JOURNAL OF COMPUTATIONAL PHYSICS **129,** 87–100 (1996)
ARTICLE NO. 0235

Spatial Interpolation Methods for Integrating Newton's Equation

Shay Gueron*,[1] and David Shalloway†,[2]

*Center for Applied Mathematics, Cornell University, Ithaca, New York 14853; and †Biophysics Program, Section of Biochemistry, Molecular and Cell Biology, Cornell University, Ithaca, New York 14853

Received May 13, 1996

Numerical integration of Newton's equation in multiple dimensions plays an important role in many fields such as biochemistry and astrophysics. Currently, some of the most important practical questions in these areas cannot be addressed because the large dimensionality of the variable space and complexity of the required force evaluations precludes integration over sufficiently large time intervals. Improving the efficiency of algorithms for this purpose is therefore of great importance. Standard numerical integration schemes (e.g., leap-frog and Runge–Kutta) ignore the special structure of Newton's equation that, for conservative systems, constrains the force to be the gradient of a scalar potential. We propose a new class of "spatial interpolation" (SI) integrators that exploit this property by interpolating the force in space rather than (as with standard methods) in time. Since the force is usually a smoother function of space than of time, this can improve algorithmic efficiency and accuracy. In particular, an SI integrator solves the one- and two-dimensional harmonic oscillators exactly with one force evaluation per step. A simple type of time-reversible SI algorithm is described and tested. Significantly improved performance is achieved on one- and multi-dimensional benchmark problems.

1. INTRODUCTION

Newton's equation with a force F derived from a scalar potential $U(x)$ in a space of N variables is

$$\frac{d^2x}{dt^2} = F(x) = -\nabla[U(x)]. \tag{1}$$

For clarity we assume that the mass factors normally present in Newton's equation have been absorbed by rescaling x and redefining $U(x)$. Here and throughout the paper x and F (and later v and a) refer to N-dimensional vectors.

Motivation for developing improved numerical techniques for solving Newton's equation comes from the widespread use of molecular dynamics (MD) programs that simulate dynamics by numerical integration. Two typical applications are simulations in astrophysics and in the macromolecular dynamics of proteins in which the components of x describe the positions in space of large numbers of objects (e.g., stars or atoms). In particular, protein MD has applications in biochemistry, protein-folding, structure-based drug design and biotechnology, and it is a major application for parallel computers.

The practical difficulty with MD simulations in these problems is that computational cost restricts the time period that can be integrated with feasible computational effort. The dominating cost comes from evaluating F; this has $O(N^2)$ complexity even for the simplest potentials containing only two-body terms and N is typically $\sim O(10^4)$. (In some, but not all, problems this can be reduced to $O(N \log N)$ by using fast summation algorithms [1], but evaluation of F still dominates computational cost.) As a result, the most important practical questions cannot be addressed at the present time.

For example, the substrate-enclosing "flaps" of the human immunodeficiency virus (HIV) protease protein, an important target for structure-based drug design, have important motions in the microsecond time domain [2]. An understanding of these dynamic readjustments is important for understanding the mechanisms by which mutants of this protein acquire drug-resistance. This protein contains $\sim O(10^4)$ atoms whose spatial positions are specified by three times as many coordinates. One would like to also include roughly an equal number of water molecules in the simulation to account for solvent effects. Standard protein conformational potentials (e.g., CHARMM [3] and AMBER [4]) require $\sim O(10)$ flops per pair of atoms, so a complete force evaluation requires $\sim O(10^9)$ flops. Even allowing for common simplifying approximations,[3] $\sim O(10^7$–$10^8)$ flops per force evaluation are needed. For numerical stability, the maximum allowable time-step of

[1] Current address: Dept. of Mathematics, Technion-Israel Institute of Technology, Haifa, 32000, Israel. E-mail: shay@math.technion.ac.il.
[2] E-mail: dis2@cornell.edu.

[3] The complexity of protein force evaluations is commonly reduced by eliminating the degrees-of-freedom associated with hydrogen atoms, using a "cutoff" to limit the distance range over which atomic interactions are considered and reducing the number of solvent molecules to a bare minimum. These techniques are probably overexploited; at present, it is not uncommon to encounter questionable simulations performed with overly short cutoffs and insufficient numbers of solvent molecules in an effort to extend simulation times, regardless of accuracy [17, 18].

0021-9991/96 $18.00

a simulation which includes all atomic motions is restricted to be $\sim O(10^{-15})$ s, the period of the fastest oscillations of the system. While special methods can be used to increase this by a factor of 2–5 [5–8], the minimum overall complexity of a typical simulation for a molecule like HIV protease is still $\sim O(10^{22})$ flops/s simulation time. Currently, these problems can only be effectively parallelized to $O(10)$ processors [10], so even long ($\sim$1 week) simulations on parallel supercomputers ($\sim O(10^7)$ flops/s/processor) can only simulate periods of $\sim O(10^{-9}$–$10^{-8})$ s, much less than desired.

Most simulation programs [3, 4] solve Newton's equation using standard methods for integrating second-order differential equations. Because of the high cost of force evaluations, practical experience indicates that methods that evaluate the force only once per step (e.g., leap-frog (LF)) are the most cost-effective and are used in most available MD software [5]. As a result, higher-order methods that use several force evaluations per step (e.g., Runge–Kutta (RK)) are not generally applied in this context. Because of the need to maximize the total simulation period, it is common practice to increase the size of the individual time-steps until just below the point where computational instabilities are encountered. As a result, accuracy is determined chiefly by truncation and not roundoff error. Regretfully, the resultant accuracy of computations with such large time-steps is arguable.

Because of the extensive use of such simulations and their importance, a number of groups worldwide are working to develop new algorithms and parallelization strategies for extending the computationally accessible time domain. Several numerical methods have recently been suggested (see [9] for a review), but still any improvements in accuracy and efficiency are crucial.

Standard numerical integrators are designed to solve general ordinary differential equations (ODEs) and therefore do not make use of the special structure of Newton's equation (1) in which the force is the gradient of a scalar potential. As a result of this structure, the force depends on time only implicitly, through the dependence of x on t. In this paper we present a new class of integrators that exploit this property by interpolating the force in space rather than (as with standard methods) in time. Since the force is usually a smoother function of space than of time (with scale conversion determined by the velocity), it can be more accurately modeled from the limited information available to the integrating algorithm when viewed as a function of x than when viewed as a function of t. We expect that integrating algorithms based on this *spatial interpolation* (SI) principle can achieve improved efficiency and accuracy.

We demonstrate this by testing simple SI algorithms on one- and multi-dimensional test problems. Like LF, these algorithms only require one force evaluation per step. Since LF is the most commonly used MD algorithm, we use it as the primary basis for comparison, although some comparisons with a fourth-order RK algorithm are also made. We show that, in contrast to the LF and RK methods, the SI method exactly solves the one- and two-dimensional harmonic oscillator and that it gives superior results when tested on multidimensional harmonic and anharmonic oscillator test problems.

2. MOTIVATION AND OUTLINE OF A QUADRATIC SI ALGORITHM

2.1. *Spatial vs Temporal Interpolation of the Force*

We first observe that the standard methods (e.g., LF and RK) for solving Newton's equation (1) are actually second-order ODE solvers for equations of the type

$$\frac{d^2x}{dt^2} = F(t). \tag{2}$$

That is, they work for any arbitrary (vector) force function $F(t)$. However, F in (1) has the special form of the gradient of a potential. This implies that it depends on the time only implicitly, through its dependence on x. Our approach to improving efficiency is to exploit this special form $F(t) \to F[x(t)]$.

To better understand SI algorithms, it is important to note that general-purpose second-order ODE solvers implicitly use some means to interpolate F in t. To illustrate and see later how SI differs, it is useful to review the derivation of the LF algorithm. This is based on the Taylor series expansion at $t = t_0$:

$$x(t_0 + \Delta t) = x(t_0) + \Delta t\, v(t_0) + \tfrac{1}{2}(\Delta t)^2 a(t_0) + O[(\Delta t)^3] + O[(\Delta t)^4] \tag{3}$$

$$x(t_0 - \Delta t) = x(t_0) - \Delta t\, v(t_0) + \tfrac{1}{2}(\Delta t)^2 a(t_0) - O[(\Delta t)^3] + O[(\Delta t)^4]. \tag{4}$$

Here, $v(t_0)$ and $a(t_0)$ are the velocity and the acceleration at $t = t_0$, respectively. Adding (3) and (4) yields

$$x(t_0 + \Delta t) + x(t_0 - \Delta t) = 2x(t_0) + \tfrac{1}{2}(\Delta t)^2 a(t_0) + O[(\Delta t)^4]. \tag{5}$$

This symmetric combination in time of the equations for $x(t_0 + \Delta t)$ and $x(t_0 - \Delta t)$ is the key to the better performance of LF relative to the simpler second-order Euler method. Using Newton's equation to replace $a(t)$ with $F[x(t)]$, we get the following.

LF Algorithm.
INPUT: a trajectory point $x(t_0)$ given at $t = t_0$, the force $F[x(t_0)]$ evaluated at $x(t_0)$, and a previous trajectory point $x(t_0 - \Delta t)$.
OUTPUT: $x(t_0 + \Delta t)$ calculated by

$$x(t_0 + \Delta t) = 2x(t_0) - x(t_0 - \Delta t) + (\Delta t)^2 F[x(t_0)]. \quad (6)$$

This is equivalent to treating the second-order equation (1) as a set of coupled first-order equations by first using $F[x(t_0)]$ to calculate the velocity v at the half-way time point $t_0 + \Delta t/2$,

$$v(t_0 + \Delta t/2) = v(t_0 - \Delta t/2) + \Delta t\, F[x(t_0)], \quad (7)$$

and then using the computed value of $v(t_0 + \Delta t/2)$ to calculate x at the time point $t_0 + \Delta t$,

$$x(t_0 + \Delta t) = x(t_0) + \Delta t\, v(t_0 + \Delta t/2). \quad (8)$$

Note that (7) amounts to solving the first-order equation for the velocity, $dv/dt = F(t)$ by using the step-wise constant interpolation for F,

$$F(t) = F[\Delta t\, \mathrm{Round}(t/\Delta t)], \quad (9)$$

where Round(r) denotes the integer closest to r. The accuracy of this interpolation ultimately limits the accuracy of the LF method. Unfortunately, even for the simplest problems, $F(t)$ is a complicated and rapidly varying function of t. As a result, a constant interpolation of F in t will generally be valid only over very short time intervals, and methods based on such interpolations require very small time-steps for reasonable accuracy.

This point is illustrated by considering the one-dimensional harmonic oscillator described by (1) with force

$$F(x) = -k(x - x_0) \quad (10)$$

for some $k > 0$. Its analytic solution with initial conditions $x(t_0)$, $v(t_0)$ is

$$x(t) = x_0 + \cos[\sqrt{k}\,(t - t_0)][x(t_0) - x_0] + (1/\sqrt{k}) \sin[\sqrt{k}\,(t - t_0)]v(t_0). \quad (11)$$

Even in this simple case, $F[x(t)] \to F(t)$ has the complicated form

$$F(t) = -k\{\cos[\sqrt{k}\,(t - t_0)][x(t_0) - x_0] + (1/\sqrt{k}) \sin[\sqrt{k}\,(t - t_0)]v(t_0)\} \quad (12)$$

so the LF interpolation, described by (9), is a rather poor approximation, especially for large Δt. Consequently, LF gives a solution to the harmonic oscillator [11]

$$x_{\mathrm{LF}}(n\,\Delta t) = x_0 + \cos(\omega_{\mathrm{LF}} n\,\Delta t)[x(t_0) - x_0] + (1/\sqrt{k}) \sin(\omega_{\mathrm{LF}} n\,\Delta t)v(t_0) \quad (13)$$

$$\omega_{\mathrm{LF}} \equiv (\Delta t)^{-1} \arccos[1 - k(\Delta t)^2/2] \quad (14)$$

which has a frequency error relative to the true frequency $\omega = \sqrt{k}$ of

$$\frac{\omega_{\mathrm{LF}} - \omega}{\omega} \approx \frac{1}{24} k(\Delta t)^2 + \frac{3}{640} k^2 (\Delta t)^4 + \cdots. \quad (15)$$

This solution diverges for $\Delta t \geq 2/\sqrt{k}$. Qualitatively similar behavior is expected for anharmonic problems but precise analysis is difficult in those cases.

The key to our approach is to note that while F is complicated when viewed as a function of t, it may be simple when viewed as a function of x. This is exemplified by comparing (10) and (12). This suggests that more accurate results can be obtained by interpolating F in x rather than in t. We refer to the general class of algorithms that employ this approach as SI methods.

2.2. *Quadratic SI Algorithms*

In this paper we consider a restricted class of *quadratic SI* algorithms in which we: (1) approximate F as a linear function in x (i.e., U is approximated by a quadratic function), and then (2) integrate Newton's equation analytically for this approximate function. More specifically, for a step beginning at $x(t_0)$,

1. Calculate a symmetric matrix K that governs the linear approximation $f(x)$ to $F(x)$ in the region surrounding $x(t_0)$,

$$F(x) \approx f(x) \equiv F_{t_0} - K[x - x_p(t_0)], \quad (16)$$

where $x_p(t_0)$ is a point near $x(t_0)$ (we will see below why x_p and $x(t_0)$ are not identical) and

$$F_{t_0} \equiv F[x_p(t_0)]. \quad (17)$$

f has the property $f[x_p(t_0)] = F[x_p(t_0)]$. The symmetricity of K follows from the fact that F is the gradient of a scalar potential. The procedure for calculating it will be described in the next section.

2. Solve the simplified Newton's equation $d^2x/dt^2 =$

$f(x)$ with initial conditions $x(t_0)$, $v(t_0)$. The analytic solution is

$$x(t) = x(t_0) + K^{-1} \cdot \{1 - \cos[\sqrt{K}(t - t_0)]\} \cdot \{F_{t_0} - K \cdot [x(t_0) - x_p(t_0)]\} + K^{-1/2} \cdot \sin[\sqrt{K}(t - t_0)] \cdot v(t_0). \quad (18)$$

Mimicking the derivation of the LF method, we add the solutions for $x(t_0 + \Delta t)$ and $x(t_0 - \Delta t)$ calculated from (18) to obtain

$$x(t_0 + \Delta t) = 2x(t_0) - x(t_0 - \Delta t) + (\Delta t)^2 \xi(\sqrt{K}\,\Delta t) \cdot \{F_{t_0} - K \cdot [x(t_0) - x_p(t_0)]\}, \quad (19)$$

where

$$\xi(\theta) \equiv 2\theta^{-2} \cdot [1 - \cos(\theta)]. \quad (20)$$

The cosine of the matrix $\sqrt{K}\,\Delta t$ that is required to evaluate ξ can be computed in the vector basis that diagonalizes K. Since, as discussed in Section 4, the rank of K is small even for large-dimensionality problems, this is always inexpensive. The apparent singularity in ξ occurring when K is singular is removable and causes no difficulty.

2.3. *Calculating K*

Time-reversibility is a fundamental property of conservative Hamiltonian systems described by Newton's equation and should be preserved by the trajectories computed by the appropriate numerical integrating algorithms. This is the case for each step of LF since (6) is invariant under $\Delta t \to -\Delta t$. Thus, if a trajectory propagated forwards in time from initial conditions[4] $[x(-\Delta t), x(0)]$ goes to $[x(T - \Delta t), x(T)]$, then the trajectory propagated backwards in time, starting from the initial conditions $[x(T - \Delta t), x(T)]$, will arrive back at $[x(-\Delta t), x(0)]$. The SI step equation (19) is also invariant under time-reversal, so the overall SI algorithm will be invariant as long as K is calculated by a time reversal-invariant method.

Different methods, using information obtained from calculating the force at one, two, three, or more points near x_0, can be imagined. Each will lead to a different SI algorithm. For example, using only F_{t_0} would be equivalent to assuming that the force is constant (i.e., $F = F_{t_0}$) over the interval $[t_0 - \Delta t, t_0 + \Delta t]$. This is equivalent to choosing $K = 0$ in (16). In this case, $\xi(\sqrt{K}\,\Delta t) \to 1$ and (19) reduces to (6). That is, the trivial *one-point* SI algorithm is identical to the LF algorithm. It follows that a nontrivial SI algorithm must use the values of F evaluated for at least two points to calculate K.

[4] Specifying the initial position pair is equivalent to specifying an initial position and velocity.

We could consider a two-point method that used $F[x(t_0)]$ and $F[x(t_0 - \Delta t)]$ to fix K. This approach has two deficiencies. First, the solution of Newton's equation by (19) would involve the use of an *extrapolation* of F into the region between $x(t_0)$ and $x(t_0 + \Delta t)$. This is undesirable since, with the relatively large time-steps used in typical calculations, the extrapolation could be less accurate than the constant F approximation used by LF. Second, this method would not be time-reversible; the value of K used for propagation forward in time from $x(t_0)$ to $x(t_0 + \Delta t)$ would depend on $F[x(t_0)]$ and $F[x(t_0 - \Delta t)]$, while the value of K used for propagation backward in time from $x(t_0)$ to $x(t_0 - \Delta t)$ would depend on $F[x(t_0)]$ and $F[x(t_0 + \Delta t)]$. Because the forward and backward K's would differ, the trajectory would not be retraced under time-reversal.

To overcome these deficiencies, we must calculate K using a time-reversal invariant method which uses forces evaluated at points that lie both forward (i.e., in the future) and backward (in the past) along the trajectory. The problem is how to get information in the forward direction. This is accomplished by using a predictor–corrector method in which: (1) a time-reversible predictor step is taken from $x(t_0)$ to a *predictor point* $x_p(t_0 + \Delta t)$; (2) $F_{t_0+\Delta t} \equiv F[x_p(t_0 + \Delta t)]$ is evaluated at this point; and (3) K is calculated using the three forces $F_{t_0-\Delta t}$, F_{t_0}, and $F_{t_0+\Delta t}$ (where $F_{t_0-\Delta t}$ and F_{t_0} are the forces that had been evaluated at prior predictor points). As long as $F_{t_0-\Delta t}$ and $F_{t_0+\Delta t}$ are treated symmetrically, this *three-point calculation* of K will be time-reversal invariant. We can then calculate a time-reversible corrector step, using the new K and (19) to move to the corrected *trajectory point* $x(t_0 + \Delta t)$. Since all steps in the algorithm are time-reversible, the entire algorithm will be time-reversible. Furthermore, like LF, it only requires one force evaluation per step.

Since no future force information will be available at the time that the predictor step is taken, to be time-reversible the predictor must not use information from the past; it must be guided only by the current force, F_{t_0}. As discussed above, this is equivalent to using the LF algorithm. Furthermore, to obtain a time-reversible algorithm, the future and past predictor points $x_p(t_0 + \Delta t)$ and $x_p(t_0 - \Delta t)$ must enter the algorithm symmetrically; this leads to the modified LF predictor given in (21) below. The complete time-reversible algorithm is

THREE-POINT QUADRATIC SI ALGORITHM.
INPUT: the current and past trajectory points $x(t_0)$ and $x(t_0 - \Delta t)$, the current and past predictor points $x_p(t_0)$ and $x_p(t_0 - \Delta t)$, and the forces evaluated at these predictor points $F_{t_0} \equiv F[x_p(t_0)]$ and $F_{t_0-\Delta t} \equiv F[x_p(t_0 - \Delta t)]$.
OUTPUT: the values at $t_0 + \Delta t$ calculated by the following steps:

a. *Predictor* (modified LF),

$$x_p(t_0 + \Delta t) = 2x(t_0) - x_p(t_0 - \Delta t) + (\Delta t)^2 F_{t_0}. \quad (21)$$

b. *Compute interpolation parameter K.* Evaluate

$$F_{t_0+\Delta t} \equiv F[x_p(t_0 + \Delta t)]. \tag{22}$$

and calculate K using a time-reversal-invariant procedure (to be specified) involving $F_{t_0-\Delta t}$, F_{t_0}, and $F_{t_0+\Delta t}$.

c. *Corrector.* Compute $x(t + \Delta t)$ using (19).

3. A ONE-DIMENSIONAL IMPLEMENTATION OF THE QUADRATIC SI ALGORITHM

We first consider some simple one-dimensional examples to demonstrate how the quadratic SI algorithm works. For this $N = 1$ case K is a scalar. It is time-reversibly fixed by the requirement

$$K = -\Delta F/\Delta x, \tag{23}$$

where

$$\begin{aligned} \Delta x &\equiv x_p(t_0 + \Delta t) - x_p(t_0 - \Delta t) \\ \Delta F &\equiv F_{t_0+\Delta t} - F_{t_0-\Delta t}. \end{aligned} \tag{24}$$

The interpolation f defined by (16) automatically satisfies

$$f[x_p(t_0)] = F_{t_0}, \tag{25}$$

and with (23) it also satisfies

$$f[x_p(t_0 + \Delta t)] - f[x_p(t_0 - \Delta t)] = F_{t_0+\Delta t} - F_{t_0-\Delta t}. \tag{26}$$

In the special case where the potential is quadratic f will exactly match F at all points, but this is not true in general.

3.1. *Performance on the One-Dimensional Harmonic Oscillator*

The maximum time-step that can be used in MD simulations is generally determined by the period of the fastest oscillations in the system. Usually, these are approximately sinusoidal motions, so good performance on the harmonic oscillator is critical. We are primarily interested in comparing performance to LF since it is customarily used in MD applications. However, equal-cost comparisons with fourth-order RK were also performed.

Since, for the one-dimensional harmonic oscillator, the interpolated force f exactly equals the actual force F, in contrast with the LF and RK methods, the SI method is *exact for any* Δt. This is demonstrated in Fig. 1, where we compare the performance of SI and LF methods on the one-dimensional harmonic oscillator for two different size time-steps. In these examples we set $k = 1$ and $x_0 = 0$ in (10) and $\Delta t = \sqrt{2}/2$ (Fig. 1a) or $\Delta t = 1$ (Fig. 1b). (The initial conditions were $x(0) = 0$ and $v(0) = 1$.) As discussed above, these time-steps are representative of those used in practice for LF, since they are fairly close to its critical instability limit $\Delta t = 2$ determined by (14).

As expected, the exact harmonic trajectories and the SI trajectories are identical in both plots. As predicted by (14), the LF trajectory oscillates too rapidly. For $\Delta t = \sqrt{2}/2$, $\omega_{\mathrm{LF}} = 1.022\omega$, and the LF trajectory is significantly out-of-phase within a few cycles. For $\Delta t = 1$, $\omega_{\mathrm{LF}} = 1.047\omega$, and the LF trajectory goes out-of-phase even more rapidly.

Because fourth-order RK makes four force evaluations per time-step, for an equal-cost comparison it was tested with $\Delta t = 2\sqrt{2}$. However, it is unstable under these conditions (not plotted in Fig. 1). With $\Delta t = \sqrt{2}$ (i.e., double cost) RK is stable but becomes highly inaccurate after three cycles.

3.2. *Performance on a One-Dimensional Anharmonic Oscillator*

We next compare the performance of SI, LF, and RK on the anharmonic oscillator defined by Newton's equation with a nonlinear force:

$$\frac{d^2x}{dt^2} = F \equiv -kx - \alpha x^3. \tag{27}$$

This problem poses a greater challenge for SI since the quadratic interpolation is not exact and, like LF and RK, quadratic SI cannot compute the exact solution. In Fig. 2a we compare the equal-cost performance of the SI and LF methods with $k = \alpha = 1$ and $\Delta t = \sqrt{2}/2$ (with initial conditions $x(0) = 0$ and $v(0) = 1$). Visual inspection indicates that the performance of SI is much better than that of LF. A quantitative measure of their relative accuracies is provided by the fractional errors in their dominant frequencies (determined from the numerical Fourier transform) relative to the dominant frequency ω of the exact solution.[5] We obtain $\Delta\omega_{\mathrm{LF}}/\omega \approx 0.39$ and $\Delta\omega_{\mathrm{SI}}/\omega \approx 0.026$, indicating that SI is about 15 times more accurate than LF for this problem. The equal-cost RK trajectory is unstable.

To obtain a measure of relative efficiency, the same problem was analyzed using a different time-step for each method chosen so that each method had roughly the same dominant frequency error $\Delta\omega/\omega \approx 0.02$. This occurred

[5] The exact solution can be expressed in terms of elliptic functions [19], but for simplicity was computed to very high accuracy using the RK method with the very small (and costly) time-step $\Delta t = 0.005$.

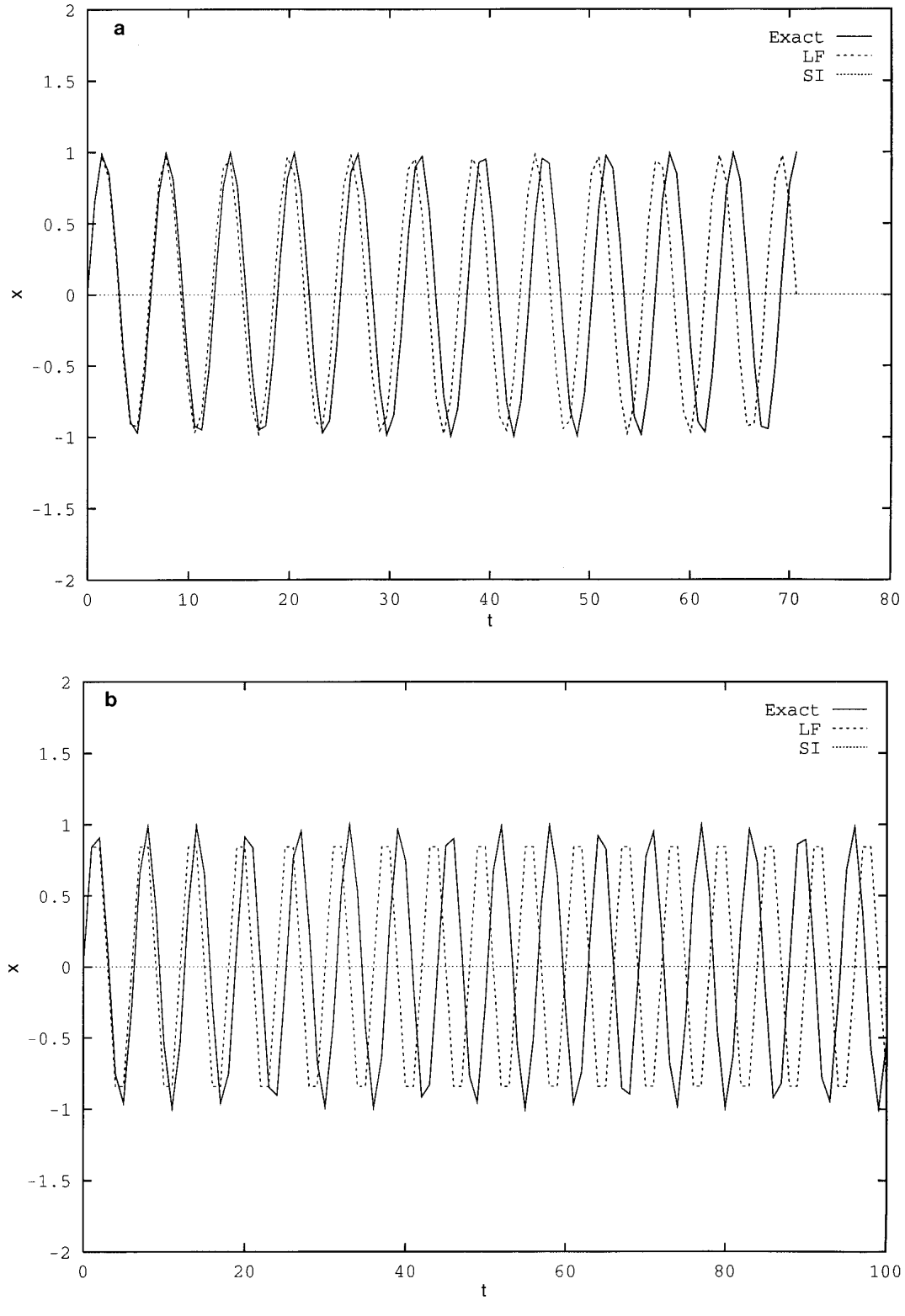

FIG. 1. Comparison of spatial interpolation (SI) and leap-frog (LF) numerical trajectories for the one-dimensional harmonic oscillator with $F(x) = -x$: (a) The numerically calculated trajectories are compared with the exact trajectory for $\Delta t = \sqrt{2}/2$ with initial conditions $x(0) = 0$ and $v(0) = 1$. (b) As in (a) except that $\Delta t = 1$. In both panels, the SI trajectory exactly overlays the exact solution and cannot be distinguished from it.

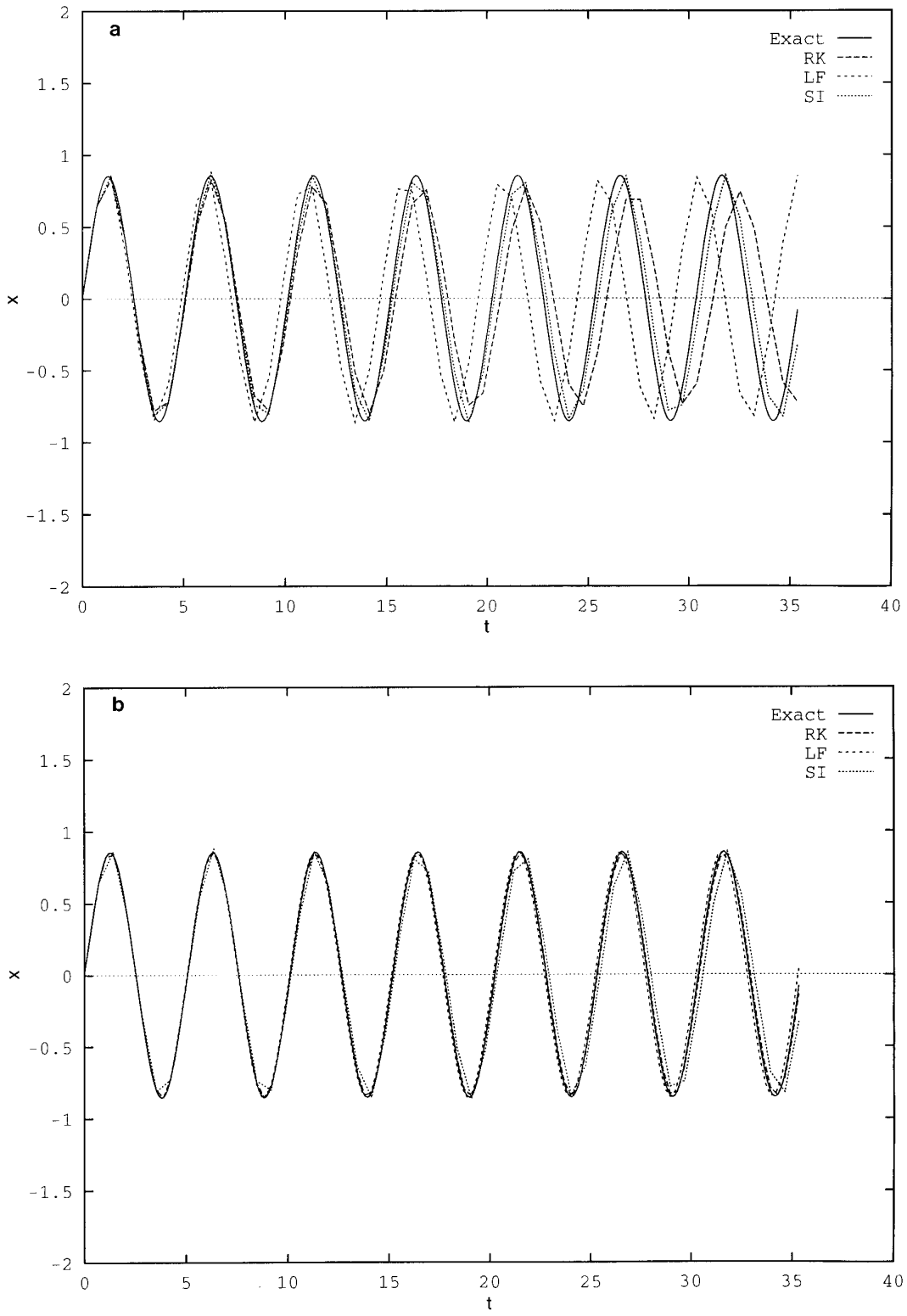

FIG. 2. Comparison of SI, LF, and RK numerical trajectories for a one-dimensional anharmonic oscillator with $F(x) = -x - x^3$: (a) Equal-cost comparison of the numerically calculated trajectories with the exact trajectory for $x(0) = 0$ and $v(0) = 1$. The time-step used for each method is proportional to its number n of force evaluations per time-step ($n_{\mathrm{LF}} = n_{\mathrm{SI}} = 1$, $n_{\mathrm{RK}} = 4$) so that each trajectory calculation required the same total number of force evaluations ($\Delta t/n = \sqrt{2}/2$). The RK trajectory is not shown since it diverges. (b) Equal-accuracy comparison with the same initial conditions. As in (a), but the time-step for each method was selected so that each method achieved approximately equal accuracy, i.e., having a dominant frequency error of ~0.02 ($(\Delta t/n)_{\mathrm{SI}} = \sqrt{2}/2$, $(\Delta t/n)_{\mathrm{LF}} = \sqrt{2}/6$, $(\Delta t/n)_{\mathrm{RK}} = \sqrt{2}/8$). Quantitative results are listed in Table I.

when $\Delta t_{SI} = \sqrt{2}/2$, $\Delta t_{LF} = \sqrt{2}/6$, and $\Delta t/n_{RK} = \sqrt{2}/2$. The trajectories are shown in Fig. 2b. Allowing for the fact that RK uses four force evaluations per time-step, we conclude that SI is about three times more efficient than LF and about four times more efficient than RK for this problem.

4. A MULTIDIMENSIONAL IMPLEMENTATION OF THE SI METHOD

In multiple dimensions K is a symmetric tensor. It is partially fixed by the multidimensional generalization of (23),

$$K \cdot \Delta x = \Delta x \cdot K = -\Delta F, \tag{28}$$

where Δx and ΔF are still defined by (24). Equation (28) is solved by

$$K_{\parallel} = -[\Delta F \otimes \varepsilon + \varepsilon \otimes \Delta F - (\Delta F \cdot \varepsilon)\varepsilon \otimes \varepsilon]/|\Delta x|, \tag{29}$$

where

$$\varepsilon \equiv \Delta x/|\Delta x|. \tag{30}$$

The components of $K_{\parallel}$ that are orthogonal to both Δx and ΔF are not determined by (28) and the symmetricity requirement; (29) sets them to zero.

As long as $x_p(t_0)$, $x_p(t_0 - \Delta t)$, and $x_p(t_0 + \Delta t)$ are not colinear, the two independent force differences between these points will constrain the elements of K in two distinct directions. Some, but not all, of this information is included in (28). We would like to use a second constraint to improve our approximation for K, but we cannot simply impose a second condition like

$$K \cdot \Delta x_- = \Delta x_- \cdot K = \Delta F_- \quad \text{(not used)}, \tag{31}$$

where

$$\Delta x_- \equiv x_p(t_0) - x_p(t_0 - \Delta t) \tag{32}$$

$$\Delta F_- \equiv F_{t_0} - F_{t_0 - \Delta t}, \tag{33}$$

because it cannot, in general, be simultaneously satisfied along with (28). A number of alternative conditions could be applied. A simple, although possibly not optimal, approach is to impose the second condition only in the subspace orthogonal to ε. Defining the orthogonal space projection operator

$$\mathscr{P}_{\perp} \equiv 1 - \varepsilon \otimes \varepsilon, \tag{34}$$

we impose as a second constraint,

$$\mathscr{P}_{\perp} \cdot K \cdot \Delta x_- = \Delta x_- \cdot K \cdot \mathscr{P}_{\perp} = -\mathscr{P}_{\perp} \cdot \Delta F_-. \tag{35}$$

Taking

$$K = K_{\parallel} + K_{\perp}, \tag{36}$$

where

$$K_{\perp} \cdot \varepsilon = \varepsilon \cdot K_{\perp} = 0, \tag{37}$$

and using (35), we get the condition on $K_{\perp}$

$$K_{\perp} \cdot \Delta x_- = \Delta x_- \cdot K_{\perp} = -\Delta F_{\perp}, \tag{38}$$

where

$$\Delta F_{\perp} \equiv \mathscr{P}_{\perp} \cdot [\Delta F_- - \Delta F(\varepsilon \cdot \Delta x_-)/|\Delta x|]. \tag{39}$$

This has the solution

$$K_{\perp} = -[\Delta F_{\perp} \otimes \varepsilon_{\perp} + \varepsilon_{\perp} \otimes \Delta F_{\perp} - (\Delta F_{\perp} \cdot \varepsilon_{\perp})\varepsilon_{\perp} \otimes \varepsilon_{\perp}]/|\Delta x_{\perp}|, \tag{40}$$

where

$$\Delta x_{\perp} \equiv \mathscr{P}_{\perp} \cdot \Delta x_- \tag{41}$$

$$\varepsilon_{\perp} \equiv \mathscr{P}_{\perp} \cdot \Delta x_-/|\mathscr{P}_{\perp} \cdot \Delta x_-|. \tag{42}$$

As with (29), (40) sets the components of $K_{\perp}$ that are not fixed by (38) and the symmetricity requirement to zero.

Using (28) it is easy to show that definition (40) of $K_{\perp}$ is invariant under the substitutions

$$\Delta x_- \to \Delta x_+ \equiv x_p(t_0) - x_p(t_0 + \Delta t)$$

$$\Delta F_- \to \Delta F_+ \equiv F_{t_0} - F_{t_0 + \Delta t}.$$

Thus $K_{\perp}$, as well as K calculated by (36), is time-reversal-invariant. This invariance can be made explicit by rewriting (39) as

$$\Delta F_{\perp} \equiv \mathscr{P}_{\perp} \cdot [\Delta F_a - \Delta F(\varepsilon \cdot \Delta x_a)/|\Delta x|], \tag{43}$$

where

$$\Delta x_a \equiv (\Delta x_- + \Delta x_+)/2$$

$$\Delta F_a \equiv (\Delta F_- + \Delta F_+)/2.$$

However, (39) is simpler for computation.

In two dimensions, if U is quadratic the interpolated force f specified by (16) with (29), (36), and (40) will exactly

match F everywhere, just as in the one-dimensional quadratic case. Thus, SI will also give the exact solution for any time-step for the two-dimensional harmonic oscillator. However, when the dimensionality is three or greater, $F_{t_0-\Delta t}$, F_{t_0}, and $F_{t_0+\Delta t}$ will not provide sufficient information to completely determine all the elements of K, even when the potential is quadratic. Thus, we will not exactly solve the multidimensional ($N \geq 3$) harmonic oscillator. However, we do expect that using even the approximate K specified by the above interpolation will provide improved accuracy relative to LF both for harmonic and anharmonic problems.

If the matrix operations required to evaluate K and $\xi(\sqrt{K}\ \Delta t)$ (see (20)) had to be performed in the full N-dimensional space, the SI method would have $O(N^3)$ complexity and would not be useful. The key to its practical utility is that all these operations can be performed in a restricted subspace of small dimensionality. Note that K operates only within the four-dimensional subspace spanned by ΔF, $\Delta F_\perp$, ε, and $\varepsilon_\perp$, so all K-dependent operations can be restricted to this subspace. ε and $\varepsilon_\perp$ already provide two orthonormal basis vectors for the subspace; two additional basis vectors can be obtained from ΔF and $\Delta F_\perp$ by Gramm–Schmidt orthogonalization with only $O(N)$ complexity. Then the cosine and pseudo-inverse needed to evaluate ξ using (20) can be computed within the eigenvector basis of K at negligible $O(1)$ cost. Similarly, the matrix–vector multiplications in the SI step equation (19) can be efficiently calculated by first projecting the relevant vectors (i.e., ΔF_{t_0} and $x(t_0) - x_p(t_0)$) into the four-dimensional subspace at cost $O(N)$. Thus, like LF, the overall complexity of the quadratic SI algorithm ignoring the force evaluation is $O(N)$, and the $O(N^2)$ complexity of evaluating F still dominates the cost.

4.1. *Performance on a Multidimensional Harmonic Oscillator*

The performance of the quadratic SI method using the three-point interpolation for K was tested using the 10-dimensional harmonic oscillator specified by (1) with a force term $F(x) = -K(x - x_0)$, where k was a randomly chosen symmetric positive-definite 10-dimensional matrix and, without loss of generality, $x_0 = 0$. The eigenvalues of K spanned an ~100-fold range from 0.9180 down to 0.0096.[6] The analytic solution of this problem is the multidimensional generalization of (11) and can be computed in the K eigenvector basis.

The performances of SI, LF, and RK with randomly chosen initial conditions[7] were compared on an equal-cost basis using $\Delta t/n = 1$ (where n is the number of function evaluations per time-step). For graphical comparison (Fig. 3), the computed trajectories for the components of x corresponding to the smallest (0.0096; part a) intermediate (0.4798; part b), and largest (0.9180; part c) eigenvalues of K (corresponding to oscillation frequencies $\omega = \sqrt{K}$) are plotted in comparison with the exact trajectory components. In all of these cases, the SI trajectory matches the exact solution almost perfectly (and can thus be barely distinguished from the exact trajectories in the figures). The performances of LF and RK deteriorate as the eigenvalues increase. For the smallest eigenvalue all three methods are very accurate. For the intermediate size eigenvalue, LF goes out of phase significantly after seven cycles and the equal-cost RK trajectory yields highly inaccurate results. For the largest eigenvalue, LF goes quickly out of phase and the corresponding RK trajectory is unstable.

The accuracy of the SI and LF methods were quantitatively assessed by computing the errors in the dominant frequencies of their trajectories (Table I). (The RK trajectory was too inaccurate for such analysis). This shows that SI is more or less equivalent to LF for the smallest eigenvalue, ~16 times more accurate than LF for the intermediate eigenvalue, and >20 times more accurate for the largest eigenvalue. Since the overall performance (and maximum usable time-step) is determined by the largest eigenvalue, the advantage of the SI method is well established.

To compare efficiencies, we determined the time-step for each method that would give the same error ($(\Delta\omega/\omega) \sim 0.03$) for the largest eigenvalue mode. The results (Table I) indicate that, in this problem, SI is approximately 4 times more efficient than LF and 5 times more efficient than RK.

4.2. *Performance on a Multidimensional Anharmonic Oscillator*

A multidimensional nonlinear test was performed using the anharmonic oscillator defined by

$$F(x) = -K \cdot x - \alpha G(x), \tag{44}$$

where $G(x)$ is the nonlinear part of the force term. The ith component of $G_i(x)$ was chosen as

$$G_i(x) = x_i^3 |x_{1+i \bmod N}|, \quad i = 1 \cdots N, \tag{45}$$

where N is the dimensionality of the problem. Since G is not diagonal, all the components are mixed. We used $\alpha = 0.3$ and the same 10-dimensional ($N = 10$) matrix K and initial conditions[7] as were used in Section 4.1.

[6] The eigenvalues of K were 0.9180, 0.7990, 0.7934, 0.7559, 0.6465, 0.6449, 0.6391, 0.4798, 0.3743, and 0.0096.

[7] The components of $x(0)$ and $v(0)$ in the K eigenvector basis were (0.8837, 0.6670, 0.4908, 0.5314, 0.6501, 0.6986, 0.2733, 0.5744, 0.1614, 0.9761) and (0.3491, 0.0854, 0.1542, 0.4506, 0.0805, 0.3850, 0.4598, 0.4489, 0.0301, 0.4072), respectively.

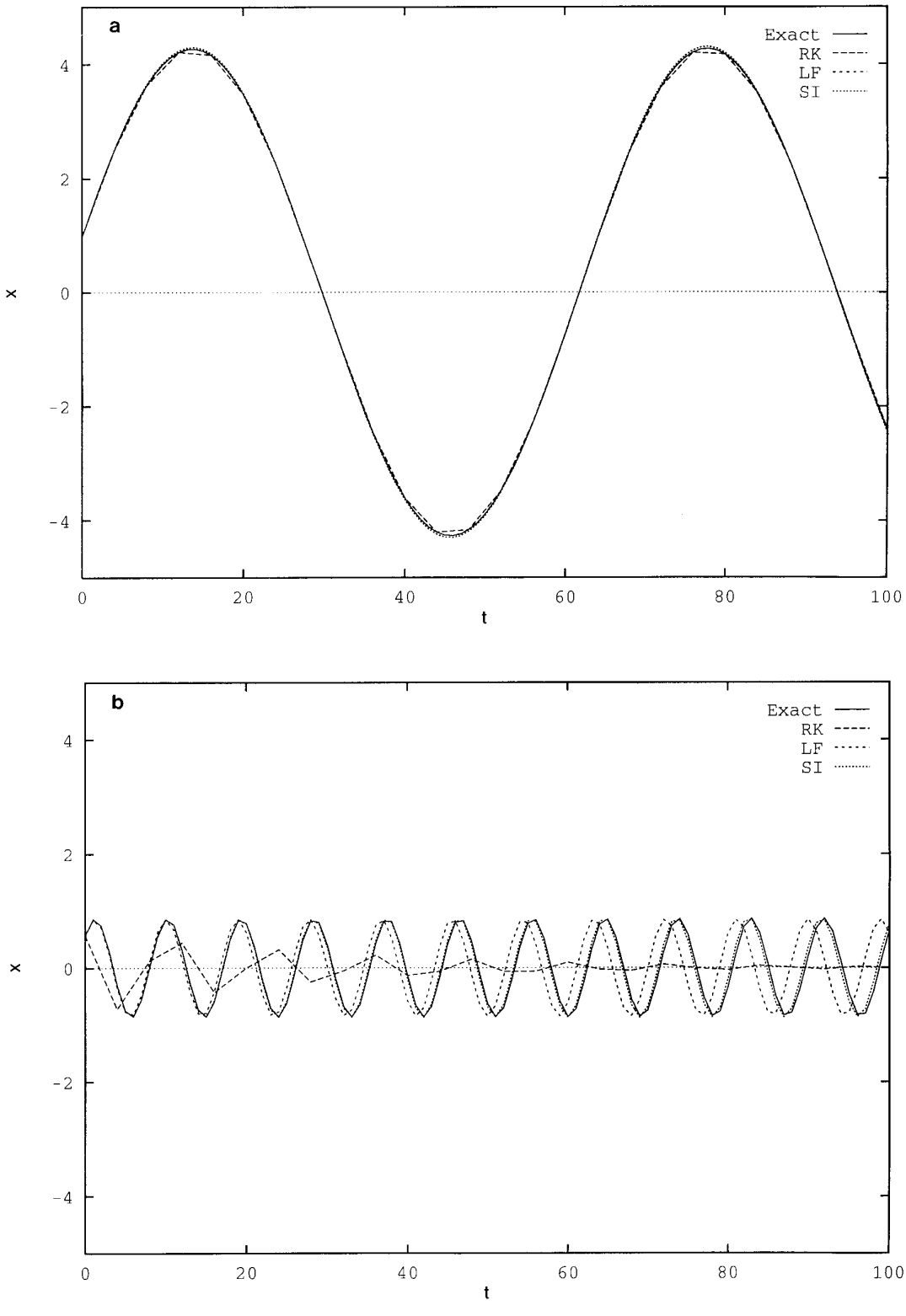

FIG. 3. Equal-cost comparison of SI, LF, and RK numerical trajectories for a 10-dimensional harmonic oscillator with $F(x) = -K \cdot x$. The eigenvalues of K were randomly selected positive values lying between 0 and 1. These and initial conditions are described in Section 4.1. The number of force evaluations per time interval was $\Delta t/n = 1$. SI, LF, RK, and exact solution trajectories for the (a) low ($\omega_i^2 = k_i = 0.0096$), (b) medium ($k_i = 0.4798$), and (c) high ($k_i = 0.9180$) frequency components of the $x(t)$ trajectory are plotted. The SI trajectory is almost identical with and can barely be distinguished from the exact trajectory. Quantitative results are listed in Table I.

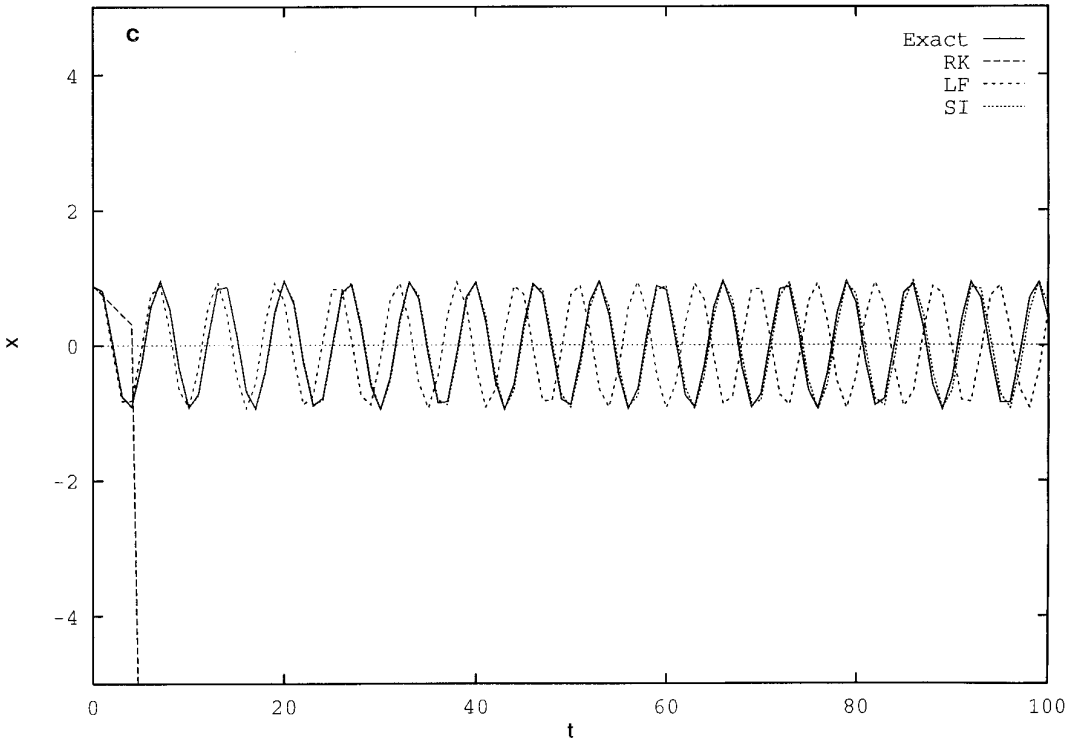

FIG. 3—*Continued*

For an equal-cost comparison we computed trajectories with $\Delta t/n = 1$. The SI, LF, and RK trajectory components corresponding to the 0.0096, 0.4798, and 0.9180 eigenvalues of K are compared with the exact trajectory components[8] in Fig. 4. (The equal-cost RK trajectory is not shown since it is unstable and blows up quickly.) LF initially displays acceptable performance (although it is clearly inferior to SI) but becomes unstable for $t > 80$ because of accumulating errors in the 0.0096 eigenvalue mode (part a). (The instability occurs first in this mode because of the greater relative importance of the anharmonic term when K is small.) In contrast, the SI trajectory is stable over the entire interval.

To compare efficiencies, we determined the time-step for each method that would give the same error ($(\Delta\omega/\omega) \sim 0.188$) for largest eigenvalue mode. The results (Table I) indicate that, in this problem, SI is approximately 4.5 times more efficient than LF and 8 times more efficient than RK.

5. CONCLUSION

General-purpose ODE integrators ignore the fact that the force in Newton's equation (for conservative systems) is the spatial gradient of a scalar potential. We have described a new class of spatial interpolation algorithms that exploit this property to improve integrator performance by interpolating the force in space rather than in time and we have demonstrated the superior performance of a simple algorithm of this type.

While the performance of standard ODE integrators can be formally described in terms of their order of accuracy in Δt, this is not possible for SI algorithms. Their accuracy depends both on Δt and on the spatial fluctuation of F in a complicated manner. Thus, we have empirically compared our algorithm with the LF and fourth-order RK methods on a few one- and multi-dimensional test problems. The results for trajectories of moderate length demonstrate two related points:

1. The SI algorithm is more accurate than LF for equal cost; SI was over an order-of-magnitude more accurate than LF in some of the test cases when equal time-steps were used.

2. The SI algorithm is more efficient than LF for equal accuracy; in the tested cases, SI required about 3–4 fewer force evaluations than LF for equal accuracy.

Performance was even better when compared against the fourth-order RK method.

However, while SI generally outperformed and was more stable than LF, we have found that it can be less

[8] The "exact" trajectory was computed using RK with $\Delta t = 0.005$.

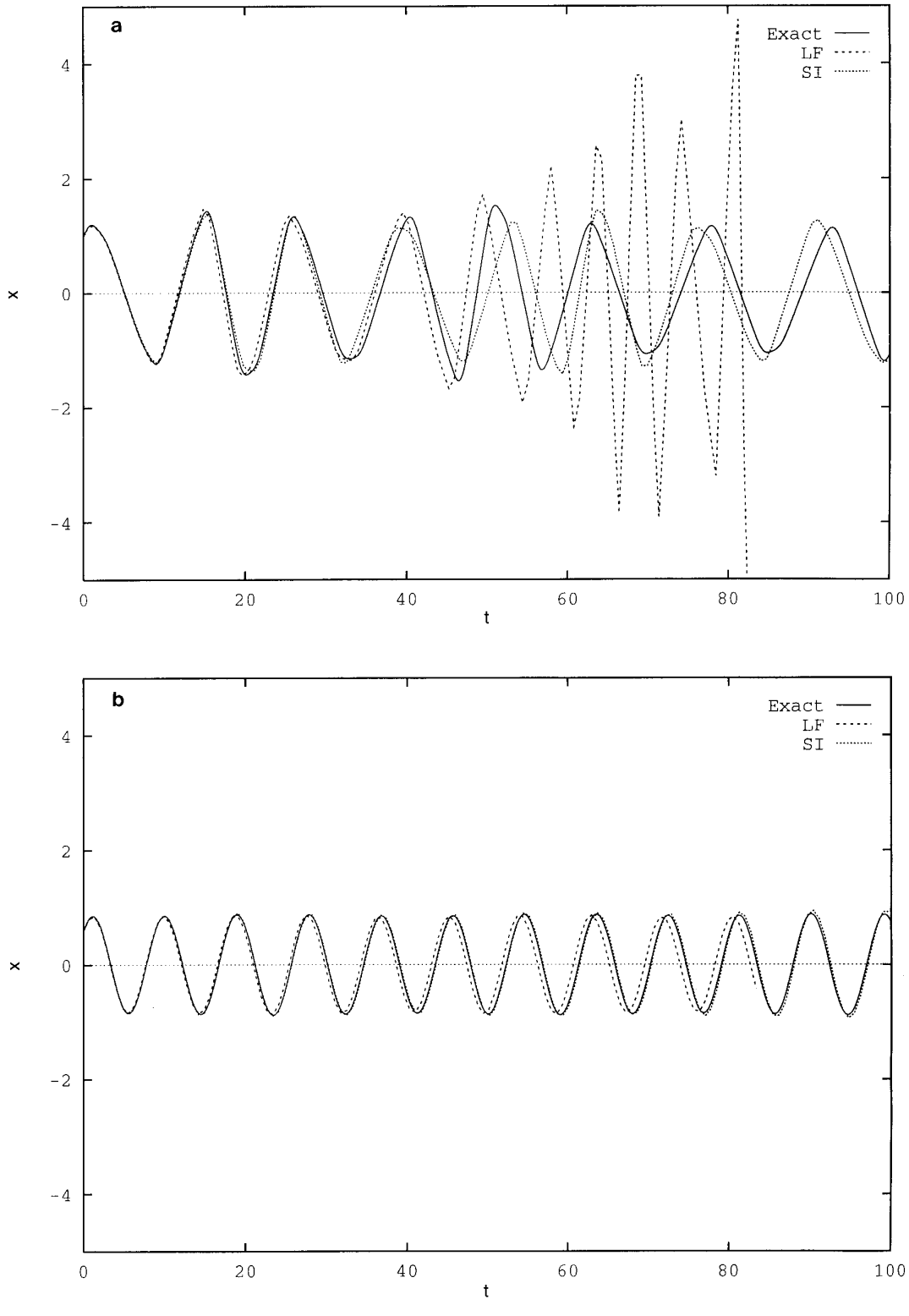

FIG. 4. Equal-cost comparison of SI and LF numerical trajectories for a 10-dimensional anharmonic oscillator with $F(x) = -(K \cdot x)_i - 0.3x_i^3|x_{1+i \bmod N}|$. The matrix K, time steps, and initial conditions were the same as in Fig. 3. SI, LF, and exact solution trajectories for the (a) low ($\omega_i^2 = k_i = 0.0096$), (b) medium ($k_i = 0.4798$), and (c) high ($k_i = 0.9180$) frequency components of the $x(t)$ trajectory are plotted. The RK trajectories are not shown because they are unstable. The LF trajectory is unstable for $t > 83$ so the plots of all its components are terminated. Quantitative results are listed in Table I.

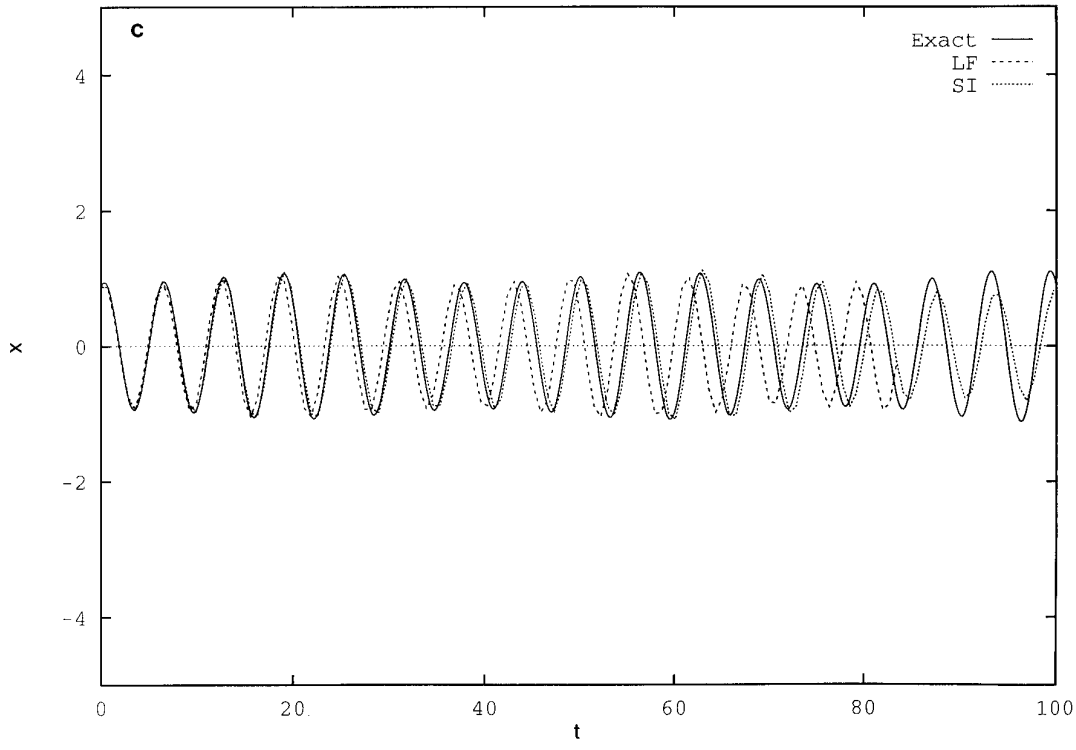

FIG. 4—*Continued*

stable than LF in some situations. For example, although SI outperforms LF over the time domain considered in the multidimensional harmonic oscillator example, after many hundreds of steps it becomes unstable while LF remains stable. This occurs because the inherent nonlinearity in F of the SI step (19) "mixes" the motions in the different coordinates to generate trajectories which curve within a single step. This is in contrast with the linear LF step (6) which treats motion in each component independently. This independence is appropriate for simple problems like the harmonic oscillator but it may be a disadvantage for more complex problems, where the different components are coupled. In most cases the nonlinearity of the SI step is helpful and permits it to model the curved trajectories with higher accuracy. However, this can cause slowly accumulating errors in energy transfers between dynamic modes of the system. This error is probably not important in realistic MD problems where velocity-rescaling or other algorithmic modifications to ensure long-term energy conservation are already used to cope with errors induced by other approximations. It may be possible to control this error simply by resetting the algorithm every few hundred steps, but more investigation is needed.

Our primary goal here has been to evaluate the utility of the SI principle in basic test cases. The implementation we have presented is simple but not necessarily optimal, and there are many ways to improve the SI algorithm before proceeding to performance testing in large-scale problems. In particular, more sophisticated methods for interpolating the force and potential could be applied. For example, the three-point interpolation described in Section 4 only utilizes knowledge of the force $F(x)$, not of the potential $U(x)$. However, $U(x)$ can be evaluated at little additional cost and can provide additional information that is useful in maintaining long-term energy conservation. Furthermore, it is not necessary to restrict SI to the use of quadratic potential interpolations. While the simple analytic form (19) of the SI time-step in a quadratic potential simplifies computation, the SI steps for more complex interpolating potentials could be calculated numerically. The cost for this would be minimal since, as discussed in Section 4, all motion within a single SI step occurs within a subspace of small dimensionality. A single step could be subdivided into many smaller mini-steps, and the force at each mini-step could be evaluated from the interpolating potential. The cost of projecting the relevant x, x_p, and $F(x_p)$ vectors into the subspace is only $O(N)$ and, because all subsequent mini-step force evaluations would occur within the small-dimensionality subspace, the mini-steps could be computed in $O(1)$. We believe that further developments along these lines can lead to even greater enhancement in performance.

TABLE I

Fractional Errors ($\Delta\omega/\omega$) in the Dominant Frequencies of the Numerical Trajectories Computed by the Spatial Interpolation (SI), Leap-Frog (LF), and Fourth-Order Runge–Kutta (RK) Methods

Problem	$\Delta t/n$	k	SI	LF	RK
1D HO	$\sqrt{2}/2$	1	0	0.022	Unstable
	1	1	0	0.047	Unstable
1D Anharmonic O	$\sqrt{2}/2$	1	0.026	0.39	Unstable
	$\sqrt{2}/6$	1		~0.02	
	$\sqrt{2}/8$	1			~0.02
10D HO	1	0.0096	0.021	0.024	0.020
	1	0.4798	0.020	0.337	Decays
	1	0.9180	0.030		
	0.25	0.9180		~0.030	
	0.1875	0.9180			~0.030
10D Anharmonic O	$\sqrt{2}/2$	0.0096	0.028	Unstable	Unstable
	$\sqrt{2}/2$	0.4798	0.035	Unstable	Unstable
	$\sqrt{2}/2$	0.9180	0.188		
	$\sqrt{2}/9$	0.9180		~0.188	
	$\sqrt{2}/16$	0.9180			~0.188

Note. Dominant frequencies were determined by Fourier analysis from the numerical trajectories shown in Figs. 1–4. Errors are expressed as fractions relative to the dominant frequency of the exact solution trajectory. $\Delta t/n$ is the time-step divided by the number of force evaluations per step ($n_{SI} = n_{LF} = 1$; $n_{RK} = 4$). This provides an inverse measure of the cost.

Motivated by the need to extend the computed time domain in MD simulations, many other methods for improving the efficiency of integrating Newton's equations have been developed in recent years [9]. For example, the widely used SHAKE and RATTLE algorithms [6, 7] constrain the most rapid motions (e.g., those of covalent hydrogen bonds), thereby permitting larger time-steps to be used when integrating the remaining variables. Berne and coworkers are developing a different approach that allows degrees-of-freedom and force terms to be separated into ''slow'' and ''fast'' classes [12, 13]. Since the slowly varying force terms need not be evaluated at each step, efficiency is increased. A similar philosophy is employed in the multiple time-step method proposed by Grubmüller *et al.* [10]. Schlick's group has developed an alternate method for extending the time-step in which the rapidly varying motions are damped out by including friction terms into Newton's equation [14–16]. This converts Newton's equation into the Langevin equation and allows the more stable backwards-Euler integrator to be used with a longer time-step. In addition, there are many methods for approximating F to reduce the cost of its evaluation (e.g., see [1]). In all of these methods the conservative part of F is a function of x so the SI principle is still applicable. Thus, it may be possible to combine their benefits with those of SI in hybrid algorithms.

ACKNOWLEDGMENTS

We thank Dr. Alex Ulitsky for many helpful conversations and his help in developing the time-reversible three-point interpolation. This work was supported by NIH Grant GM48874, AFOSR Grant F49620, the Cornell Theory Center, and by the Technion V.P.R. fund.

REFERENCES

1. L. Greengard, *Science* **265,** 909 (1994).
2. L. K. Nicholson, T. Yamazaki, D. A. Torchia, S. Grzesiek, A. Bax, S. J. Stahl, J. D. Kaufman, P. T. Wingfield, P. Y. S. Lam, P. K. Jadhav, C. N. Hodge, P. J. Domaille, and C.-H. Chang, *Struct. Biol.* **2,** 274 (1995).
3. B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, *J. Comput. Chem.* **4,** 187 (1983).
4. S. P. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta Jr., and P. J. Weiner, *J. Am. Chem. Soc.* **106,** 765 (1984).
5. J. A. McCammon and S. C. Harvey, *Dynamics of Proteins and Nucleic Acids* (Cambridge Univ. Press, New York, 1989).
6. J. P. Ryckaert, G. Ciccotti, and H. J. C. Berendsen, *J. Comput. Phys.* **23,** 327 (1977).
7. H. C. Andersen, *J. Comput. Phys.* **52,** 24 (1983).
8. E. Barth, K. Kuczera, B. Leimkuhler, and R. D. Skeel, *J. Comput. Chem.* **16,** 1192 (1995).
9. R. Elber, *Curr. Opin. Struct. Biol.* **6,** 232 (1996).
10. H. Grubmüller, H. Heller, A. Windemuth, and K. Schulten, *Mol. Simul.* **6,** 121 (1991).
11. G. D. Venneri and W. G. Hoover, *J. Comput. Phys.* **73,** 468 (1987).
12. M. Tuckerman, B. J. Berne, and G. J. Martyna, *J. Chem. Phys.* **97,** 1990 (1992).
13. R. Zhou and B. J. Berne, *J. Chem. Phys.* **104,** 9444 (1995).
14. C. S. Peskin and T. Schlick, *Commun. Pure Appl. Math.* **42,** 10001 (1989).
15. T. Schlick, *Comput. Chem.* **15,** 251 (1991).
16. P. Derreumaxu and T. Schlick, *Proteins* **21,** 281 (1995).
17. H. Schreiber and O. Steinhauser, *Chem. Phys.* **168,** 75 (1992).
18. M. Saito, *J. Chem. Phys.* **101,** 4055 (1994).
19. T. E. Stern, *Theory of Nonlinear Networks and Systems* (Addison–Wesley, Reading, MA, 1965), 386.