

$$\langle X_n^2\rangle = 2kT\Delta/(m\gamma). \tag{3.9}$$

Brownian dynamics simulations are carried out on systems where the collision frequency is high with respect to the relevant frequencies and motion is therefore diffusive (i.e. inertial corrections are small [17]). Taking the limit of (3.7 *a*) for large γ results in

$$\langle \hat{x}^2\rangle_{\text{h.o.}} = (1-\xi)^{-1} \tag{3.10}$$

where

$$\xi = \omega^2\Delta/2\gamma, \tag{3.11}$$

Thus, to obtain an error comparable to that resulting from the Langevin algorithms, the system must be considerably overdamped (a h.o. is overdamped when $(\gamma/2)^2 > \omega^2$ [6]). For example, for a h.o. with $\omega = 100\,\text{ps}^{-1}$ and $\Delta = 0{\cdot}001$ ps, only when γ is $2000\,\text{ps}^{-1}$ or greater is the error in $\langle x^2\rangle$ less than 0·25 per cent.

Lastly, we note that van Gunsteren and Berendsen [1] have proposed the diffusive algorithm

$$x_{n+1} = x_n + (\Delta/m\gamma)F_n + (\Delta^2/2m\gamma)\dot{F}_n + X_n, \tag{3.12}$$

where

$$\dot{F}_n = (F_n - F_{n-1})/\Delta \tag{3.13}$$

Then, using the procedure in § 3.1, it can be shown that

$$\langle \hat{x}^2\rangle_{\text{h.o.}} = \left[1 - \frac{2\xi^2}{(1-\xi)}\right]^{-1}. \tag{3.14}$$

The error is now second order in ξ. Thus, at the expense of including the derivative of the force, errors are greatly reduced when compared with (3.8); in fact, with $\omega = 100\,\text{ps}^{-1}$, $\Delta = 0{\cdot}001$ ps, and $\gamma = 200\,\text{ps}^{-1}$(critical damping), the error in $\langle x^2\rangle$ is 0·125 per cent.

4. Molecular dynamics: the Verlet algorithm

In the limit $\gamma \to 0$, (2.20)–(2.22) all reduce to the Verlet three-step algorithm commonly used in molecular dynamics simulations:

$$x_{n+1} = 2x_n - x_{n-1} + (\Delta^2/m)F_n. \tag{4.1}$$

Using methods described in the Appendix, the solution to (4.1) for a harmonic potential for initial positions x_0 and x_1 is

$$x_n = x_0 \cos n\theta + \frac{x_1 - \alpha x_0}{\omega\Delta\beta} \sin n\theta, \tag{4.2}$$

where

$$\alpha = 1 - \tfrac{1}{2}\omega^2 \Delta^2, \tag{4.3 a}$$

$$\beta = (1 - \tfrac{1}{4}\omega^2 \Delta^2)^{1/2}, \tag{4.3 b}$$

$$\theta = \cos^{-1} \alpha \approx \omega\Delta(1 + \tfrac{1}{24}\omega^2 \Delta^2). \tag{4.3 c}$$

Expressions equivalent to (4.2) have been recently derived independently by Amini, Eastwood and Hockney [18], and Venneri and Hoover [19]

In the most applications x_1 is calculated from an initial position x_0 and an initial velocity v_0 by a two-step formula:

$$x_1 = x_0 + v_0 \Delta + \tfrac{1}{2}(\Delta^2/m)F_0. \tag{4.4}$$

In this case (4.2) simplifies to

$$x_n = x_0 \cos n\theta + (v_0/\omega\beta) \sin n\theta. \tag{4.5 a}$$

The velocities u_n and v_n defined by (2.8 *a*) and (2.8 *b*) are

$$u_n = [\tfrac{1}{2}\omega^2 \Delta x_0 + v_0] \cos n\theta - [\omega\beta x_0 - \tfrac{1}{2}(\omega\Delta v_0/\beta)] \sin n\theta, \tag{4.5 b}$$

$$v_n = v_0 \cos n\theta - \omega\beta x_0 \sin n\theta. \tag{4.5 c}$$

Note from (4.5 *a*) that, even though the solution is stable as long as $\omega\Delta < 2$, the period is in error by approximately a factor of $(1 - \frac{1}{24}\omega^2 \Delta^2)$ compared to the exact result. Further, it is straightforward to show for the h.o. that the total energy

$$E_n = \tfrac{1}{2}m\omega^2 x_n^2 + \tfrac{1}{2}mv_n^2 \tag{4.6}$$

is *not* conserved on a point by point basis. However, if v_n is divided by β, the resulting expression for the total energy is

$$E_n' = \tfrac{1}{2}m\omega^2 x_n^2 + \tfrac{1}{2}m(v_n/\beta)^2 = \tfrac{1}{2}m\omega^2 x_0^2 + \tfrac{1}{2}m(v_0/\beta)^2 = \text{constant}. \tag{4.7 a}$$

In addition, we note that the following expression for the total energy

$$E_n'' = \tfrac{1}{2}m\omega^2 x_n^2 + \tfrac{1}{2}mu_{n+1}u_n = \tfrac{1}{2}m\omega^2 \beta^2 x_0^2 + \tfrac{1}{2}mv_0^2 = \beta^2 E_n' \tag{4.7 b}$$

is also conserved.

The time averages of x_n^2, v_n^2 and u_n^2 are given by

$$2\bar{x}^2 = x_0^2 + (v_0/\omega\beta)^2, \tag{4.8 a}$$

$$2\bar{u}^2 = \omega^2 x_0^2 + (v_0/\beta)^2, \tag{4.8 b}$$

$$2\bar{v}^2 = (\omega\beta x_0)^2 + v_0^2, \tag{4.8 c}$$

where the bar denotes the time average. It immediately follows from these equations

$$\omega^2 \bar{x}^2 = \bar{u}^2 = \bar{v}^2/\beta^2. \tag{4.9}$$

Thus, for arbitrary initial condition x_0 and v_0, the virial is satisfied if the velocity is calculated using either (2.8 *a*), or with (2.8 *b*) divided by β.

In a Langevin simulation the ensemble averages are independent of initial conditions, and the systematic errors are determined by the algorithm alone. In contrast, ensemble averages must be calculated by averaging over specific initial conditions for a molecular dynamics simulation, and will vary depending on how those initial conditions are chosen. For example, suppose one runs a large number of trajectories starting at arbitrary x_0, with v_0 sampled from the exact Maxwell–Boltzmann distribution (i.e. $\langle v_0^2 \rangle = kT/m$). The resulting time averages now depend only on x_0 and are given by

$$2\omega^2 \bar{x}^2 = \omega^2 x_0^2 + kT/m\beta^2, \tag{4.10 a}$$

$$2\bar{u}^2 = \omega^2 x_0^2 + kT/m\beta^2, \tag{4.10 b}$$

$$2\bar{v}^2 = \beta^2 \omega^2 x_0^2 + kT/m. \tag{4.10 c}$$

If one further averages over the exact Boltzmann distribution of starting points, the resulting time averages will all contain errors of order $\omega^2 \Delta^2$. This is simply a reflection of the fact that the integration algorithm is not exact. To obtain self-consistency of the initial positions and final average positions (i.e. $\langle x_0^2 \rangle = \bar{x}^2$) it follows from (4.10) that the initial positions should be sampled from a distribution with variance

$$\langle x_0^2 \rangle = kT/m\omega^2 \beta^2. \tag{4.11}$$

The resulting averages $\bar{u}^2$ and $\bar{v}^2$ are then $kT/m\beta^2$ and kT/m, respectively. Note that $\bar{v}^2$ equals $\langle v_0^2 \rangle$, and that the temperature of the system can be calculated accurately from $\bar{v}^2$. As can be seen from (3.5)–(3.7), these are precisely the results obtained from the Langevin algorithms in the limit of zero γ.

Finally, as has been pointed out by a number of authors [14, 18, 19], the trajectory generated by the Verlet algorithm is *identical* to one produced by variants such as the leap-frog, velocity Verlet and Beeman. Thus, the analysis presented here is directly applicable to these algorithms. In the leap-frog and velocity form, velocity terms are directly calculated in the generation of the trajectory as a way of reducing numerical round-off errors. This does not imply, however, that alternative definitions of the velocity introduced cannot be implemented at various stages of the analysis. The modification where the Verlet velocity is divided by β (4.7 *a*) is similar, though not identical, to a Beeman corrected velocity for a h.o. It is for this reason that the Beeman algorithm leads to better (though not exact) energy conservation than does the Verlet, even though the positions calculated with both algorithms are identical.

5. Summary and conclusions

For parameters that might typically be used in a Langevin simulation of a protein or other biopolymer in aqueous solution, the mean square displacement of a harmonic oscillator can be obtained with good accuracy by algorithms valid in the low $\gamma\Delta$ limit. We recommend (2.20), the algorithm of Brunger, Brooks and Karplus [9], to propagate positions; however, as summarized below, the calculation of velocities will depend on its usage. Diffusive or brownian algorithms were examined, and we recommend including the derivative of the force as in the van Gunsteren and Berendsen algorithm [1].

Positional averages were calculated for the Verlet algorithm. These depend on the initial conditions, and errors are specified. If the initial positions and velocities are chosen from (4.11) and the exact distributions, respectively, the final time averages are self-consistent. Further, they are equal to those calculated with the Langevin algorithms in the limit of zero γ.

It is not necessary to calculate a velocity when generating trajectories using either the Verlet algorithm or the Langevin algorithms discussed in this paper. In fact, since the random force serves to equilibrate the system, a calculation of temperature for the purpose of rescaling the velocities is inappropriate in a Langevin simulation. Nevertheless, the temperature, pressure, kinetic energy and total energy are all useful indicators of the stability of a simulation, and an evaluation of the velocity is required to calculate these quantities. In molecular dynamics simulations, the velocities are often scaled to a particular temperature during the heating and equilibration phases, and further, velocity autocorrelation functions are required for a hydrodynamic analysis of fluids [20]. We have shown that if the velocity is calculated in standard Verlet form, (2.8 *b*), it is not possible with the Langevin algorithms discussed here to calculate the virial correctly; i.e. the potential energy for a harmonic oscillator is not equal to the kinetic energy. Using (2.20) and (2.8 *a*) results in equal (and small) errors in both of these quantities, and therefore the correct virial. The same result is demonstrated for the Verlet algorithm: due to systematic errors in the potential energy, the virial for the harmonic oscillator is best calculated using (2.8 *a*). If the kinetic energy is calculated with either (2.8 *a*) or (2.8 *b*), the total energy of the oscillator is not conserved. However, a suitable redefinition of the kinetic energy as in (4.7 *a*) or (4.7 *b*) leads to constants of the motion for the Verlet algorithm. The extension of this analysis to more complex systems and to constant pressure and constant temperature molecular dynamics simulations is considered in [15].

Appendix A

Difference equations

The second order inhomogenous difference equation with constant coefficients,

$$x_{n+1} + Bx_n + Cx_{n-1} = G_n, \qquad n = 1, 2, \ldots, \tag{A 1}$$

can be readily solved subject to the initial conditions x_1 and x_0 [21]. The homogeneous equation is solved using the trial function $x_n = \lambda^n$. The unknown parameter λ satisfies a quadratic equation with roots $\lambda_\pm$ and hence

$$x_n = c_1 \lambda_+^n + c_2 \lambda_-^n. \tag{A 2}$$

The method of variation of parameters can then be used to solve the inhomogeneous equation. In this way one finds

$$x_n = x_0 \frac{\lambda_+^n \lambda_- - \lambda_+ \lambda_-^n}{\lambda_- - \lambda_+} + x_1 \frac{\lambda_-^n - \lambda_+^n}{\lambda_- - \lambda_+} + \sum_{k=1}^{n-1} G_k \frac{\lambda_-^{n-k} - \lambda_+^{n-k}}{\lambda_- - \lambda_+}, \tag{A 3}$$

where

$$\lambda_\pm = \frac{-B \pm (B^2 - 4C)^{1/2}}{2}. \tag{A 4}$$

To calculate $\langle x_n^2 \rangle$ as $n \to \infty$, we take $B = -(1+a)$, $C = a$, $G_k = bR_k$, $x_0 = x_1 = 0$. From (A 4) it follows that $\lambda_+ = 1$, $\lambda_- = a$. Squaring (A 3), averaging using the fact that $\langle R_k R_{k'} \rangle = \delta_{kk'} \langle R^2 \rangle$, and noting the $a^n \to 0$ as $n \to \infty$, since $a < 1$, we have

$$\lim_{n \to \infty} \langle x_n^2 \rangle = \frac{b^2 \langle R^2 \rangle}{(\lambda_- - \lambda_+)^2} \sum_{k=1}^{n} 1 = \frac{nb^2 \langle R^2 \rangle}{(1-a)^2} \tag{A 5}$$

in agreement with equation (2.5) of the text.

References

[1] van Gunsteren, W. F., and Berendsen, H. J. C., 1982, *Molec. Phys.*, **45**, 637, and references therein.

[2] See, for example, Allison, S. A., and McCammon, J. A., 1984, *Biopolymers*, **23**, 2173. McCammon, J. A., Northrup, S. H., Karplus, M., and Levy, R. M., 1980, *Biopolymers*, **19**, 2033. Lee, S., and Karplus, M., 1984, *J. chem. Phys.*, **81**, 6106. Brooks, C. L., Brunger, A., and Karplus, M., 1985, *Biopolymers*, **24**, 843. Pastor, R. W., Venable, R. M., and Karplus, M., 1988, *J. chem. Phys.*, **89**, 1112.

[3] Verlet, L., 1967, *Phys. Rev.*, **159**, 98.

[4] Ermak, D., and Buckholtz, H., 1980, *J. comput. Phys.*, **35**, 169.

[5] Allen, M. P., 1980, *Molec. Phys.*, **40**, 1073, 1982, *Ibid.*, **47**, 599.

[6] Chandrasekhar, S., 1943, *Rev. mod. Phys.*, **15**, 1.

[7] Pastor, R. W., and Karplus, M. 1988, *J. phys. Chem.*, **92**, 2636.

[8] Venable, R. M., and Pastor, R. W., 1988, *Biopolymers*, **27**, 1001.

[9] Brunger, A., Brooks, C. B., and Karplus, M., 1984, *Chem. Phys. Lett.*, **105**, 495.

[10] Buneman, O., 1967, *J. comput. Phys.*, **159**, 98.

[11] Hockney, R. W., 1970, *Meth. comput. Phys.*, **1**, 517.

[12] Swope, W. C., Anderson, H. C., Berens, P. H., and Wilson, K. R., 1982, *J. chem. Phys.*, **76**, 637.

[13] Beeman, D., 1976, *J. comput. Phys.*, **20**, 130.

[14] van Gunsteren, W. F., and Berendsen, H. J. C., 1986, *Proceedings of the Enrico Fermi School on Molecular-Dynamics Simulation of Statistical-Mechanical Systems*, edited by G. Ciccotti, and W. G. Hoover (North-Holland), pp. 43–65.

[15] Brooks, B. R. (in preparation).

[16] Schneider, T., and Stoll, E., 1977, *Phys. Rev.* **B**, **17**, 1302.

[17] Pastor, R. W., and Karplus, M., *J. chem. Phys.* (submitted).

[18] Amini, M., Eastwood, J. W., and Hockney, R. W., 1987, *Comput. Phys. Commun.*, **44**, 83.

[19] Venneri, G. D., and Hoover, W. G., 1987, *J. comput. Phys.*, **73**, 468.

[20] Boon, J. P., and Yip, S., 1980, *Molecular Hydrodynamics* (McGraw-Hill), and references therein.

[21] Bender, C. M., and Orszag, S. A., 1978, *Advanced Mathematical Methods for Scientists and Engineers* (McGraw-Hill), p. 49.