Fast evaluation of polarizable forces

Wei Wang[a)]
Beckman Institute, University of Illinois, Urbana, Illinois 61801

Robert D. Skeel[b)]
Department of Computer Science, Purdue University, West Lafayette, Indiana, 47907-2066

(Received 8 July 2005; accepted 16 August 2005; published online 26 October 2005)

Polarizability is considered to be the single most significant development in the next generation of force fields for biomolecular simulations. However, the self-consistent computation of induced atomic dipoles in a polarizable force field is expensive due to the cost of solving a large dense linear system at each step of a simulation. This article introduces methods that reduce the cost of computing the electrostatic energy and force of a polarizable model from about 7.5 times the cost of computing those of a nonpolarizable model to less than twice the cost. This is probably sufficient for the routine use of polarizable forces in biomolecular simulations. The reduction in computing time is achieved by an efficient implementation of the particle-mesh Ewald method, an accurate and robust predictor based on least-squares fitting, and non-stationary iterative methods whose fast convergence is accelerated by a simple preconditioner. Furthermore, with these methods, the self-consistent approach with a larger timestep is shown to be faster than the extended Lagrangian approach. The use of dipole moments from previous timesteps to calculate an accurate initial guess for iterative methods leads to an energy drift, which can be made acceptably small. The use of a zero initial guess does not lead to perceptible energy drift if a reasonably strict convergence criterion for the iteration is imposed. © *2005 American Institute of Physics*. [DOI: 10.1063/1.2056544]

I. SUMMARY

Polarization refers to the electron density redistribution due to the ambient electric field. Current generation nonpolarizable force fields for biomolecular simulations have serious limitations because polarization is treated only in an average sense by the parameterizations.[1,2] The treatment can neither reflect the dependency of the electron density on atomic positions, nor can it respond dynamically to different environments, which vary from almost nonconductive inside protein cavities to very conductive on protein-water interfaces. The explicit inclusion of polarization can significantly improve a force field's: (i) *Accuracy*, when being compared to quantum computation or experimental results,[3] and (ii) *transferability*, when applying the same force field to a wide range of temperatures and pressures.[4] Much research has shown promising results for polarizable force fields.[5–9] Polarizable models have the prospect to enable accurate computation of the binding energies of proteins and ligands in drug design.[1] What is more, polarizability is indispensable for studies of interfaces[10] and solvation processes.[11] However, proposed models and algorithms for polarizability incur a high computational cost and/or employ dubious approximations. As a result, few simulations actually incorporate the effects of induced polarization. Presented in this article is a self-consistent algorithm for modeling induced dipoles that is only twice the cost of simulating a fixed-charge model and is practically free of numerical artifacts.

Several reviews[1,12–14] survey polarizable force field modeling principles and their computational costs.

A. Polarization models

Two principal types of polarization models are point dipole models and fluctuating charge models.[12] A variation of the first of these is the shell (Drude) model, and similar to the second are charge-flow polarizabilities.[15]

A point dipole model represents the charge distribution of an atom by a charge and an induced dipole. In such a model, the pairwise potential energy between atoms at $\vec{r}_i$ and $\vec{r}_j$, with charges q_i and q_j and dipole moments $\vec{d}_i$ and $\vec{d}_j$, respectively, is

$$(q_i + \vec{d}_i \cdot \nabla_i)(q_j + \vec{d}_j \cdot \nabla_j)\frac{1}{|\vec{r}_i - \vec{r}_j|}. \tag{1}$$

In general, there is also a constant pre-factor, which is omitted here. The total electrostatic energy for an N-atom system can be represented in matrix form as

$$E^{\mathrm{el}}(\mathbf{r},\mathbf{d}) = \frac{1}{2}\mathbf{q}^T\mathbf{G}_0(\mathbf{r})\mathbf{q} + \mathbf{d}^T\mathbf{G}_1(\mathbf{r})\mathbf{q} + \frac{1}{2}\mathbf{d}^T\mathbf{G}_2(\mathbf{r})\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{D}_\alpha^{-1}\mathbf{d}, \tag{2}$$

where $\mathbf{q}$ is the collection of charges, $\mathbf{d}$ is the collection of dipole moments, $\mathbf{G}_0$, $\mathbf{G}_1$, and $\mathbf{G}_2$ are charge-charge, charge-dipole, and dipole-dipole interaction matrices defined

[a)]Present address: Goldman Sachs and Company, 85 Broad Street, Floor 25, New York, New York 10004. Electronic mail: weiwang1@ks.uiuc.edu
[b)]Present address: Department of Computer Sicence, Purdue University, 250 North University Street, West Lafayette, Indiana 47907-2066. Electronic mail: skeel@cs.purdue.edu

 © 2005 American Institute of Physics

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

through Eq. (1), $\mathbf{D}_\alpha$ is a block diagonal matrix incorporating the polarizability of each atom, and T denotes the transpose. The last term is the energy needed to create dipoles. Induced dipole moments assume values that minimize the energy (2):

$$\frac{\partial}{\partial \mathbf{d}} E^{\mathrm{el}}(\mathbf{r},\mathbf{d}) = 0 \Rightarrow (\mathbf{D}_\alpha^{-1} + \mathbf{G}_2)\mathbf{d} = -\mathbf{G}_1\mathbf{q}. \tag{3}$$

Once the dipole is known, the energy and force can be computed. It is necessary that $\mathbf{D}_\alpha^{-1}+\mathbf{G}_2$ be positive definite for the energy to have a minimum. However, $\mathbf{D}_\alpha^{-1}+\mathbf{G}_2$ can become indefinite when two dipoles are too close to each other—a shortcoming of the model known as a "polarization catastrophe."[16]

In shell models,[4,17] each polarizable atom is represented by a pair of point charges of opposite sign bound by a stiff spring. Shell models can be easily implemented in a computer program since the charge-charge and bonding interactions are already present in force fields. However, the number of charges is doubled and the computational cost is doubled at least.

Polarization can also be modeled by allowing the value of partial charges to change in response to the local electric field. Computationally speaking, the fluctuating charge model is probably the most efficient model: It does not have dipole interactions, and its number of charges remains the same as the number of atoms. However, the fluctuating charge model has limitations. For example, according to this model, water, a planar three-atom molecule, should not respond to the electric field perpendicular to the plane, although experiments show nearly isotropic polarizability for water.[12,18] To overcome this shortcoming, fluctuating charge models are combined with point dipole models.[19]

In this study, we choose to implement the point dipole model, since it has less modeling limitations and has been implemented in AMBER.[20]

B. Computational methods

Forces are computed in various contexts—deterministic dynamics, stochastic dynamics, Monte Carlo simulations, and energy minimization—and the context has important implications for polarizable force computation. Considered here is deterministic molecular dynamics (MD).

Two methods are widely used for polarizable force field computation:[12,18] The self-consistent method and the extended Lagrangian method. At each timestep, the self-consistent method overtly minimizes the electrostatic energy with respect to the polarizable degrees of freedom, which are induced dipoles in a point dipole model, auxiliary charge positions in a shell model, and charge values in a fluctuating charge model. This is generally very expensive. The extended Lagrangian method[21,22] treats the polarizable degrees of freedom as dynamic variables, which are assigned kinetic energies with fictitious masses. Their masses serve for computational convenience and do not have physical meaning. Starting from a configuration with the electrostatic potential minimized, the method simply does dynamics with a small timestep. Although it does not explicitly minimize the electrostatic energy at each timestep, it can keep the electrostatic energy close to its minimal values for a certain time. The length of the time depends on the coupling of the fictitious subsystem with the rest of the system: Weaker coupling leads to longer time.

The extended Lagrangian method is faster than the self-consistent method, but it has drawbacks:

(1) The fictitious mass must be small to reduce coupling between polarizable degrees of freedom and atomic coordinates so that a low temperature of the polarizable degrees of freedom is maintained.[23] This, in turn, requires a smaller integration timestep. For example, timesteps of 0.2 and 1 fs have to be used in Refs. 24 and 25, respectively.

(2) It introduces artifacts. First, the physical system's linear momentum is no longer conserved.[26] Second, since the polarizable degrees of freedom are, in fact, at a much lower temperature than that of the rest of the system, the system is in a metastable state.[27] The heat flow from other degrees of freedom to the polarizable degree of freedom is undesirable, yet unavoidable. In a recent study,[28] even though the timestep is limited to 0.75–1 fs, the system has to have a full energy minimization every 300 ps.

In this article, we choose the self-consistent approach because it is suitable for kinetic as well as thermodynamic calculations and because it is a standard for induced dipole calculation against which other more compromised approaches can be compared.

The self-consistent computation of point dipole models has been very expensive. In fact, the slow adoption of polarizable force fields is partly due to its much higher computation cost.[13] Compared with the charge-only models, Refs. 29 and 30 report, respectively, a nine- and eight-fold increase in the computational cost with the standard Ewald sum implementation, while Ref. 25 reports a more than six-fold increase with the particle-mesh Ewald[31] implementation. Review[12] says the computation of the Ewald sum of a point dipole model is, as "a widely used rule of thumb," four times more expensive than that of a charge-only model.

C. Outline

The fundamental problem is solving Eq. (3) for the dipole moments efficiently. This is addressed in Sec. IV. The problem is intimately related to two other topics in MD: Fast electrostatic solvers and dynamics. The first topic is treated in Sec. II and the second in Sec. V.

The slow $1/r$ decay of the charge-charge interaction makes the computation of the electrostatic interaction *the* most expensive task in biomolecular simulations. A simple cut-off treatment can have serious unphysical effects. On the other hand, the Ewald sum is considered a reliable way for describing the electrostatic interaction, although it could lead to bias in free energy computations[32] and to artificial stability if the cell size is too small.[33] For a given accuracy, direct implementation of the Ewald sum is $O(N^{3/2})$ at best. Fast electrostatic solvers, which have cost $O(N)$ or $O(N \log N)$, include the particle-mesh Ewald (PME) method,[31] particle-

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

TABLE I. Computational costs of reference and new algorithms in work units.

	Cost of computing dipoles (cost per iteration × iterations)	Cost of computing energy and force	Overall cost
Reference 25	$(\approx 1)\times 6$	1.4	7.5–7.8
New algorithm	0.34×2	1.28	1.94

particle particle-mesh (P^3M) method,[34] fast multipole method,[35] and multilevel summation method.[36,37] We choose the PME method because it is efficient and widely used in biomolecular simulators, such as CHARMM,[38] AMBER, and NAMD.[39]

Section II presents a matrix formulation and an implementation of the Ewald sum and the PME method for a point dipole model. The energy/force computation with the dipole moments given incurs only about one quarter extra cost compared to a charge-only model. The most significant extra cost of the implementation in Ref. 25 comes from solving the dipole Eq. (3) iteratively: One iteration is almost as expensive as one full energy/force evaluation for a charge-only model. The main contribution of this section is an algorithm that reduces the cost to one-third by storing selected intermediate values to avoid most re-computations.

Section III presents two water models and simulation details for tests of accuracy and efficiency of the methods.

Section IV presents new methods that reduce the number of iterations from six in Ref. 25 to two when solving the dipole equation to the same accuracy level. The improvement comes from: (i) A more accurate initial guess and (ii) a faster converging iteration. We first show that polynomial extrapolation of degree from four to six gives very accurate predictions (even though the underlying integrator is only second order accurate). However, the accuracy of polynomial extrapolation is sensitive to the degree selected. So, we propose and implement a new predictor based on a least-squares fitting of previous values of the dipole moment. The new predictor is as accurate or more accurate than polynomial extrapolation, and it is robust, because the prediction quality does not degrade if too many previous values are used. To accelerate convergence for the iteration process, we propose and implement the Chebyshev semi-iterative method[40] and a modified conjugate gradient (CG) method. A disadvantage of the standard CG implementation is that two matrix-vector multiplications are needed to get the first update to the solution. We make use of the extra matrix-vector multiplication by a suboptimal last step so that CG becomes competitive with the Chebyshev method. The performance of the two iterative methods depends on the condition number of the matrix in the linear system being solved. To reduce the condition number, a simple preconditioner is constructed from a local approximation to the dipole-dipole interaction matrix and a polynomial approximation to the inverse of the local approximation.[41] The comparison of the computational cost between a polarizable point-dipole model and a charge-only model shows that our implementation incurs less than 100% extra cost. Table I summarizes the computational costs of the new algorithm. Costs are expressed in work units where one work unit is the computation cost for a full energy/force evaluation of a charge-only model. Additionally, the new algorithm with a larger timestep, e.g., 2 fs, is faster than the extended Lagrangian method.

Section V discusses the effect of dipole moment quality on dynamics. Foremost, we discuss the energy drift problem of the self-consistent computation. For Hamiltonian dynamics, conservation of the Hamiltonian is strongly aided by symplectic integration. However, self-consistent computation compromises the symplecticness in two ways: (i) Inexact solution makes the force nonconservative, and (ii) prediction from previous values makes the inexact solution history dependent. By examining each effect separately, we find nonconservativeness alone does not cause significant energy drift if the computed dipole moments are reasonably accurate. But history dependence is more detrimental and for tolerable energy drift, it requires the computed dipole moments to be two orders of magnitude more accurate than what is suitable in MD simulations. If the self-consistent computation does not use history, it preserves volume in phase space.

Additional details omitted from this article are available in Ref. 42. Also presented there is a noniterative method that avoids the secular energy drift problem: The electrostatic energy is (re)defined through a polynomial approximation to the matrix inverse and the force is computed as the exact negative gradient of the energy.

D. Discussion

Following are recommendations for methods and parameters. For a short timestep ($\Delta t=1$ or 2 fs), use the least-squares predictor with 8 or more previous dipoles and require high accuracy [4 parts per million (ppm)] for the dipole solution (to keep the energy drift tolerable). If a longer timestep ($\Delta t>2$ fs) is used, e.g., in a multiple-time-stepping method, prediction helps very little to obtain an accurate initial guess. So use zero as an initial guess and declare convergence for a relatively low accuracy (400 ppm), which is good enough to compute the force with an accuracy suitable for MD simulations and maintain the energy at a constant level for long time simulations. For simulations of very long duration, use the noniterative method.

II. COMPUTATION OF THE EWALD SUM

Sections II A and II B present the matrix formulation of the Ewald sum of an induced point dipole model and the PME method, respectively.

A. Ewald sum for a point dipole model

With periodic boundary conditions, the system is replicated infinitely often to fill space. The simulation box is commonly a parallelepiped with edges given by linearly independent basis vectors $\vec{a}_1,\vec{a}_2,\vec{a}_3$ and with volume $V=\det(\vec{a}_1,\vec{a}_2,\vec{a}_3)$. For a system of atoms, each having a point charge and an induced point dipole, the electrostatic potential with periodic boundary conditions is given as

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

$$E^{\mathrm{el}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\vec{n}_r}{}'(q_i + \vec{d}_i\cdot\nabla_i)(q_j + \vec{d}_j\cdot\nabla_j)\frac{1}{|\vec{r}_i - \vec{r}_j + \vec{n}_r|} + E^{\mathrm{polar}}. \tag{4}$$

Here $\vec{n}_r$ is a lattice vector defined as $\vec{n}_r = n_1\vec{a}_1 + n_2\vec{a}_2 + n_3\vec{a}_3$ for integers n_1, n_2, and n_3. The prime on the summation over $\vec{n}_r$ means some terms might be excluded: if $j \in \chi(i)$, where $\chi(i)$ is the list of excluded atoms for atom i including i itself, the $\vec{n}_r = \vec{\nu}$ term is excluded where $\vec{\nu} = \vec{\nu}(\vec{r} - \vec{r}')$ is the lattice vector such that $|\vec{r} - \vec{r}' + \vec{\nu}|$ is minimal among all possible lattice vectors. The polarization energy is

$$E^{\mathrm{polar}} = \frac{1}{2}\sum_{i=1}^{N}\vec{d}_i^{\,T}\boldsymbol{\alpha}_i^{-1}\vec{d}_i, \tag{5}$$

where the polarizability $\boldsymbol{\alpha}_i$ of atom i is a 3×3 matrix, which is diagonal for a simple model.

Unfortunately, the infinite sum in Eq. (4) converges only conditionally, if the total charge is 0, and diverges otherwise. Reference 43 proves that for the standard summation order the result is the Ewald sum plus a surface term. However, most biomolecular simulations omit the surface term, and this practice is adopted here.

Before presenting the Ewald sum, define reciprocal lattice basis vectors $\vec{b}_1$, $\vec{b}_2$, and $\vec{b}_3$ so that for $\alpha, \beta = 1, 2, 3$, $\vec{a}_\alpha \cdot \vec{b}_\beta = \delta_{\alpha\beta}$, where $\delta_{\alpha\beta}$ is 1 if $\alpha = \beta$, and 0 otherwise. Also, define the following functions:

$$g^{\mathrm{dir}}(\vec{r}, \vec{r}') = \sum_{\vec{n}_r} \frac{\mathrm{erfc}(\beta|\vec{r} - \vec{r}' + \vec{n}_r|)}{|\vec{r} - \vec{r}' + \vec{n}_r|}, \tag{6}$$

$$g^{\mathrm{dirx}}(\vec{r}, \vec{r}') = \sum_{\vec{n}_r \neq \vec{\nu}} \frac{\mathrm{erfc}(\beta|\vec{r} - \vec{r}' + \vec{n}_r|)}{|\vec{r} - \vec{r}' + \vec{n}_r|} - \frac{\mathrm{erf}(\beta|\vec{r} - \vec{r}' + \vec{\nu}|)}{|\vec{r} - \vec{r}' + \vec{\nu}|}, \tag{7}$$

$$g^{\mathrm{rec}}(\vec{r}, \vec{r}') = \frac{1}{\pi V}\sum_{\vec{m} \neq 0} \times \frac{\exp(-\pi^2|\vec{m}|^2/\beta^2)}{|\vec{m}|^2}\exp(2\pi i\vec{m}\cdot(\vec{r} - \vec{r}')), \tag{8}$$

where $\mathrm{erf}(x) = 2\pi^{-1/2}\int_0^x \exp(-t^2)dt$, $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$, β is a positive adjustable parameter, and $\vec{m} = m_1\vec{b}_1 + m_2\vec{b}_2 + m_3\vec{b}_3$, for integers m_1, m_2, m_3. Here, $g^{\mathrm{dirx}}(\vec{r}, \vec{r})$ is taken to be the limiting value $\Sigma_{\vec{n}_r \neq \vec{\nu}}\mathrm{erfc}(\beta|\vec{n}_r|)/|\vec{n}_r| - 2\beta/\sqrt{\pi}$. Then,

$$E^{\mathrm{el}} = E^{\mathrm{dir}} + E^{\mathrm{rec}} + E^{\mathrm{polar}}, \tag{9}$$

where

$$E^{\mathrm{dir}} = \frac{1}{2}\sum_i \sum_{j \notin \chi(i)} (q_i + \vec{d}_i\cdot\nabla_i)(q_j + \vec{d}_j\cdot\nabla_j)g^{\mathrm{dir}}(\vec{r}_i, \vec{r}_j) + \frac{1}{2}\sum_i \sum_{j \in \chi(i)} (q_i + \vec{d}_i\cdot\nabla_i)(q_j + \vec{d}_j\cdot\nabla_j)g^{\mathrm{dirx}}(\vec{r}_i, \vec{r}_j), \tag{10}$$

$$E^{\mathrm{rec}} = \frac{1}{2}\sum_i \sum_j (q_i + \vec{d}_i\cdot\nabla_i)(q_j + \vec{d}_j\cdot\nabla_j)g^{\mathrm{rec}}(\vec{r}_i, \vec{r}_j). \tag{11}$$

Note that the self-energy ($j = i$) has been absorbed into the direct sum.

Next, we define the matrix form of Eqs. (9)–(11) and (5). Define the "direct sum" $\mathbf{G}$ matrices as

$$(\mathbf{G}_0^{\mathrm{dir}})_{ij} = g^{\mathrm{dirx}}(\vec{r}_i, \vec{r}_j), \quad (\mathbf{G}_1^{\mathrm{dir}})_{ij} = \nabla g^{\mathrm{dirx}}(\vec{r}_i, \vec{r}_j), \tag{12}$$

$$(\mathbf{G}_2^{\mathrm{dir}})_{ij} = \nabla(\nabla')^T g^{\mathrm{dirx}}(\vec{r}_i, \vec{r}_j)$$

for $j \in \chi(i)$ and

$$(\mathbf{G}_0^{\mathrm{dir}})_{ij} = g^{\mathrm{dir}}(\vec{r}_i, \vec{r}_j), \quad (\mathbf{G}_1^{\mathrm{dir}})_{ij} = \nabla g^{\mathrm{dir}}(\vec{r}_i, \vec{r}_j), \tag{13}$$

$$(\mathbf{G}_2^{\mathrm{dir}})_{ij} = \nabla(\nabla')^T g^{\mathrm{dir}}(\vec{r}_i, \vec{r}_j),$$

otherwise, where ∇ and ∇' are gradients with respect to the first and second arguments of the target functions. Define the reciprocal $\mathbf{G}$ matrices as

$$(\mathbf{G}_0^{\mathrm{rec}})_{ij} = g^{\mathrm{rec}}(\vec{r}_i, \vec{r}_j), \quad (\mathbf{G}_1^{\mathrm{rec}})_{ij} = \nabla g^{\mathrm{rec}}(\vec{r}_i, \vec{r}_j), \tag{14}$$

$$(\mathbf{G}_2^{\mathrm{rec}})_{ij} = \nabla(\nabla')^T g^{\mathrm{rec}}(\vec{r}_i, \vec{r}_j)$$

and define the overall $\mathbf{G}$ matrices as the sum of the direct and reciprocal $\mathbf{G}$ matrices:

$$\mathbf{G}_0 = \mathbf{G}_0^{\mathrm{dir}} + \mathbf{G}_0^{\mathrm{rec}}, \quad \mathbf{G}_1 = \mathbf{G}_1^{\mathrm{dir}} + \mathbf{G}_1^{\mathrm{rec}}, \quad \mathbf{G}_2 = \mathbf{G}_2^{\mathrm{dir}} + \mathbf{G}_2^{\mathrm{rec}}. \tag{15}$$

Note that $\mathbf{G}_1^{\mathrm{dir}}$ is an $N\times N$ matrix of 3×1 blocks, each block being a gradient of the corresponding $\mathbf{G}_0^{\mathrm{dir}}$ element; $\mathbf{G}_2^{\mathrm{dir}}$ is an $N\times N$ matrix of 3×3 blocks, each block being a Hessian of the corresponding $\mathbf{G}_0^{\mathrm{dir}}$ element. Note, also, that $(\mathbf{G}_0^{\mathrm{dir}})_{ij} = (\mathbf{G}_0^{\mathrm{dir}})_{ji}$, $(\mathbf{G}_1^{\mathrm{dir}})_{ij} = -(\mathbf{G}_1^{\mathrm{dir}})_{ji}$, and $(\mathbf{G}_2^{\mathrm{dir}})_{ij} = (\mathbf{G}_2^{\mathrm{dir}})_{ji}$. The same properties hold for the $\mathbf{G}^{\mathrm{rec}}$ and $\mathbf{G}$ matrices. Combining Eqs. (10)–(14), we get

$$E^{\mathrm{dir/rec}}(\mathbf{r}, \mathbf{d}) = \frac{1}{2}\mathbf{q}^T\mathbf{G}_0^{\mathrm{dir/rec}}(\mathbf{r})\mathbf{q} + \mathbf{d}^T\mathbf{G}_1^{\mathrm{dir/rec}}(\mathbf{r})\mathbf{q} + \frac{1}{2}\mathbf{d}^T\mathbf{G}_2^{\mathrm{dir/rec}}(\mathbf{r})\mathbf{d}, \tag{16}$$

where $\mathbf{q} = [q_1, q_2, \ldots, q_N]^T$ and $\mathbf{d} = [\vec{d}_1^{\,T}, \vec{d}_2^{\,T}, \ldots, \vec{d}_N^{\,T}]^T$. From Eqs. (9), (16), (15), and (5), the total electrostatic energy is as given by Eq. (2), $E^{\mathrm{el}}(\mathbf{r}, \mathbf{d}) = \frac{1}{2}\mathbf{q}^T\mathbf{G}_0(\mathbf{r})\mathbf{q} + \mathbf{d}^T\mathbf{G}_1(\mathbf{r})\mathbf{q} + \frac{1}{2}\mathbf{d}^T\mathbf{G}_2(\mathbf{r})\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{D}_\alpha^{-1}\mathbf{d}$, where $\mathbf{D}_\alpha = \mathrm{diag}(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_N)$ is the block diagonal polarizability matrix.

The induced dipole moments take the values that minimize the total electrostatic energy, $(\partial/\partial\mathbf{d})E^{\mathrm{el}}(\mathbf{r}, \mathbf{d}) = 0$, which gives Eq. (3) for the dipole moments, $(\mathbf{D}_\alpha^{-1} + \mathbf{G}_2)\mathbf{d} = -\mathbf{G}_1\mathbf{q}$. A component of the force $F_{k\sigma}$, $1 \leq k \leq N$, $\sigma = x, y, z$, is defined as the negative derivative of $E^{\mathrm{el}}(\mathbf{r}, \mathbf{d})$ with respect to $r_{k\sigma}$, taking into account the dependence of $\mathbf{d}$ on $\mathbf{r}$. However, it can be shown that because of the minimizing property, Eq. (3), the force is obtained by differentiating Eq. (2) as though $\mathbf{d}$ were independent of $\mathbf{r}$, yielding

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

$$F_{k\sigma}^{\mathrm{el}} = -\frac{1}{2}\mathbf{q}^T\left(\frac{\partial}{\partial r_{k\sigma}}\mathbf{G}_0\right)\mathbf{q} - \mathbf{d}^T\left(\frac{\partial}{\partial r_{k\sigma}}\mathbf{G}_1\right)\mathbf{q} - \frac{1}{2}\mathbf{q}^T\left(\frac{\partial}{\partial r_{k\sigma}}\mathbf{G}_2\right)\mathbf{d} \tag{17}$$

with analogous equations for $F_{k\sigma}^{\mathrm{dir}}$ and $F_{k\sigma}^{\mathrm{rec}}$. With the optimal dipole vector satisfying Eq. (3), the energy given by Eq. (2) simplifies to

$$E^{\mathrm{el}} = \frac{1}{2}\mathbf{q}^T\mathbf{G}_0\mathbf{q} + \frac{1}{2}\mathbf{d}^T\mathbf{G}_1\mathbf{q}. \tag{18}$$

B. Particle-mesh Ewald method

In this article, PME method refers to the smooth PME method.[31] In PME, the Ewald parameter β is chosen to be large, so that for the direct sum a small cut-off radius is needed for a given accuracy. This leads to an $O(N)$ computational cost for the direct sum. For the reciprocal sum, fast Fourier transforms (FFTs) are used and the computation cost is $O(N \log N)$. The implementations of the direct sum and the reciprocal sum are considered separately.

1. Direct sum implementation

The direct sum is truncated beyond a specified cutoff distance r_c. The implementation employs the recurrence given in Refs. 25, 42, and 44. The matrix $\mathbf{G}_2$ is used repeatedly when solving the dipole equation iteratively. To reduce re-computation of $\mathbf{G}_2$ for each iteration, three scalar quantities are stored for each pair of atoms within the real space cutoff radius r_c. The typical biomolecular systems has a density of 0.1 atom/Å^3, so for $r_c=8$ Å, the memory requirement is $3003N$ bytes, and for $r_c=10$ Å, the memory requirement is less than $6000N$ bytes. For a system with $N=10\,000$ atoms, this requires only 30 or 60 megabytes of memory, respectively. For larger systems, parallel computation is needed. For example, NAMD recommends 1000 atoms per processor for optimal efficiency.[45] The idea of storing dipole-dipole tensors to speed up iterations has been tried before, in a reaction-field model.[46]

2. Reciprocal sum

Two approximations are made in PME for the reciprocal sum computation.

The *first* approximation is

$$g^{\mathrm{rec}}(\vec{r},\vec{r}') \approx \frac{1}{\pi V}\hat{\sum_{\vec{m}\neq\vec{0};}}\frac{\exp(-\pi^2|\vec{m}|^2/\beta^2)}{|\vec{m}|^2}\times\exp(2\pi i\vec{m}\cdot(\vec{r}-\vec{r}')), \tag{19}$$

where the hat denotes a truncated sum with $-K_\alpha/2 \leq m_\alpha \leq K_\alpha/2-1$, $\alpha=1,2,3$, and where K_1, K_2, and K_3 are large even integers chosen so that the truncation error is negligible.

Corresponding to the truncated reciprocal lattice, we have a grid in real space with K_1, K_2, and K_3 grid points along each dimension. This motivates the following "u-representation" for a position vector $\vec{r}$:

$$\vec{r} = \vec{r}_{\mathrm{corner}} + [\vec{a}_1\vec{a}_2\vec{a}_3]\begin{bmatrix} u_1/K_1 \\ u_2/K_2 \\ u_3/K_3 \end{bmatrix}, \tag{20}$$

where $\vec{r}_{\mathrm{corner}}$ is the position of a corner of the simulation box, chosen so that $0 \leq u_\alpha < K_\alpha$, $\alpha=1,2,3$, constitutes the simulation box. Equivalently,

$$\vec{u} = \begin{bmatrix} K_1\vec{b}_1^T \\ K_2\vec{b}_2^T \\ K_3\vec{b}_3^T \end{bmatrix}(\vec{r}-\vec{r}_{\mathrm{corner}}) = \mathbf{T}(\vec{r}-\vec{r}_{\mathrm{corner}}). \tag{21}$$

The *second* approximation made by PME is an interpolation using periodic B-spline basis functions. Define a B-spline function of period $\mathbf{K}$ by

$$\Phi_K(u) = \sum_{i=-\infty}^{\infty} M_p(u-iK), \tag{22}$$

where $M_p(u)$ is a B-spline of degree $p-1$ with integer break points and support $0 \leq u \leq p$.

The basic approximation is

$$\exp\left(2\pi i\frac{mu}{K}\right) \approx b_K(m)\sum_{n=0}^{K-1}\Phi_K(u-n)\exp\left(2\pi i\frac{mn}{K}\right), \tag{23}$$

where m is an integer, and

$$b_K(m) = 1\Big/\sum_{n=1}^{p-1}\Phi_K(n)\cdot\exp(2\pi im(n-p)/K). \tag{24}$$

Remarkably, this approximation is exact at integer values of u; hence, this is an *interpolation* of $\exp(2\pi imu/K)$ from the grid.

The actual three-dimensional interpolation is

$$\begin{aligned}\exp(2\pi i\vec{m}\cdot(\vec{r}-\vec{r}_{\mathrm{corner}})) &= \exp\left(2\pi i\left[\frac{u_1m_1}{K_1}+\frac{u_2m_2}{K_2}+\frac{u_3m_3}{K_3}\right]\right)\\ &\approx b_{\vec{K}}(\vec{m})\sum_{\vec{n}}\varphi_{\vec{n}}(\vec{r})\exp\left(2\pi i\left[\frac{m_1n_1}{K_1}+\frac{m_2n_2}{K_2}+\frac{m_3n_3}{K_3}\right]\right),\end{aligned} \tag{25}$$

where

$$b_{\vec{K}}(\vec{m}) = b_{K_1}(m_1)b_{K_2}(m_2)b_{K_3}(m_3), \tag{26}$$

$$\varphi_{\vec{n}}(\vec{r}) = \Phi_{K_1}(u_1-n_1)\Phi_{K_2}(u_2-n_2)\Phi_{K_3}(u_3-n_3). \tag{27}$$

So

$$\exp(2\pi i\vec{m}\cdot(\vec{r}-\vec{r}_{\mathrm{corner}})) \approx \sum_{\vec{n}}\varphi_{\vec{n}}(\vec{r})\mathbf{F}_{\vec{n},\vec{m}}b_{\vec{K}}(\vec{m}), \tag{28}$$

with

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

$$\mathbf{F}_{\vec{n},\vec{m}} = \exp\left(2\pi i\left[\frac{m_1 n_1}{K_1} + \frac{m_2 n_2}{K_2} + \frac{m_3 n_3}{K_3}\right]\right). \tag{29}$$

Multiplication by $\mathbf{F}$ corresponds to a discrete Fourier transform. (The range of indices $\vec{m}$ is not what is needed by the FFT, but this discrepancy is easily fixed.)

From Eqs. (19) and (28), we have

$$g^{\mathrm{rec}}(\vec{r},\vec{r}') \approx \sum_{\vec{m}\neq\vec{0};} \sum_{\vec{n}} \sum_{\vec{n}'} \varphi_{\vec{n}}(\vec{r})\mathbf{F}_{\vec{n},\vec{m}}\mathbf{D}(\vec{m})\mathbf{F}^{*}_{\vec{n}',\vec{m}}\varphi_{\vec{n}'}(\vec{r}'), \tag{30}$$

where

$$\mathbf{D}(\vec{m}) = b_{\vec{K}}(\vec{m})\frac{1}{\pi V}\frac{\exp(-\pi^2|\vec{m}|^2/\beta^2)}{|\vec{m}|^2}b_{\vec{K}}(\vec{m})^{*}. \tag{31}$$

This gives

$$g^{\mathrm{rec}}(\vec{r},\vec{r}') \approx \sum_{\vec{n}} \sum_{\vec{n}'} \varphi_{\vec{n}}(\vec{r})(\mathbf{F}\mathbf{D}\mathbf{F}^{H})_{\vec{n}\vec{n}'}\varphi_{\vec{n}'}(\vec{r}'), \tag{32}$$

where $\mathbf{D}$ is the diagonal matrix having diagonal elements $\mathbf{D}(\vec{m})$ and H denotes the complex conjugate transpose.

Equation (14) gives

$$(\mathbf{G}_0^{\mathrm{rec}})_{ij} \approx (\mathbf{I}_h^0\mathbf{F}\mathbf{D}\mathbf{F}^{H}(\mathbf{I}_h^0)^T)_{ij}, \quad \text{or } \mathbf{G}_0^{\mathrm{rec}} \approx \mathbf{I}_h^0\mathbf{F}\mathbf{D}\mathbf{F}^{H}(\mathbf{I}_h^0)^T, \tag{33}$$

with

$$(\mathbf{I}_h^0)_{i\vec{n}} = \varphi_{\vec{n}}(\mathbf{r}_i). \tag{34}$$

The matrix $\mathbf{I}_h^0$ is a prolongation (or interpolation) operator which maps grid values to particle positions. Correspondingly, $(\mathbf{I}_h^0)^T$ restricts data at particle positions onto the grid. $\mathbf{I}_h^0$ is *sparse* due to the local support of the B-spline basis functions and the $(\mathbf{I}_h^0)_{i\vec{n}}$ can be regarded as weights for distributing charge q_i to grid nodes $\vec{n}$.

From Eqs. (33), (14), (32), and (34), we have

$$\mathbf{G}_0^{\mathrm{rec}} \approx \mathbf{I}_h^0(\mathbf{F}\mathbf{D}\mathbf{F}^{H})(\mathbf{I}_h^0)^T, \quad \mathbf{G}_1^{\mathrm{rec}} \approx \mathbf{I}_h^1(\mathbf{F}\mathbf{D}\mathbf{F}^{H})(\mathbf{I}_h^0)^T, \tag{35}$$

$$\mathbf{G}_2^{\mathrm{rec}} \approx \mathbf{I}_h^1(\mathbf{F}\mathbf{D}\mathbf{F}^{H})(\mathbf{I}_h^1)^T,$$

where each "element"

$$(\mathbf{I}_h^1)_{i\vec{n}} = \nabla\varphi_{\vec{n}}(\mathbf{r}_i) \tag{36}$$

is a 3×1 block. The matrix-vector multiplication for each $\mathbf{G}_i^{\mathrm{rec}}$ is simple and direct from the above expressions. For example, to compute $\mathbf{G}_2^{\mathrm{rec}}\mathbf{d}$, first restrict (distribute) $\mathbf{d}$ to grid nodes, then do a backward FFT, then multiply by the diagonal matrix $\mathbf{D}$, then do a forward FFT, and, in the end, interpolate back to real space.

We define the following quantities:

$$\text{Grid charges: } \mathbf{q}_h(\mathbf{I}_h^0)^T\mathbf{q}, \tag{37}$$

$$\text{Grid potential: } \mathbf{v}_h = \mathbf{F}\mathbf{D}\mathbf{F}^{H}(\mathbf{q}_h + (\mathbf{I}_h^1)^T\mathbf{d}). \tag{38}$$

From Eqs. (18) and (35), the reciprocal energy is

$$E^{\mathrm{rec}} = \frac{1}{2}\mathbf{q}_h^T\mathbf{v}_h, \tag{39}$$

while from Eqs. (17) and (35), the force is

$$F_{k\sigma}^{\mathrm{rec}} \approx -\left(\mathbf{q}^T\left(\frac{\partial}{\partial r_{k\sigma}}\mathbf{I}_h^0\right) + \mathbf{d}^T\left(\frac{\partial}{\partial r_{k\sigma}}\mathbf{I}_h^1\right)\right)\mathbf{v}_h, \tag{40}$$

giving

$$\mathbf{F}_k^{\mathrm{rec}} \approx -\sum_{\vec{n}} ((\nabla\varphi_{\vec{n}}(\vec{r}_k))q_k + (\nabla\nabla^T\varphi_{\vec{n}}(\vec{r}_k))\vec{d}_k)(\mathbf{v}_h)_{\vec{n}}. \tag{41}$$

The cost for computing the energy and force, assuming the dipole is known, is low since for a given value of k, $\varphi_{\vec{n}}(\vec{r}_k) \neq 0$ for only p^3 values of $\vec{n}$. Also, only two FFTs are needed regardless of whether or not there are dipoles.

The property $(\mathbf{G}_1^{\mathrm{rec}})_{ij} = -(\mathbf{G}_1^{\mathrm{rec}})_{ji}$ no longer holds after interpolation. Because of this, the sum of all the forces contributed by the reciprocal sum is not zero but a small number instead. A consequence is the loss of linear momentum conservation. If the small extra force is subtracted out[25] to conserve linear momentum, then the force is no longer an exact negative gradient, leading to a small energy drift for long simulations.

3. Overall computation sequence

The computation has three major steps: Preparation, solving the dipole equation iteratively, and computing the electrostatic energy and force.

The iteration scheme should be formulated carefully to avoid unnecessary cost. For example, the iteration[25]

$$\mathbf{d}_{m+1} = \mathbf{D}_\alpha[-\mathbf{G}_1\mathbf{q} - \mathbf{G}_2\mathbf{d}_m], \tag{42}$$

is formulated as follows to save two FFTs:

$$\mathbf{d}_{m+1} = -\mathbf{D}_\alpha(\mathbf{G}_1^{\mathrm{dir}}\mathbf{q} + \mathbf{G}_2^{\mathrm{dir}}\mathbf{d}_m + \mathbf{I}_h^1\mathbf{F}\mathbf{D}\mathbf{F}^{H}[(\mathbf{I}_h^0)^T\mathbf{q} + (\mathbf{I}_h^1)^T\mathbf{d}_m]). \tag{43}$$

The sequencing of the computation is summarized as follows:

(1) Preparation

- (a) Direct sum: Compute and store $\frac{1}{2}\mathbf{q}^T\mathbf{G}_0^{\mathrm{dir}}\mathbf{q}$, $-\mathbf{G}_1^{\mathrm{dir}}\mathbf{q}$. Compute and store three scalar quantities for each pair of atoms within the cutoff distance.
- (b) Reciprocal sum: Compute and store $\mathbf{q}_h = (\mathbf{I}_h^0)^T\mathbf{q}$, and partially compute and store $\mathbf{I}_h^1$ and $\mathbf{I}_h^2$.

(2) Solving the dipole equation:

- (a) Use some predictor to compute the initial guess.
- (b) Use some iterative method to solve the dipole equation, e.g., Eq. (43).

(3) Computing the electrostatic energy and force:

- (a) Direct sum: compute $\frac{1}{2}\mathbf{d}^T\mathbf{G}_1^{\mathrm{dir}}\mathbf{q}$, add it to $\frac{1}{2}\mathbf{q}^T\mathbf{G}_0^{\mathrm{dir}}\mathbf{q}$ to get E^{dir}. Compute force $\mathbf{F}^{\mathrm{dir}}$ from Eq. (10).
- (b) Reciprocal sum: compute $\mathbf{v}_h = \mathbf{F}\mathbf{D}\mathbf{F}^{H}(\mathbf{q}_h + (\mathbf{I}_h^1)^T\mathbf{d})$. Then compute E^{rec} and $\mathbf{F}^{\mathrm{rec}}$ according to Eqs. (39) and (41).
- (c) The electrostatic energy is $E^{\mathrm{dir}} + E^{\mathrm{rec}}$, the electrostatic force is $\mathbf{F}^{\mathrm{dir}} + \mathbf{F}^{\mathrm{rec}}$.

The total number of FFTs is $2 + 2\times$ number of iterations.

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

TABLE II. The polynomial extrapolation error.

Degree	0	1	2	3	4	5	6	7	8
δd (ppm)	1.6*e*4	2.2*e*3	4.9*e*2	1.4*e*2	58	26	29	53	99

III. WATER MODELS

Presented here are two water models for tests of accuracy and efficiency of the methods.

In the revised polarizable (RPOL) water model,[47] water molecules are rigid. The distance between the oxygen atom and a hydrogen atom is 1 Å and the H–O–H angle is 109.5°. The Lennard-Jones interaction $E_{\mathrm{LJ}}(r)=4\epsilon((\sigma/r)^{12}-(\sigma/r)^{6})$, with $\sigma=3.196$ Å and $\epsilon=0.160$ kcal/mol, is present only between oxygen atoms. Effective charges are $q_{\mathrm{O}}=-0.730e$ and $q_{\mathrm{H}}=0.365e$, where e is the charge of a proton. The isotropic polarizabilities are $\alpha_{\mathrm{O}}=0.52$ Å^3 and $\alpha_{\mathrm{H}}=0.170$ Å^3. Electrostatic interactions between atoms in the same molecule are excluded.

In the nonpolarizable simple point charge (SPC) water model,[48] water molecules are rigid. The O–H bond length is 1 Å and the H–O–H bond angle is 109.28°. The hydrogen atom has charge $0.41e$, and the oxygen atom has charge $-0.82e$.

For both systems, there are 216 water molecules. The simulation box is a cube of length 18.688 Å. The direct sum cutoff radius is 8 Å (chosen so that the direct sum and the Lennard-Jones interactions can be computed together), the grid size for the reciprocal sum computation is $18\times18\times18$, and β is chosen[25] so that the equal sign in $\mathrm{erfc}(\beta r_c)/r_c\leqslant\epsilon$ is satisfied for $\epsilon=10^{-6}$/Å.

Correctness of the implementation is confirmed in Ref. 42 by comparison of constant temperature simulations to Ref. 25.

IV. SELF-CONSISTENT SOLUTION

Define the root-mean-square (rms) error$=(1/\sqrt{N})\|\mathbf{d}_m-\mathbf{d}_{m-1}\|_2$, where m is the iteration step. Since the rms error can badly overestimate the true error for a fast converging iteration, at least two iterations are needed unless the initial guess is very close to the exact solution. In this study, convergence is claimed if the rms error is less than 10^{-6} or 10^{-7} Debye (1 Debye$=3.33564\times10^{-30}$ Coul m). The average dipole moment of an atom in the RPOL water system is 0.25 Debye,[42] so the above convergence criteria is 4 or 0.4 ppm. Typical forces due to dipoles contribute about 28% of the total electrostatic force.

A. Initial guess

Previous predictors[25] are based mostly on polynomial extrapolation, which is ill conditioned, meaning a small error in the previous dipoles is magnified by the prediction. Degree-$(k-1)$ polynomial extrapolation uses k previous dipole moments to predict the dipole moment at timestep $n+1$:

$$\mathbf{d}_0^{n+1}=\sum_{i=1}^{k}(-1)^{i+1}\binom{k}{i}\mathbf{d}^{n+1-i}, \tag{44}$$

where superscripts index timesteps.

Table II shows the prediction error in terms of relative errors

$$\delta d=\frac{\|\mathbf{d}-\mathbf{d}_{\mathrm{exact}}\|_2}{\|\mathbf{d}_{\mathrm{exact}}\|_2} \tag{45}$$

computed for some snapshots from a MD simulation. The numbers change from case to case, but the trend remains about the same.

A good initial guess reduces the computational cost significantly (see Figs. 1 and 2; the various iterative methods are described later; the number of iterations is an average over 1000 timesteps). The results degrade when the degree is greater than five.

The least-squares prediction is obtained by answering the following question after $\mathbf{d}^n$ is computed: What is the best

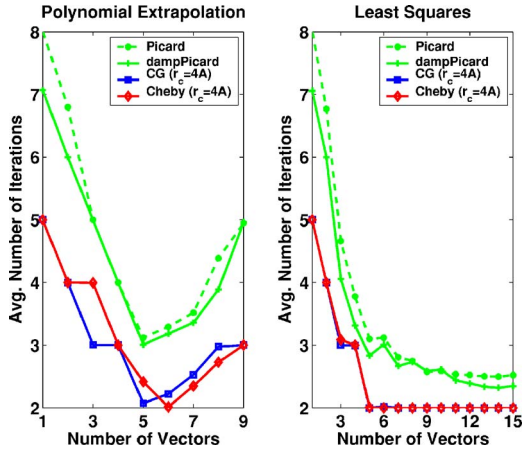

FIG. 1. Average number of iterations for different methods. The rms convergence criterion is 4 ppm. A "vector" is a set of dipoles moments at an instant in time.

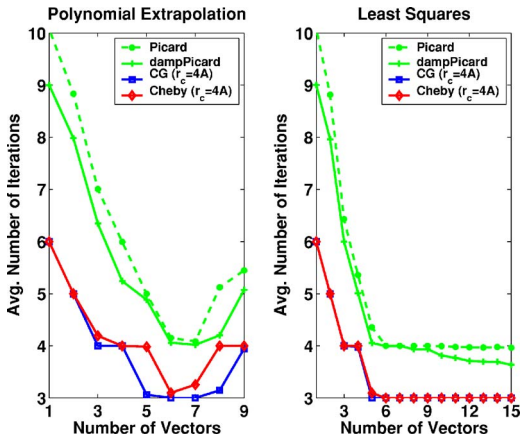

FIG. 2. Average number of iterations for different methods. The rms convergence criterion is 0.4 ppm.

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

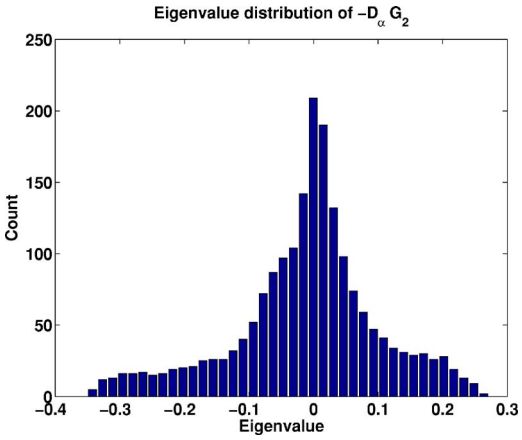

FIG. 3. Typical eigenvalue distribution of matrix $-\mathbf{D}_\alpha\mathbf{G}_2$.

prediction of $\mathbf{d}^n$ obtainable from a linear combination of $\mathbf{d}^{n-1},\ldots,\mathbf{d}^{n-k}$? One answer is to choose coefficients $c_1,c_2,\ldots,c_k$ that minimize

$$\left\|\mathbf{d}^n-\sum_{i=1}^{k}c_i\mathbf{d}^{n-i}\right\|_2^2. \tag{46}$$

The coefficients are obtained by solving the normal equations. After computing them, we use them to predict the dipole moment at the next timestep:

$$\mathbf{d}^{n+1}_{0,\text{least sqaures}}=\sum_{i=1}^{k}c_i\mathbf{d}^{n+1-i}. \tag{47}$$

The data show that least-squares prediction is a little better than the optimal polynomial prediction: $\delta d_{\text{optimal,polynomial}}\approx 2\times 10^{-5}$ and $\delta d_{\text{optimal,least squares}}\approx 1\times 10^{-5}$.

The least-squares prediction incurs a negligibly small amount of extra work at each step: In our test cases, the prediction cost is less than 1% of the overall electrostatic computation.

B. Iteration method

Iterative methods for linear systems can be classified as stationary, e.g., the Jacobi method, and nonstationary, e.g., the CG method and the Chebyshev semi-iterative method.

Stationary methods split the coefficient matrix into the sum of two matrices, one of which is easy to invert. For our problem, there is little choice but to split the matrix into $\mathbf{D}_\alpha^{-1}$ plus $\mathbf{G}_2$, and use the iteration $\mathbf{d}_{m+1}=\mathbf{D}_\alpha(-\mathbf{G}_1\mathbf{q}-\mathbf{G}_2\mathbf{d}_m)$, given previously as Eq. (42). We call it the *Picard* iteration due to the natural breakup into a simple dominant part plus a lesser part. It is not a Jacobi iteration because the diagonal elements of $\mathbf{G}_2$, although small, are not zero. The eigenvalues of the iteration matrix $-\mathbf{D}_\alpha\mathbf{G}_2$ cluster around zero, as shown in Fig. 3. If we decompose the error vector into components, each parallel to an eigenvector, those components whose corresponding eigenvalues are close to zero are damped away after one Picard iteration. So, the method is fairly efficient for solving the dipole equation, and it is widely used in the literature.

The convergence factor of a stationary method is determined by the spectral radius of the iteration matrix. For Picard iteration, the spectral radius $\rho(-\mathbf{D}_\alpha\mathbf{G}_2)$ is about 0.34 (see Fig. 3). If the spectrum of the iteration matrix is not symmetric about zero, a damping scheme can be used and the optimal damping factor can be determined by the largest and smallest eigenvalues of the iteration matrix. In our tests, damping gives only a marginal improvement, reducing the convergence factor from 0.34 to 0.29.

The CG and Chebyshev semi-iterative methods are for solving symmetric positive-definite systems. The CG method minimizes the "energy norm" of the error, while the Chebyshev method, with optimal parameters, is targeted at minimizing the 2-norm of the error. The error is bounded by[40]

$$\frac{\|\mathbf{d}_m-\mathbf{d}_{\text{exact}}\|}{\|\mathbf{d}_0-\mathbf{d}_{\text{exact}}\|}\leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m, \tag{48}$$

where $\kappa=\lambda_{\max}/\lambda_{\min}$ is the condition number of the preconditioned matrix and $\|\cdot\|$ is the energy norm for CG and the 2-norm for the Chebyshev method.

It takes two matrix-vector multiplications for the standard CG method to get the first update of the solution: One for computing the residual to determine the search direction, another for computing the optimal distance to move along the search direction. After that, each iteration requires one matrix-vector multiplication. So for the same number of updates of the solution, CG does one more (expensive) matrix-vector multiplication than other methods. This extra cost can be saved by a "peek" step which uses the available residual $\mathbf{r}$ to do one Picard iteration: $\mathbf{d}'=\mathbf{d}+\mathbf{D}_\alpha\mathbf{r}=\mathbf{D}_\alpha(-\mathbf{G}_1\mathbf{q}-\mathbf{G}_2\mathbf{d})$. The following is the pseudo-code adapted from Ref. 40 (Sec. 8.3) for the modified CG method:

$\mathbf{r}:=-\mathbf{G}_1\mathbf{q}-(\mathbf{D}_\alpha^{-1}+\mathbf{G}_2)\mathbf{d}$;
solve $\mathbf{Ms}=\mathbf{r}$ for $\mathbf{s}$;
$c:=\mathbf{r}^T\mathbf{s}$;
FOR *iter* $:=1,2,\ldots,$ *maximum_iteration*
$\mathbf{u}:=(\mathbf{D}_\alpha^{-1}+\mathbf{G}_2)\mathbf{s}$;
$a:=c/\mathbf{s}^T\mathbf{u}$; $\mathbf{d}:=\mathbf{d}+a\cdot\mathbf{s}$; $\mathbf{r}:=\mathbf{r}-a\cdot\mathbf{u}$;
/*peek*/ $\mathbf{d}':=\mathbf{d}+\mathbf{D}_\alpha\mathbf{r}$; IF $(\|\mathbf{d}'-\mathbf{d}\|_2^2<\epsilon^2)$ BREAK;
solve $\mathbf{Mt}=\mathbf{r}$ for $\mathbf{t}$;
$c_{\text{new}}:=\mathbf{r}^T\mathbf{t}$; $\mathbf{s}:=\mathbf{t}+(c_{\text{new}}/c)\cdot\mathbf{s}$; $c:=c_{\text{new}}$;
END *iter*;
claim $\mathbf{d}'$ as the solution.

Here, $\mathbf{s}$ is the search direction, $\mathbf{M}$ is a preconditioner, $\mathbf{t}$ and $\mathbf{u}$ are temporary vectors, ϵ is the convergence criteria, a is a scalar marking the optimal distance along the direction $\mathbf{s}$, and c and c_{new} are scalars. Compared to the standard CG implementation, the only change is the added "peek" step. This inexpensive $O(N)$ computation does not alter the CG search path, but finds a converged solution one step earlier than the standard CG method in most cases. It does not actually peek at the next solution computed by the CG method; rather it substitutes a suboptimal but acceptable solution for the last CG step.

The implementation of the Chebyshev method is straightforward (Ref. 40, Sec. 8.2). The method requires a good estimation of the spectral range, which, according to our experience, changes little during MD simulations. So, this expensive computation can be done once for all, and the long-time performance suffers little.

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

TABLE III. Convergence factors of different iteration methods.

Method	Condition number	Convergence factor
Picard	⋯	0.34
Damped Picard	⋯	0.29
CG, Cheby preconditioned by **M** with r_c=0 Å	1.80	0.15
CG, Cheby preconditioned by **M** with r_c=3 Å	1.44	0.09
CG, Cheby preconditioned by **M** with r_c=4 Å	1.38	0.08

C. Preconditioner

A preconditioner, an easy-to-invert approximation to the coefficient matrix of a linear system, is used to reduce the condition number κ of that matrix, thereby accelerating convergence according to Eq. (48). Because only the inverse of the preconditioner is used in the iteration process, we can approximate the inverse of $\mathbf{D}_\alpha^{-1}+\mathbf{G}_2$ directly by

$$(\mathbf{D}_\alpha^{-1}+\mathbf{G}_2)^{-1}=\mathbf{D}_\alpha-\mathbf{D}_\alpha\mathbf{G}_2\mathbf{D}_\alpha+\mathbf{D}_\alpha\mathbf{G}_2\mathbf{D}_\alpha\mathbf{G}_2\mathbf{D}_\alpha-\cdots \tag{49}$$

$$\approx\mathbf{D}_\alpha-\mathbf{D}_\alpha\mathbf{G}_2\mathbf{D}_\alpha\approx\mathbf{D}_\alpha-\mathbf{D}_\alpha\mathbf{N}\mathbf{D}_\alpha, \tag{50}$$

where **N** is a "local approximation" to $\mathbf{G}_2$. In particular, define

$$\mathbf{N}_{ij}=\begin{cases}\mathbf{T}(\vec{r}_{ij}), & |\vec{r}_{ij}|<r_c \text{ and } j\notin\chi(i),\\ 0, & \text{otherwise},\end{cases} \tag{51}$$

where the cutoff radius r_c is a parameter different from that of the direct sum and $\mathbf{T}(\vec{r})$ is the 3×3 dipole-dipole interaction tensor

$$\mathbf{T}(\vec{r})=\frac{1}{r^3}\left(\mathbf{I}-3\frac{\vec{r}\vec{r}^T}{r^2}\right), \tag{52}$$

with r as the 2-norm of $\vec{r}$ and **I** as the identity matrix. The above preconditioner considers only pairs of dipoles with separation distance $<r_c$. On average, for r_c=3 Å, we need only maintain a list of average length <4 for each atom. If the cutoff is 4 Å, the list has an average length <12.

Table III provides the convergence factor for a typical matrix arising from MD simulations. For a cutoff radius larger than 4 Å, the reduction in number of iterations is less significant, but the cost in applying the preconditioner, proportional to r_c^3, increases significantly.

The two preconditioners used by Ref. 49 are similar to ours but not as effective in reducing total computational cost because their main concern is to use existing software modules of the underlying fast electrostatic solver.

TABLE IV. The cost for computing the electrostatic energy and force of the RPOL and SPC models.

Time (s)	SPC	RPOL	Increase
Direct sum	0.02116	0.02651	25%
Reciprocal sum	0.00204	0.00331	62%
Solving dipole	⋯	0.01511	⋯
Overall	0.023 20	0.044 94	94%

D. Timing results

Table IV shows a comparison between the computation of the polarizable RPOL water model and the charge-only SPC (Ref. 48) water model. The time is averaged over 1000 MD steps with a 1 fs timestep. To solve the dipole equation for the RPOL water model, the least-squares prediction is used with ten previous dipoles and the peek-CG method is used with the preconditioner designed in Sec. IV C having a 4 Å cutoff. The code is tested on a 3.06 GHz Intel Pentium 4 CPU; the compiler is Intel C++ Compiler 8.0 with flags "-fast -unroll -xN." We define the cost for computing the charge-only model to be 1 work unit. Then, the cost of one iteration is about 0.33 work units, much faster than about 1 unit in Ref. 25 It is not clear how they implement the matrix-vector multiplication. Possibly, they do not use a neighbor list or reuse previously computed values (see Sec. II B 1). Note also that with dipole moments given, the polarizable model incurs only about 28% overhead with respect to the nonpolarizable model computations. This is consistent with Ref. 25, in which the corresponding cost is 25–30%. Detailed timing results are given in Fig. 4.

The prediction cost is negligible: Even when 15 previous dipole moments are used and only two iterations are needed for convergence, the extra cost for prediction is only 0.88% of the total electrostatic computation.

The preconditioner is constructed before the beginning of the iteration process. This has an estimated cost of 0.08 work units. This step could be integrated into the "preparation" phase, thereby saving even more time.

Table V provides timing results in work units for the peek-CG method with 4 Å cutoff when the initial guess is

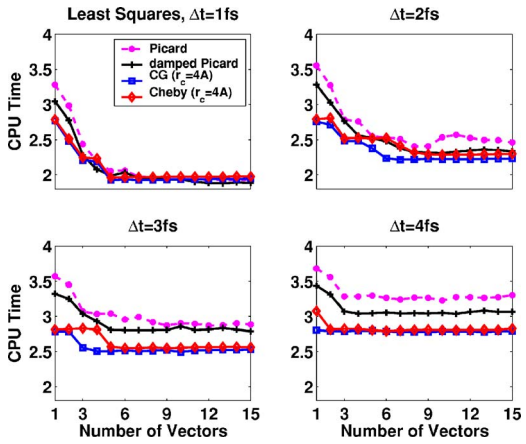

FIG. 4. Computational cost in work units for different timesteps. The relative rms convergence criteria is 4 ppm.

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

TABLE V. The cost of the self-consistent computation if the iteration starts from $\mathbf{d}_0=0$.

Relative convergence criteria	400 ppm	40 ppm	4 ppm
Average number of iterations	4.000	5.000	6.000
Computational cost	2.50	2.77	3.09

zero. Estimated from Table V, each iteration costs less than 0.3 work units. Note that when we do not make a prediction, the computational cost is independent of the timestep. Table V is discussed further in Sec. V.

E. Comparison to the extended Lagrangian approach

The extended Lagrangian approach is considered faster than the self-consistent approach, since the former does not solve the dipole equation. However, the longest timestep an extended Lagrangian method can take is 1 fs,[24,25] while the self-consistent approach does not pose an upper limit on the possible timestep and the cost increase is modest when a larger timestep is used. In MD simulations, the timestep for computing the electrostatic energy/force can vary from 1 fs for velocity-Verlet method with fully flexible bonds, to 2 fs when covalent hydrogen bonds are rigid, to as large as 6 fs using multiple-time stepping.[45,50] We carry out simulations with timesteps up through 4 fs, the largest that one can use without incurring significant energy drift with the velocity-Verlet method for the RPOL and SPC water systems.

Table VI tells us that the self-consistent computation with a timestep of 2 fs or more is faster than the extended Lagrangian approach. The dipole equation is solved by least-squares prediction with ten previous dipoles and the peek-CG method whose preconditioner is built with a 4 Å cutoff radius. Timing results are given in Fig. 4.

The average number of iterations needed for different timesteps is given in Fig. 5. The upper left figure is the same as Fig. 1. The relative rms convergence tolerance is 4 ppm, and all graphs in the figure have the same legend. For all timesteps, the least-squares predictor is consistently better than, or as good as, the polynomial predictor, which is not shown here. For larger timesteps, prediction helps less, while the iteration method is more important.

V. ENERGY DRIFT

For deterministic simulations, conservation of energy or equivalent conserved quantity is very important. But Fig. 6 shows that self-consistent computations can lead to significant energy drift unless fully converged. This is because the symplecticness of the underlying integrator is compromised

TABLE VI. The cost in work units per femtosecond for computing the electrostatic energy and force.

Timestep	1 fs	2 fs	3 fs	4 fs	Extended Lagrangian[a]
Computational cost per fs	1.94	1.11	0.83	0.70	1.25–1.30

[a]See Ref. 25.

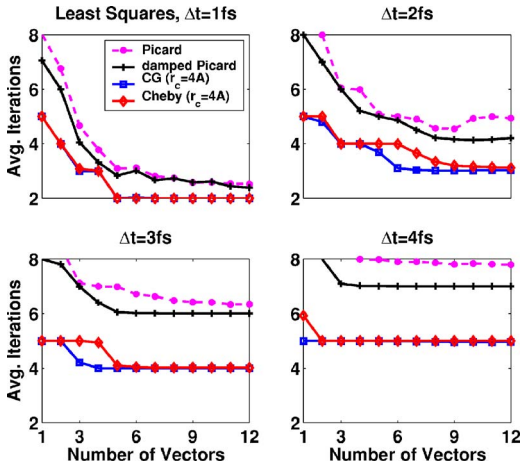

FIG. 5. Average number of iterations for different timesteps.

in two ways: (i) The computed force is not conservative due to the iterative solution being inexact, and (ii) the solution is history dependent due to the prediction. We find that (ii), which destroys the volume-preserving property, is more detrimental than (i) in causing the energy drift. The volume-preserving property is maintained if history is not used.[42] Integrators which do not preserve volume can lead to serious problems, such as the flying-ice cube phenomenon.[51]

Because energy drift is unavoidable for MD simulations, some energy drift should be tolerated when evaluating a method. Currently, typical simulation lengths are tens of nanoseconds, so a reasonable criterion is that the total energy drift in a 20 ns simulation should be no more than a 5 K change in system temperature.

We first, however, determine a suitable convergence criterion for the iteration based on the PME error. For this purpose, we look at relative 2-norm errors of the dipole moment and the electrostatic force (with exact values computed by the standard Ewald sum method using very stringent error tolerances).

Table VII shows the error introduced by PME as well as by iteration with different convergence criteria. The quantity δd is defined in Eq. (45) and δF^{el} is defined similarly. For the "0 ppm" column, the relative dipole convergence criteria is set to 4×10^{-15}, so the observed error comes from the PME

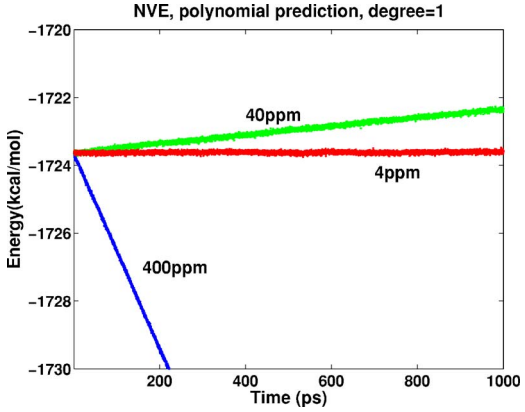

FIG. 6. Energy drifts from 1 ns simulations for a RPOL water system with timestep 1 fs.

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

TABLE VII. The error of the dipole and the electrostatic force for the PME method for different convergence criteria.

	0 ppm	4 ppm	40 ppm	400 ppm	4000 ppm
δd (ppm)	153	153	153	167	689
δF^{rel} (ppm)	136	136	136	138	240

method. Observe that the relative rms convergence tolerance can be as high as 400 ppm without introducing significant error.

To study dependency of energy drift on predictions, we exclude other factors which can cause the energy drift. Constraints are enforced by the SETTLE method,[52] the net force generated by the PME method is not subtracted out, and the van der Waals potential is cut off with a C^1 switching function. Specifically, the potential is multiplied by $s(r)=(r_c^2-r^2)^2(r_c^2+2r^2-3r_s^2)/(r_c^2-r_s^2)^3$ for $r_s<r<r_c$, where $r_s=6$ Å and $r_c=8$ Å.

The energy drift is not perceptible using 400 ppm as the relative rms convergence criteria with 0 initial guess. The integrator is the velocity-Verlet method, the iteration method is the peek-CG method, and the preconditioner is constructed with a 4 Å cutoff. Hence, an inexact solution alone, and a nonconservative force, does not necessarily cause significant energy drift.

On the other hand, when history is used, the drift can be significant as is seen from Fig. 7. The iteration method is peek-CG with an $r_c=4$ Å cutoff preconditioner. The line with legend "400 ppm, 0" is the energy drift from a simulation in which the dipole equation is solved by 0 initial guess and a relative rms convergence criteria of 400 ppm. We observe from Fig. 7 that the energy drift is approximately proportional to the rms convergence error. Also, for polynomial extrapolation, the energy drift strongly depends on the polynomial degree. Higher degree leads to less drift in general. For least-squares prediction, the dependence on the number of previous dipole moments is not so strong.

Accuracy needs and cost determine which method to use. Although symplecticness is not preserved if the dipole solution is not exact, phase space volume preservation and energy conservation probably suffice. So, with respect to quality, zero-guess self-consistent computation is better. However, as Table V shows, the extra cost will be about 150% compared to the charge-only computation. On the other hand, if we use accurate prediction, we can obtain very accurate dipole moments with about 94% extra cost compared with the charge-only computation and keep the energy drift negligibly small. When the computed dipole moment has a small error, we expect the phase space volume change to be small.

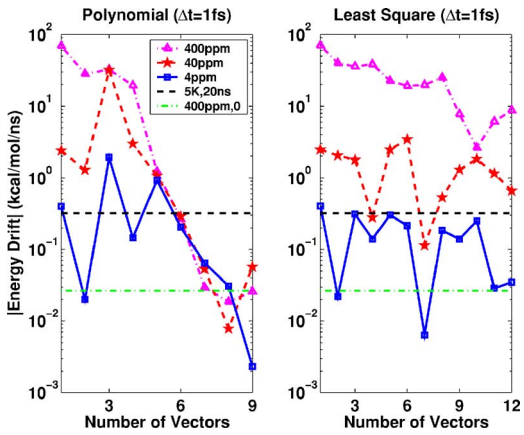

FIG. 7. Energy drifts when dipoles from previous timesteps are used for an accurate initial guess.

The always-stable-prediction-corrector (ASPC) method proposed in Ref. 53 tries to reduce the energy drift by partially recovering the time reversibility of the numerical integration. With a timestep of 1 fs, the ASPC method can maintain a constant energy level with only one damped Picard iteration by using a quasi-time-reversible predictor and a proper damping factor. The method is very fast since only one iteration is needed, but it fails to conserve energy when the timestep is 2 fs. The major drawback of the method is that it is not volume preserving. A minor drawback of the method is its low accuracy and lack of direct accuracy control. In Ref. 42, the ASPC method is improved by an accuracy control mechanism with a small extra cost, which is achieved by the efficient iteration algorithm as well as the combination of the quasi-time-reversible predictor in the ASPC method with the least-squares predictor.

ACKNOWLEDGMENTS

This material is based upon work performed at the University of Illinois at Urbana—Champaign and supported by the NIH (Grant No. P41-RR05969) and by the National Science Foundation (Grant No. DMS 0503657). The authors are grateful to Sanghyun Park for conducting initial encouraging computer experiments and to Klaus Schulten and Emad Tajkhorschid for helpful information.

[1] J. W. Ponder and D. A. Case, in *Advances in Protein Chemistry*, Protein Simulations Vol. 66, edited by F. M. Richards, D. S. Eisendberg, and J. Kuriyan (Elsevier, Amsterdam, 2003), pp. 27–85.
[2] A. Glattli, X. Daura, and W. F. van Gunsteren, J. Chem. Phys. **116**, 9811 (2002).
[3] G. Tiraboschi, B. Roques, and N. Gresh, J. Comput. Chem. **20**, 1379 (1999).
[4] H. Saint-Martin, B. Hess, and H. J. C. Berendsen, J. Chem. Phys. **120**, 11133 (2004).
[5] J. W. Caldwell and P. A. Kollman, J. Am. Chem. Soc. **117**, 4177 (1995).
[6] Y. Ding, D. N. Bernardo, K. Krogh-Jespersen, and R. M. Levy, J. Phys. Chem. **99**, 11575 (1995).
[7] H. Xu, H. A. Stern, and B. J. Berne, J. Phys. Chem. B **106**, 2054 (2002).
[8] J. M. Hermida-Ramon, S. Brdarski, G. Karlström, and U. Berg, J. Comput. Chem. **24**, 161 (2002).
[9] J. Baucom, T. Transue, M. Fuentes-Cabrera, J. M. Krahn, T. A. Darden, and C. Sagui, J. Chem. Phys. **121**, 6998 (2004).
[10] P. Salvador, J. E. Curtis, D. J. Tobias, and P. Jungwirth, Phys. Chem. Chem. Phys. **5**, 3752 (2003).
[11] D. N. Bernado, Y. Ding, K. Krogh-Jespersen, and R. M. Levy, J. Phys. Chem. **98**, 4180 (1994).
[12] S. W. Rick and S. J. Stuart, in *Reviews in Computational Chemistry*, edited by K. B. Lipkowitz and D. B. Boyd (Wiley-VCH, Berlin, 2002), Vol. 18, pp. 89–146.
[13] A. D. MacKerell Jr., J. Comput. Chem. **25**, 1584 (2004).
[14] C. Chipot and J. Àngyán, New J. Chem. **29**, 411 (2005).
[15] J. G. Àngyán, C. Chipot, and F. Dehez, C Hättig, G. Jansen, and C. Millot, J. Comput. Chem. **24**, 997 (2003).
[16] J. Applequist, J. R. Carl, and K. K. Fung, J. Am. Chem. Soc. **94**, 2952 (1972).

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp

[17] G. G. Ferenczy and C. A. Reyholds, J. Phys. Chem. **105**, 11470 (2001).
[18] T. A. Halgren and W. Damm, Curr. Opin. Struct. Biol. **11**, 236 (2001).
[19] H. A. Stern, G. A. Kaminski, J. L. Banks, R. Zhou, B. J. Berne, and R. A. Friesner, J. Phys. Chem. B **103**, 4730 (1999).
[20] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, J. K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman, J. Am. Chem. Soc. **117**, 5179 (1995).
[21] R. Car and M. Parrinello, Phys. Rev. Lett. **55**, 2471 (1985).
[22] M. Sprik and M. L. Klein, J. Chem. Phys. **89**, 7556 (1988).
[23] D. K. Remler and P. A. Madden, Mol. Phys. **70**, 921 (1990).
[24] D. Spangberg and K. Hermansson, J. Chem. Phys. **120**, 4829 (2004).
[25] A. Toukmaji, C. Sagui, J. Board, and T. Darden, J. Chem. Phys. **113**, 10913 (2000).
[26] T. Morishita and S. Nosé, Phys. Rev. B **59**, 15126 (1999).
[27] M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias, and J. D. Joannopoulos, Rev. Mod. Phys. **64**, 1045 (1992).
[28] E. Harder, B. Kim, R. A. Friesner, and B. J. Berne, J. Chem. Theory Comput. **1**, 169 (2005).
[29] J. A. C. Rullmann and R. T. van Duijnen, Mol. Phys. **63**, 451 (1988).
[30] T. N. Nymand and P. Linse, J. Chem. Phys. **112**, 6386 (2000).
[31] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. Pederson, J. Chem. Phys. **103**, 8577 (1995).
[32] M. A. Kastenholz and P. H. Hünenberger, J. Phys. Chem. B **108**, 774 (2004).
[33] W. Weber, P. H. Hünenberger, and J. A. McCammon, J. Phys. Chem. B **104**, 3668 (2000).
[34] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (McGraw–Hill, New York, 1981).
[35] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems* (MIT Press, Cambridge, MA, 1988).
[36] R. D. Skeel, I. Tezcan, and D. J. Hardy, J. Comput. Chem. **23**, 673 (2002).
[37] A. Brandt and A. A. Lubrecht, J. Comput. Phys. **90**, 348 (1990).
[38] A. D. MacKerell Jr., D. Bashford, M. Bellott *et al.*, J. Phys. Chem. B **102**, 3586 (1998).
[39] L. Kalé, R. Skeel, R. Brunner, M. Bhandarkar, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten, J. Comput. Phys. **151**, 283 (1999).
[40] P. Deuflhard and A. Hohmann, *Numerical Analysis in Modern Scientific Computing: An Introduction* (Springer, Berlin, 2003).
[41] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra* (SIAM, Philadelphia, 1997).
[42] W. Wang, Ph.D. thesis, University of Illinois at Urbana—Champaign (2005), also Department of Computer Science, Report No. UIUCDCS-R-2005-2522, University of Illinois at Urbana—Champaign, Ill. 2005. Available online at ⟨http://www.cs.uiuc.edu/research/techreports.php⟩.
[43] S. W. de Leeuw, J. W. Perram, and E. R. Smith, Proc. R. Soc. London, Ser. A **373**, 27 (1980).
[44] W. Smith, Collaborative Computational Projects 5 Quarterly **26**, 43 (1987), a revised version is available from the author.
[45] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, and K. Schulten, J. Comput. Chem. **26**, 1781 (2005).
[46] G. Ruocco and M. Sampoli, Mol. Phys. **82**, 875 (1994).
[47] L. X. Dang, J. Chem. Phys. **97**, 2659 (1992).
[48] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, and J. Hermans, in *Intermolecular Forces*, edited by B. Pullman (Reidel, Dordrecht, 1981), pp. 331–342.
[49] A. Nakano, Comput. Phys. Commun. **104**, 59 (1997).
[50] M. Tuckerman, B. J. Berne, and G. J. Martyna, J. Chem. Phys. **97**, 1990 (1992).
[51] S. C. Harvey, R. K. Z. Tan, and T. E. Cheatham, J. Comput. Chem. **19**, 726 (1998).
[52] S. Miyamoto and P. A. Kollman, J. Comput. Chem. **13**, 952 (1992).
[53] J. Kolafa, J. Comput. Chem. **25**, 335 (2003).

Downloaded 27 Oct 2005 to 128.210.126.199. Redistribution subject to AIP license or copyright, see http://jcp.aip.org/jcp/copyright.jsp