Multiple Grid Methods for Classical Molecular Dynamics

ROBERT D. SKEEL, ISMAIL TEZCAN, DAVID J. HARDY
Department of Computer Science and Beckman Institute, University of Illinois, 1304 West Springfield Avenue, Urbana, Illinois 61801-2987

Received 5 June 2001; Accepted 2 January 2002
DOI 10.1002/jcc.10072

Abstract: Presented in the context of classical molecular mechanics and dynamics are multilevel summation methods for the fast calculation of energies/forces for pairwise interactions, which are based on the hierarchical interpolation of interaction potentials on multiple grids. The concepts and details underlying multigrid interpolation are described. For integration of molecular dynamics the use of different time steps for different interactions allows longer time steps for many of the interactions, and this can be combined with multiple grids in space. Comparison is made to the fast multipole method, and evidence is presented suggesting that for molecular simulations multigrid methods may be superior to the fast multipole method and other tree methods.

 J Comput Chem 23: 673–684, 2002

Key words: N-body solver; multigrid method; fast multipole method; fast electrostatics; molecular dynamics

Introduction

Calculation of pairwise interactions for a set of N particles is the computational bottleneck for many physical problems. In particular, this is the case for simulations of biological molecules, which require enormous numbers of such calculations. Not surprisingly, there is a great need for faster N-body calculations using algorithms suitable for large-scale parallelism. It is well known that inverse-square-law forces between pairs of N particles can be calculated to a given accuracy in a time proportional to N or $N \log N$ by methods such as the fast multipole method[1] and multipole generalizations of other tree algorithms like those in refs. 2 and 3. One readily available implementation, the parallel program DPMTA,[4,5] together with the parallel molecular dynamics program NAMD,[6] has been used for a number of biomolecular studies.[7] However, there is a large constant multiplying the N in the running time. This can be partly compensated for by the use of multiple time stepping, in which longer time steps are used for the full electrostatics calculation. It is tempting to reduce the cost further by computing the force with errors that are comparable to those introduced by the temporal discretization (which can be estimated by the method of modified equations[8–10]). The problem with this is that the "quality" of such force evaluation errors can be far worse than discretization errors introduced by a good integrator. Indeed, experience[11,12] indicates a need to use many terms of a multipole expansion to avoid energy drift. Expensive high-order approximations would not be necessary, it seems, if the force approximations were continuous as functions of particle positions. Investigated here are alternative algorithms that calculate *continuous* forces in linear time using hierarchical interpolation of interaction potentials on multiple grids.

The idea of using multiple grids, introduced a decade ago for computing integral transforms,[13] is less well known than tree methods. These "multigrid" methods *are not iterative.* At a fundamental level such methods share with tree methods a linear $\mathcal{O}(N)$ efficiency based on a separation of spatial scales.[14] Both types of methods pool the effects of neighboring particles to approximate their interactions with more distant particles. They differ in how they separate spatial scales: tree methods hierarchically *separate particle pairs* into near pairs and far pairs, whereas multigrid methods hierarchically *separate the force potential* into a short-range part plus a smooth part.[15] The multigrid method approximates the smooth part of the potential between each particle using basis functions defined on a fine grid. The calculation is thus reduced to calculating interactions between pairs of points on the fine grid, and this is approximated on a coarser grid in the same manner that the irregular particle level calculation was approximated on the fine grid. The ideas underlying multigrid interpolation are described in greater detail in the next section.

Only recently has the multigrid method been applied to the N-body problem. The one published implementation[16] is for particle monopoles and dipoles in two dimensions, and gives timing

Correspondence to: R. D. Skeel; e-mail: skeel@cs.uiuc.edu

Contract/grant sponsor: NSF; contract/grant numbers: DMS-9971830 and DBI-9974555

Contract/grant sponsor: NIH; contract/grant number: P41RR05969

comparisons against the direct calculation. Given in this article in the Method Details section are the mathematical details of an implementation of the multigrid method for nonperiodic boundary conditions in three dimensions. (Modification for periodic boundary conditions is in progress.) Also, recently, a more traditional multigrid solver has been applied to a Poisson equation formulation for calculating the smooth part of the interactions,[12] motivated by the desire to run more efficiently than FFT-based methods such as PME (particle mesh Ewald)[17] on massively parallel computers.

The Comparison to Tree Methods section compares multigrid to the fast multipole method. Tests on a 20,544-atom model of water show that for the same accuracy the multigrid is twice as fast as the DPMTA implementation of the fast multipole method. Even greater speedups are expected in the context of dynamics for two reasons. First, multigrid methods calculate a continuously differentiable approximation to the potential energy function, whereas tree methods calculate a discontinuous approximation, which necessitates the use of smaller error tolerances. Second, only the multigrid method provides a *smoothly varying* partitioning of forces into different length scales, which can be exploited by multiple time stepping. Other advantages of the multigrid method are its relative simplicity and its applicability to general potentials, for example, van der Waals. On the other hand, the proposed method has not been demonstrated to be as efficient for high accuracy as the fast multipole method, especially recent versions such as in ref 18.

The Algorithm Details section discusses algorithm details, and then we give additional experimental analysis of accuracy.

Basic Idea

The electrostatic energy due to a collection of N charged atoms can be expressed as

$$U^{\mathrm{el}}(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\notin\chi(i)}\frac{q_i q_j}{4\pi\epsilon_0|\vec{r}_j - \vec{r}_i|}, \qquad (1)$$

where $\vec{r}_i$ is position, q_i is (partial) charge, ϵ_0 is the dielectric constant, and $\chi(i)$ is the set of exclusions for atom i. These exclusions consist of i itself and typically other atoms j that are present in the same covalent energy terms as i (bond lengths, angles, dihedrals, and improper dihedrals). The forces are obtained as gradients

$$\vec{F}_i^{\mathrm{el}} = -\nabla_i U^{\mathrm{el}},$$

and this is also the case for the approximations presented in this article. The fast methods presented can also be applied to the product of the Hessian of U^{el} with a $3N$-dimensional vector.

A variety of boundary conditions are used in molecular dynamics, for example, the set of atoms may be simply in a vacuum, they may be harmonically restrained to a sphere to prevent drift, or (most commonly) they may be periodically replicated in all three directions. In the case of a simple vacuum it may be appropriate to use an adaptive algorithm designed for nonuniform distributions, but not in the other two cases. Treated in this article is the nonadaptive nonperiodic case.

The qualities required of the approximation depend on the type of calculation being performed, whether it is a Monte Carlo calculation, energy minimization, or molecular dynamics. The last of these commonly involves the integration of Newton's Law of Motion

$$m_i\frac{d^2}{dt^2}\vec{r}_i(t) = -\nabla_i U(\vec{r}_1(t), \vec{r}_2(t), \ldots, \vec{r}_N(t)), \quad i = 1, 2, \ldots, N,$$

where m_i is the mass of the ith atom and the potential energy U is a mostly empirical sum of contributions representing $O(N^2)$ nonbonded electrostatic and van der Waals interactions as well as $O(N)$ interactions due to covalent bonding:

$$U(\cdots) = \text{bonded energies} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\notin\chi(i)} U_{ij}^{\text{nonbonded}}(|\vec{r}_j - \vec{r}_i|).$$

The integration becomes analytically trivial if forces are approximated by a sequence of impulses:

$$m_i\frac{d^2}{dt^2}\vec{r}_i(t) = \sum_n \Delta t\,\delta(t - n\Delta t)(-\nabla_i U(\cdots)).$$

This is the popular leapfrog/Störmer/Verlet scheme, each new step consisting of half a kick followed by a drift followed by half a kick.

There are various criteria by which one might judge the merit of an approximation $\tilde{U}^{\mathrm{el}} \approx U^{\mathrm{el}}$. The most obvious is the smallness of error(s) measured in one way or another. Also of interest in some contexts is the continuity of $\tilde{U}^{\mathrm{el}}$ and its derivatives. In particular, if the force is computed as the gradient of a discontinuous potential (and we do not compute impulses at these discontinuities), the force will not be conservative. It is believed that continuity of the forces is sufficient to prevent energy drift when used with a symplectic integrator such as the leapfrog scheme. Also of interest in molecular dynamics is conservation of linear momentum and of angular momentum.

Three elements are essential to a multilevel N-body solver:

1. *Separation of length scales.* Pairwise interactions are separated into short-range interactions, which are calculated directly, and slowly varying interactions. For multigrid N-body solvers this is achieved by splitting pair potentials into short-range and slowly varying parts.
2. *Coarsening.* The slowly varying part of the energy is approximated with fewer terms. If this is done only for the source of an interaction, the result is an $\mathbb{O}(N \log N)$ Barnes–Hut type of algorithm. If this is done for both source and destination, the result is an $\mathbb{O}(N)$ Greengard–Rokhlin type of algorithm. For multigrid N-body solvers, the approximation (or interpolation) uses gridded basis functions.
3. *Hierarchy.* Steps 1 and 2 are applied recursively. (An alterna-

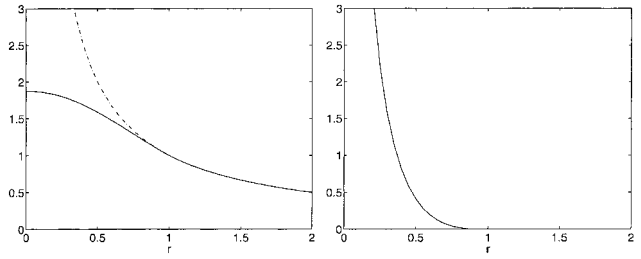

Figure 1. (a) $1/r$ and the softened $1/r$, (b) $1/r$ − (the softened $1/r$).

tive to the multilevel approach is to use an FFT to do the gridded calculation.)

Separation of Length Scales

We begin with a splitting

$$\frac{1}{r} = \left(\frac{1}{r} - g_a(r)\right) + g_a(r)$$

where $g_a(r)$ is defined so that $r^{-1} - g_a(r)$ vanishes for r beyond some cutoff $a > 0$ and $g_a(\sqrt{x^2 + y^2 + z^2})$ and its first few derivatives are slowly varying everywhere (see Fig. 1).

The slowly varying part of the energy (with coefficient omitted) is

$$\sum_{i=1}^{N} \sum_{j=1}^{N} q_i q_j g_a(|\vec{r}_j - \vec{r}_i|).$$

The other part can be calculated in $\mathbb{O}(N)$ time.

Coarsening

First, the softened potential is approximated as a function of the source $\vec{r}'$:

$$g_a(|\vec{r} - \vec{r}'|) \approx \sum_k g_a(|\vec{r} - \vec{r}_{h,k}|)\phi_k(\vec{r}')$$

where $\vec{r}_{h,k}$ are points on a 3D grid Ω_h with grid size h and ϕ_k are nodal basis functions with "local support," for example, piecewise polynomials that are identically zero on all but a small number of grid cells. A 1D piecewise cubic basis function is illustrated in Figure 2. Second, the coefficients of the basis functions are approximated as functions of the destination $\vec{r}$:

$$g_a(|\vec{r} - \vec{r}_{h,k}|) \approx \sum_m g_a(|\vec{r}_{h,m} - \vec{r}_{h,k}|)\phi_m(\vec{r}).$$

The end result is

$$g_a(|\vec{r} - \vec{r}'|) \approx \sum_k \sum_m \phi_m(\vec{r})\, g_a(|\vec{r}_{h,m} - \vec{r}_{h,k}|)\phi_k(\vec{r}')$$

—a separable approximation Σ function($\vec{r}$) · function($\vec{r}'$).

This approximation is used for the slowly varying part of the energy:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} q_i q_j g_a(|\vec{r}_j - \vec{r}_i|) \approx \sum_{i=1}^{N} \sum_{j=1}^{N} q_i q_j \sum_k \sum_m \phi_m(\vec{r}_j)\, g_a(|\vec{r}_{h,m} - \vec{r}_{h,k}|)$$

$$\times \phi_k(\vec{r}_i) = \sum_k \sum_m g_a(|\vec{r}_{h,m} - \vec{r}_{h,k}|) \sum_{i=1}^{N} q_i \phi_k(\vec{r}_i) \sum_{j=1}^{N} q_j \phi_m(\vec{r}_j).$$

Define "grid point charges"

$$q_{h,k} = \sum_{i=1}^{N} q_i \phi_k(\vec{r}_i), \tag{2}$$

and the slowly varying part of the energy becomes

$$\sum_k \sum_m q_{h,m} q_{h,k} g_a(|\vec{r}_{h,k} - \vec{r}_{h,m}|).$$

Thus,

$$\sum_{\text{particle pairs}} \quad \text{reduced to} \quad \sum_{\text{grid point pairs}}.$$

Hierarchy

What about $\Sigma_k \Sigma_m q_{h,m} q_{h,k} g_a(|\vec{r}_{h,k} - \vec{r}_{h,m}|)$? Just as we approximate a softened $1/r$ on a grid with spacing h, we approximate a further softened $g_a(r)$ on a coarser grid with spacing $2h$. For this purpose we do a splitting

$$g_a(r) = (g_a(r) - g_{2a}(r)) + g_{2a}(r),$$

in which the first part $g_a(r) - g_{2a}(r)$ vanishes for r beyond $2a$. (A better approximation for $g_{2a}(r)$ than for $g_a(r)$ is possible, because $g_{2a}(r)$ is needed only at values $r = |\vec{r}_{h,k} - \vec{r}_{h,m}|$.)

The repeated use of this idea yields an $\mathbb{O}(N)$ multilevel algorithm. Each level above the particle level has a grid, and at the grid points of each grid is computed a charge array and then a potential array. The potential for grid point $\vec{r}_{h,k}$ on the finest grid Ω_h is $\Sigma_m q_{h,m} g_a(|\vec{r}_{h,k} - \vec{r}_{h,m}|)$. (An $\mathbb{O}(N \log N)$ algorithm instead calculates potential values only at particle positions but does the calculation using the entire hierarchy of charges.)

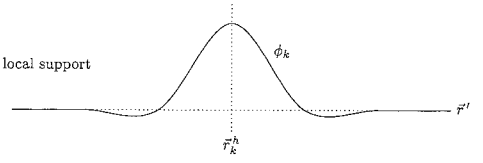

Figure 2. A piecewise cubic basis function.

Method Details

The next two subsections present possible smoothings and basis functions, and then we compare them on the basis of numerical experiments. The optimal choice of grid size and cutoff is determined analytically after that. The details of an efficient $\mathbb{O}(N)$ implementation are then deferred until the last subsection.

Smoothings

For the smoothed potential we use

$$g_a(r) = \begin{cases} \frac{1}{a}\left(\frac{15}{8} - \frac{5}{4}\left(\frac{r}{a}\right)^2 + \frac{3}{8}\left(\frac{r}{a}\right)^4\right), & r \le a, \\ \frac{1}{r}, & r \ge a. \end{cases}$$

This is C^2 continuous and enables second-order accuracy for the smoothed force (an $\mathbb{O}((h/a)^2)$ approximation error relative to the maximum smoothed force over all r) if the gridded approximation scheme is at least second-order accurate. The quadratic part of the function above is based on the Taylor expansion of $\varphi(s) = s^{-1/2}$ about $s = a^2$ where $s = r^2$. In other words, to obtain $g_a(r)$ substitute r^2 for s in

$$\varphi(s) = \begin{cases} \varphi(a^2) + (s - a^2)\varphi'(a^2) + \frac{(s - a^2)^2}{2}\varphi''(a^2), & s \le a^2, \\ s^{-1/2}, & s \ge a^2. \end{cases}$$

This type of approximation is used also by ref. 19. We experiment with two other smoothings that include one less and one more term in the Taylor expansion. The formula for a first-order Taylor expansion is

$$g_a(r) = \begin{cases} \frac{1}{a}\left(\frac{3}{2} - \frac{1}{2}\left(\frac{r}{a}\right)^2\right), & r \le a, \\ \frac{1}{r}, & r \ge a. \end{cases}$$

That for a third-order expansion is

$$g_a(r) = \begin{cases} \frac{1}{a}\left(\frac{35}{16} - \frac{35}{16}\left(\frac{r}{a}\right)^2 + \frac{21}{16}\left(\frac{r}{a}\right)^4 - \frac{5}{16}\left(\frac{r}{a}\right)^6\right), & r \le a, \\ \frac{1}{r}, & r \ge a. \end{cases}$$

All three smoothings are plotted for $a = 1$ in Figure 3.

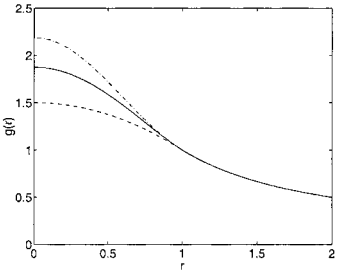

Figure 3. Smoothed potentials: dash–dash C^1, solid C^2, dash–dot C^3.

Basis Functions

Here, the key idea is to blend two quadratic interpolants to obtain a C^1 piecewise cubic interpolation function. Let $Q_k(x)$ be the quadratic polynomial interpolating a function at three consecutive grid points x_{k-1}, x_k, x_{k+1}. The blend is given by

$$\frac{x_{k+1} - x}{x_{k+1} - x_k} Q_k(x) + \frac{x - x_k}{x_{k+1} - x_k} Q_{k+1}(x), \quad x_k \le x \le x_{k+1}.$$

This defines a piecewise cubic interpolant, and it is easily verified that this is C^1 continuous. (An alternative derivation of this interpolant is to use cubic Hermite interpolation with first derivatives approximated by centered differences.[20])

Taking this approach, we obtain the 3D nodal basis functions

$$\phi_k(x, y, z) = \Phi((x - x_{h,k})/h)\Phi((y - y_{h,k})/h)\Phi((z - z_{h,k})/h)$$

where

$$\Phi(\xi) = \begin{cases} (1 - |\xi|)\left(1 + \left|\xi\right| - \frac{3}{2}\xi^2\right), & |\xi| \le 1, \\ -\frac{1}{2}(|\xi| - 1)(2 - |\xi|)^2, & 1 \le |\xi| \le 2, \\ 0, & |\xi| \ge 2. \end{cases}$$

This gives an exact approximation for quadratic but not for cubic and higher degree polynomials.

We also test other possibilities for a basis function whose general form is given as follows:

Proposition 1 *Let $\Phi(\xi)$ be a C^1 piecewise cubic with knots -2, -1, 0, 1, 2, and support $[-2, 2]$ such that $\tilde{f}'(x) \equiv f'(x)$ for quadratic polynomials* f(x) *where*

$$\tilde{f}(x) = \sum_k f(x_k)\Phi\left(\frac{x - x_k}{h}\right)$$

with $x_k = x_0 + kh$. *Then*

$$\Phi(\xi) = \begin{cases} \frac{1}{2} - \frac{1}{4}\xi^2 + \alpha\left(\frac{1}{3} - \frac{3}{2}\xi^2 + |\xi|^3\right), & |\xi| \le 1, \\ (2 - |\xi|)^2\left(\frac{1}{4} + \alpha\left(\frac{1}{6} - \frac{1}{3}|\xi|\right)\right), & 1 \le |\xi| \le 2, \\ 0, & |\xi| \ge 2, \end{cases}$$

where α is an arbitrary constant.

The proof is given in Appendix A.

Table 1. C^1 Piecewise Cubic Interpolation.

C^3 Smoothing				
h	Avg FE	Max FE	EE	Time
4.36	0.41	2.36	0.0100	2.16 (0.63)
2.77	0.20	0.86	0.0026	3.43 (1.90)
2.03	0.15	0.62	0.0016	8.89 (7.37)
C^2 Smoothing				
h	Avg FE	Max FE	EE	Time
4.36	0.29	1.32	0.0013	2.12 (0.62)
2.77	0.17	0.67	0.0024	3.37 (1.87)
2.03	0.14	0.58	0.0017	8.72 (7.22)
C^1 Smoothing				
h	Avg FE	Max FE	EE	Time
4.36	0.50	1.87	0.0060	2.12 (0.64)
2.77	0.45	1.75	0.0043	3.40 (1.92)
2.03	0.42	1.43	0.0060	8.94 (7.46)

Approximation using these general basis functions is not interpolation except for the original choice $\alpha = \frac{3}{2}$. For this case, $\tilde{f}(x) \equiv f(x)$ for quadratic $f(x)$; otherwise, the approximation is exact only for linear polynomials $f(x)$. If a C^2 approximation is wanted, this can be attained for $\alpha = \frac{1}{2}$, which gives a cubic B-spline basis function.

The other option we consider is a C^1 piecewise quintic nodal basis function with support $-3 \le \xi \le 3$:

$$\Phi(\xi) = \begin{cases} (1-\xi^2)(2-|\xi|)\left(\frac{1}{2}+\frac{1}{4}|\xi|-\frac{5}{12}\xi^2\right), & |\xi| \le 1, \\ -(|\xi|-1)(2-|\xi|)(3-|\xi|)\left(\frac{1}{6}+\frac{3}{8}|\xi|-\frac{5}{24}\xi^2\right), & 1 \le |\xi| \le 2, \\ \frac{1}{24}(|\xi|-1)(|\xi|-2)(3-|\xi|)^2(4-|\xi|), & 2 \le |\xi| \le 3, \\ 0, & |\xi| \ge 3. \end{cases}$$

This is obtained using the same technique as that given for the piecewise cubic.

Choice of Smoothing and of Basis Function

We investigate empirically the relative efficiency of different combinations of smoothing and basis functions. In particular, we study the variation of force and energy error and of running time.

The multigrid method was implemented and tested on a 60 Å cube of 6848 water molecules, which is a total of 20,544 atoms. All interactions are nonbonded. Interactions between atoms of the same molecule are not excluded. The water positions are a result of molecular dynamics equilibration.

Four different performance measures are specified: *average force error, maximum force error, energy error,* and *total CPU time.* The force errors are relative to the average force normalized by the square root of the atom's mass. More specifically, the maximum force error is given by

$$\max \text{FE} = \frac{\max_i m_i^{-1/2} |\vec{F}_i^{\text{el}} - \vec{F}_{i,d}^{\text{el}}|}{N^{-1} \sum_i m_i^{-1/2} |\vec{F}_{i,d}^{\text{el}}|},$$

the average force error is given by

$$\text{avg FE} = \frac{N^{-1} \sum_i m_i^{-1/2} |\vec{F}_i^{\text{el}} - \vec{F}_{i,d}^{\text{el}}|}{N^{-1} \sum_i m_i^{-1/2} |\vec{F}_{i,d}^{\text{el}}|},$$

and the energy error is given by

$$\text{EE} = \frac{|U^{\text{el}} - U_d^{\text{el}}|}{|U_d^{\text{el}}|},$$

where $\vec{F}_{i,d}^{\text{el}}$ and U_d^{el} denote, respectively, the force and energy obtained from a direct calculation.

The tests in this section all use a cutoff radius a of 8 Å and a total of three levels (particles and two grids). Presented in Tables 1–3 are values for h: grid size (Å), avg FE: average force error (%), max FE: maximum force error (%), EE: energy error (%), and time: CPU time (number in parenthesis is for the smooth part).

From the tabulated data, we plot CPU time vs. maximum force error for different combinations of basis and smoothing functions.

Table 2. C^2 Cubic B-Spline Approximation.

C^3 Smoothing				
h	Avg FE	Max FE	EE	Time
4.36	0.80	4.10	0.036	2.24 (0.72)
2.77	0.50	2.67	0.022	3.70 (2.18)
2.03	0.32	1.76	0.013	9.56 (8.04)
C^2 Smoothing				
h	Avg FE	Max FE	EE	Time
4.36	0.57	2.19	0.012	2.22 (0.72)
2.77	0.34	1.37	0.0044	3.65 (2.14)
2.03	0.23	0.96	0.0012	9.39 (7.89)
C^1 Smoothing				
h	Avg FE	Max FE	EE	Time
4.36	0.60	2.08	0.0038	2.22 (0.74)
2.77	0.51	1.76	0.0050	3.70 (2.23)
2.03	0.47	1.62	0.0044	9.57 (8.09)

Table 3. C^1 Piecewise Quintic Interpolation.

C^3 Smoothing				
h	Avg FE	Max FE	EE	Time
5.08	0.35	1.68	0.0079	4.08 (2.36)
3.05	0.12	0.62	0.0015	6.11 (4.45)
2.18	0.06	0.37	0.0003	13.06 (11.42)
C^2 Smoothing				
h	Avg FE	Max FE	EE	Time
5.08	0.26	1.26	0.0025	4.05 (2.36)
3.05	0.14	0.56	0.0030	6.06 (4.41)
2.18	0.11	0.41	0.0026	12.88 (11.27)
C^1 Smoothing				
h	Avg FE	Max FE	EE	Time
5.08	0.50	1.87	0.0052	3.99 (2.33)
3.05	0.44	1.67	0.0037	6.00 (4.38)
2.18	0.41	1.50	0.0025	12.77 (11.18)

Figure 4 shows the performance of various smoothing functions for a given nodal basis function. Figure 5 shows experiments with different basis functions while keeping the smoothing function fixed.

From the tables and figures, we conclude that for lower accuracy the best choice is C^1 piecewise cubic interpolation with a C^2 splitting and for higher accuracy the best choice is C^1 piecewise quintic interpolation with a C^3 splitting.

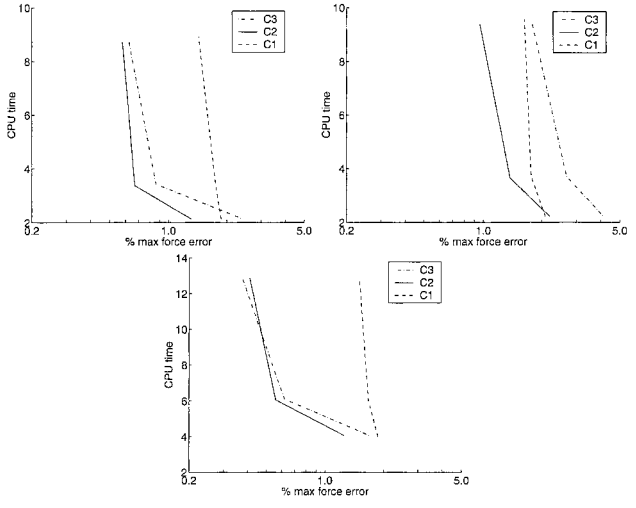

Figure 4. (a) C^1 piecewise cubic interpolation, (b) C^2 cubic B-spline approximation, and (c) C^1 piecewise quintic interpolation with different choices of splitting functions.

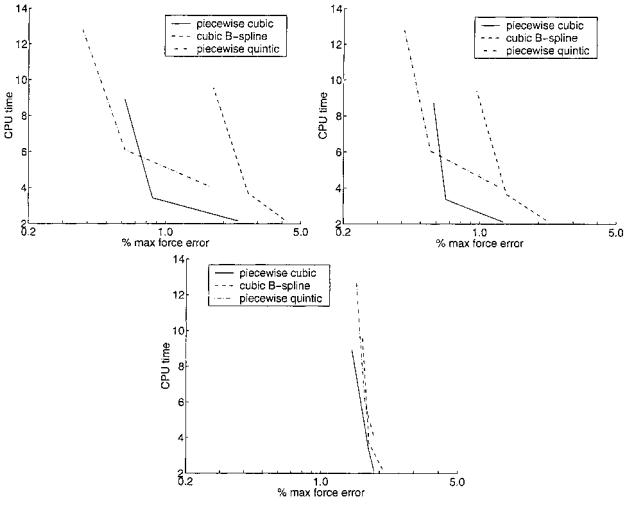

Figure 5. (a) C^3, (b) C^2, and (c) C^1 splitting functions with different choices of basis functions.

Choice of Grid Size and Cutoff

The optimal choice of grid size and cutoff can be determined analytically under realistic assumptions.

We begin by determining the cost of the computation assuming that calculating a pairwise interaction costs one unit. For an $\ell \times \ell \times \ell$ box the particle density is h_*^{-3} where $h_* = N^{-1/3}\ell$. The value h_* is a measure of the distance between nearest neighbors. The cost of calculating the short range particle–particle interactions is

$$\text{level 0 cost} = \frac{1}{2} N \cdot \frac{4}{3} \pi a^3 \cdot h_*^{-3}.$$

The cost of calculating interactions between grid points on the finest grid is

$$\text{level 1 cost} = \theta \cdot \frac{1}{2} \left(\frac{\ell}{h}\right)^3 \cdot \frac{4}{3} \pi (2a)^3 \cdot h^{-3}$$

where θ is the ratio of the cost of calculating a pairwise interaction between grid points to that for particles. The cost for each coarser level is $\frac{1}{8}$ as great as the previous level, so the total cost of grid point interactions is obtained by multiplying the level 1 cost by $1 + \frac{1}{8} + \frac{1}{64} + \cdots = \frac{8}{7}$. Neglecting the cost of transferring charges and potentials between levels, we get

$$\text{cost} = \frac{2}{3} \pi N \left(\frac{a}{h_*}\right)^3 \left(1 + \frac{64}{7} \theta \left(\frac{h_*}{h}\right)^6\right). \tag{3}$$

If we use as a rule of thumb that the two parts of the cost should be balanced, this leads to the choice

$$h = \left(\frac{64}{7} \theta\right)^{1/6} h_*,$$

Figure 6. Source in its cell and destination.

which is independent of the cutoff a. Hence, the cutoff a is the parameter to vary to obtain the desired accuracy.

A more detailed analysis yields almost the same result. For an approximation of order p to the smooth part of the force, the relative error is proportional to $(h/a)^p$. The ratio of the smooth part of the force to the total force is proportional to a^{-2}/h_*^{-2}. Hence, we can approximate the relative error as $C_p h^p h_*^2 a^{-p-2}$ for some constant C_p. Equating this to an error tolerance ϵ gives the constraint

$$C_p h^p h_*^2 a^{-p-2} = \epsilon.$$

Minimizing the cost in eq. (3) subject to this constraint gives

$$h = \left(1 + \frac{4}{p}\right)^{1/6} \left(\frac{64}{7}\,\theta\right)^{1/6} h_* \quad \text{and} \quad a = \left(h^p h_*^2 \frac{C_p}{\epsilon}\right)^{1/(p+2)}.$$

Comparison to Tree Methods

The fast multipole method is briefly sketched, and the multigrid method is compared to it both empirically and theoretically. Most of the discussion applies also to other tree methods (e.g., refs. 21–24). (The method in ref. 22 is classified as a tree method because of the way that it separates length scales).

The separation of length scales in tree methods is obtained from a decomposition of the domain into cells. A pairwise potential between two particles is regarded as slowly varying if the two parent cells are "well separated." Otherwise, the pairwise potential is short range, and it is computed directly. In a nutshell, tree methods partition particle pairs (i, j) and equate "slowly varying" with *long-range;* whereas, multigrid methods partition the potential $1/r$ and equate "slowly varying" with *smoothed.*

To explain the coarsening of long-range interactions, consider the approximation of the potential $|\vec{r} - \vec{r}'|^{-1}$ at $\vec{r}$ due to unit charge at $\vec{r}'$ where source and destination points are contained in cells that are well separated. Let $\vec{c}'$ be the center of the cell containing $\vec{r}'$, as illustrated by Figure 6, and Taylor expand $|\vec{r} - \vec{r}'|^{-1}$ about $\vec{r}' = \vec{c}'$. If we exploit the harmonicity of the function, we get

$$|\vec{r} - \vec{r}'|^{-1} = \sum_{p=0}^{\infty} \sum_{q=-p}^{p} C_p^q(\vec{r} - \vec{c}')S_p^q(\vec{r}' - \vec{c}')$$

where the pth degree term of the outer summation is a linear combination of the $2p + 1$ spherical harmonics S_p^q of degree p. This is to be contrasted with a general Taylor expansion, which has $\frac{1}{2}(p + 1)(p + 2)$ terms of degree p. For example, a seventh-degree expansion in spherical harmonics has only 64 terms instead of the 120 terms of a general Taylor expansion. To complete the coarsening, the coefficients $C_p^q(\vec{r} - \vec{c}')$ are Taylor expanded about $\vec{r} = \vec{c}$, where $\vec{c}$ is the center of the cell containing $\vec{r}$. Thus a separable approximation for the slowly varying part of the energy is created. Although tree methods can exploit harmonicity to economize on the number of terms, multigrid methods can exploit the higher accuracy of interpolation to obtain the same effect.

We compare our multigrid implementation against DPMTA using the same test problem as described earlier. The timing results have been obtained on a 360-MHz Sun Ultra-60, with each program compiled using "`cc -fast -xtarget=ultra`." The multigrid experiments show results for the C^1 piecewise cubic basis function paired with the C^2 splitting function and for the C^1 piecewise quintic basis function paired with the C^3 splitting function. Three levels are used with the finest-level grid sizes as described earlier, $h = 2.77$ and $h = 3.05$, respectively, for these two multigrid choices. As suggested by the theoretical analysis earlier, we change only the cutoff value as a control of accuracy.

The DPMTA experiments show results for two different values of the theta parameter, 0.5 and 0.75. The theta parameter is the separation ratio for the multipole acceptance criterion, which should be between 0 and 1 with optimal values in the range 0.5 to 0.75. A lower value gives better accuracy but requires greater CPU time. Four levels are used, where the number of levels refers to the number of times that the spatial domain is subdivided. A value of four levels should be sufficient for systems up to 10,000 particles. The FFT flag is set to "yes," which improves performance substantially if eight or more multipole terms are used. Accuracy in this case is controlled by changing the number of multipole terms.

The experimental results shown in Figures 7 and 8 graph the relationship between the percent relative maximum force error (our strictest measure of accuracy) and CPU time for the two multigrid and two DPMTA alternatives detailed above. Figure 7 reveals that the performance of multigrid is superior to DPMTA for lower accuracy solutions, appropriate for molecular dynamics. The performance of DPMTA is significantly better than multigrid for higher accuracy. Smoother splittings and higher degree basis functions should make multigrid more competitive for higher accuracies. Figure 8 extends these plots to compare the convergence of both methods. As expected, whenever the multigrid

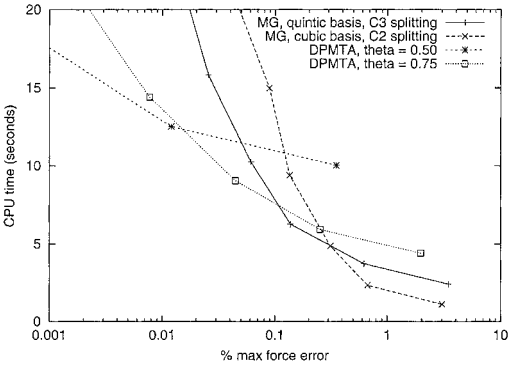

Figure 7. The cutoff a for MG takes values 5, 8, . . . , 20 and the number of terms for DPMTA takes values 4, 8, . . . , 20.

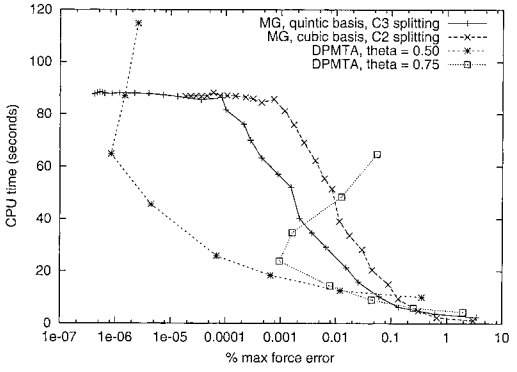

Figure 8. The cutoff a for MG takes values 5, 8, . . . , 32, 35, 40, . . . , 120 and the number of terms for DPMTA takes values 4, 8, . . . , 32.

cutoff value is large enough to encompass all pairs, the plot levels off so that the error drops to zero for larger cutoff values without additionally increasing the CPU time. The DPMTA plots show the effect of roundoff error as the number of multipole terms increases.

On balance, multigrid methods are simpler than tree methods, because they avoid the complicated operations required for spherical harmonics as well as the interaction lists required for cells. A simpler algorithm makes for easier incorporation of the force evaluation code into the dynamics code, thus eliminating the modularity overhead.

The multigrid method is more general than the fast multipole method and, for example, can be easily applied to give faster van der Waals force evaluations (to a limited extent).

Neither linear nor angular momentum can be expected to be conserved by multigrid solvers. The reason is that they use a gridding of space, which perturbs the value of the potential energy function so that it does not retain the property of being invariant under a rigid body translation and/or rotation of particle positions. It is expected that the momenta will fluctuate but not drift. The fast multipole method conserves linear momentum but not angular momentum.

An artifact arising from the fact that the approximate potential energy is not invariant under a uniform translation of all particles is the possibility of a "self" force. (This can be avoided by choosing the smoothed potential and the basis functions so that interpolation of the smoothed potential is exact, and by choosing a cutoff a that is long enough compared to the support of the basis functions.) We believe that this is no more serious than the lack of conservation of linear momentum. Also, a simulation study based on the first 170 particles (of 20,544 in total) reveals that the maximum normalized self force has a magnitude of 0.0441, which is quite small compared to the normalized average force (71.4) over all particles. Sandak[16] corrects for self-force by computing it and then subtracting it off.

As stated in the introduction of this article, tree methods calculate potential energies that are discontinuous as functions of particle positions, which results in unstable dynamics unless these discontinuities are made very small by calculating very accurate forces. The multigrid method, on the other hand, can easily be implemented so that it computes continuously varying forces, thus permitting stable time stepping for less accurate, and hence, less expensive force values.

An experiment was performed to demonstrate the destabilizing effect of the fast multipole method on dynamics, while a somewhat less accurate multigrid approximation maintains stability. The test problem is a set of 10,002 equilibrated water molecules harmonically restrained to a 42.5 Å-radius sphere. The water is based on the TIP3P model[25] without electrostatic cutoffs and with flexibility incorporated by adding bond stretching and angle bending harmonic terms (cf. ref. 26). The fast multipole method computed by DPMTA used an eight-term multipole expansion with theta = 0.75 for a reasonably good electrostatic force approximation, whereas the multigrid method used a cutoff of 8 Å and a grid size of 2.5 Å, which provided a less accurate approximation than computed by the fast multipole method. Figure 9 shows plots of the energy vs. time for 1000 fs for a step size $\Delta t = 1$ fs with the energy sampled every step. There is, during the first picosecond, a very noticeable upward energy drift for the fast multipole method indicating instability, vs. no discernible drift in energy for the more cheaply computed multigrid method over a duration of 100 picoseconds (not shown here). Experiments from ref. 11 show that DPMTA requires at least 12 multipole terms for stable dynamics, so the timing comparisons presented in Figure 7 indicate that multigrid is actually three to four times faster than DPMTA for stable dynamics. The testing was done with a molecular dynamics program written by the third author, which is compatible with NAMD but limited in features to facilitate algorithm testing.

Appropriate time steps for different interactions range from $\Delta t = 0.8$ fs for bond length stretching to more than 80 fs for electrostatics interactions (except that numerical stability requirements typically limit the largest time step to a smaller value than 80 fs). Unnecessary evaluations of slowly varying interactions can be avoided by multiple time stepping (MTS), which separates the interactions into different time scales and evaluates them at different time increments. The popular (and a good) way to do this is known as Verlet-I[27] or r-RESPA.[28] Given a partitioning $U = U^{\text{fast}} + U^{\text{slow}}$, the approximation

$$m_i \frac{d^2}{dt^2}\vec{r}_i(t) = \sum_n \delta t\,\delta(t - n\delta t)(-\nabla_i U^{\text{fast}}(\cdots)) + \sum_n \Delta t\,\delta(t - n\Delta t)(-\nabla_i U^{\text{slow}}(\cdots))$$

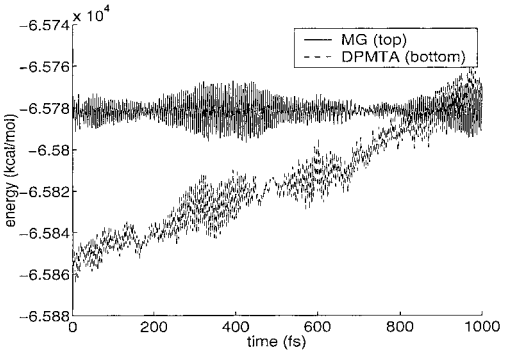

Figure 9. Energy for DPMTA and for multigrid with step size 1 fs.

allows the more numerous (long-range) slow forces to be computed less frequently.

The use of MTS with a time step fixed for each interaction is inadequate for nonbonded interactions because the time scale can vary greatly depending on the interparticle distance r. We want to vary the time step for such an interaction depending on the distance between two atoms. Therefore, for each nonbonded interaction $U_{ij}(|\vec{r}_j - \vec{r}_i|)$, we introduce an artificial splitting[28,29]

$$U_{ij}(r) = U_{ij}^{\text{fast}}(r) + U_{ij}^{\text{slow}}(r)$$

such that $U_{ij}^{\text{fast}}(r)$ vanishes for $r \geq r_{\text{cut}}$ and $U_{ij}^{\text{slow}}(r)$ is "slow" for all r. So $U_{ij}^{\text{slow}}(r)$ never requires a small time step. The effect of this is to permit a large time step whenever r exceeds the cutoff.

More than two different time steps can be used. With three different time steps the force values computed over one long time step are as follows:

$$\text{one half of } F^{\text{slow}} + \frac{1}{2}F^{\text{medium}} + \frac{1}{4}F^{\text{fast}},$$

$$\frac{1}{4}F^{\text{fast}},$$

$$\frac{1}{2}F^{\text{medium}} + \frac{1}{4}F^{\text{fast}},$$

$$\frac{1}{4}F^{\text{fast}},$$

$$\text{one half of } F^{\text{slow}} + \frac{1}{2}F^{\text{medium}} + \frac{1}{4}F^{\text{fast}}.$$

These combinations are efficiently computed by a multigrid solver.

For stable multiple time stepping it is necessary that each part of the force be continuous as a function of particle positions. Multigrid methods are easily implemented so that they continuously vary the partitioning of forces, but this is not so true for tree methods.

Algorithm Details

The double sum in eq. (1) can be formulated as the vector–matrix–vector product

$$U^{\text{el}} = \frac{1}{8\pi\epsilon_0} q^T G q, \quad G_{ij} = \begin{cases} |\vec{r}_j - \vec{r}_i|^{-1}, & i,j \text{ included} \\ 0, & \text{otherwise} \end{cases}$$

where q is an array of the particle charges and G is a symmetric matrix of values of the Green's function for the Laplacian except for the zeros due to exclusions. The special geometrical properties of G make it possible to do an approximate fast matrix–vector product:

$$Gq = \text{potentials}.$$

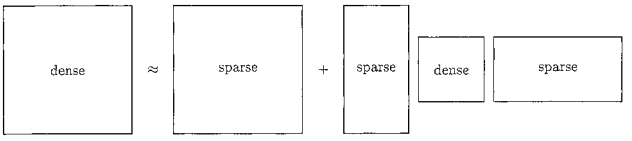

Figure 10. Matrices for a two-level approximation.

We write

$$G = \hat{G} + \tilde{G}$$

where

$$\tilde{G}_{ij} = g_a(|\vec{r}_j - \vec{r}_i|), \quad \hat{G}_{ij} = \begin{cases} f_a(|\vec{r}_j - \vec{r}_i|), & j \notin \chi(i), \\ -g_a(|\vec{r}_j - \vec{r}_i|), & j \in \chi(i), \end{cases}$$

and

$$f_a(r) = \frac{1}{r} - g_a(r).$$

The matrix $\hat{G}$ is sparse with a number of nonzeros proportional to a^3N. The matrix $\tilde{G}$ has slowly varying elements, and this property can be exploited to get a fast approximation to $\tilde{G}q$.

As described earlier, the approximation to $\tilde{G}$ is obtained by approximating the potential $g_a(|\vec{r}' - \vec{r}|)$ on the grid Ω_h. The result is

$$G \approx \hat{G} + I_h^* G_h I_*^h,$$

where

$$G_{h,km} = g_a(|\vec{r}_{h,m} - \vec{r}_{h,k}|), \quad I_{h,ik}^* = \phi_k(\vec{r}_i), \quad I_*^h = (I_h^*)^T.$$

The matrix I_*^h is sparse but G_h is dense. This is depicted in Figure 10.

The computation proceeds as follows:

$$q_h \overset{\text{def}}{=} I_*^h q \quad [\text{cf. eq. (2)}],$$

$$e_h \overset{\text{def}}{=} G_h q_h,$$

$$U^{\text{el}} \approx \frac{1}{2}\sum_i \sum_{j \notin \chi(i)} q_i q_j f_a(|\vec{r}_j - \vec{r}_i|)$$

$$- \frac{1}{2}\sum_i \sum_{j \in \chi(i)} q_i q_j g_a(|\vec{r}_j - \vec{r}_i|) + \frac{1}{2} q_h^T e_h,$$

$$\vec{F}_i^{\text{el}} \approx q_i \sum_{j \notin \chi(i)} q_j \frac{f_a'(|\vec{r}_j - \vec{r}_i|)}{|\vec{r}_j - \vec{r}_i|}(\vec{r}_j - \vec{r}_i)$$

$$- q_i \sum_{j \in \chi(i)} q_j \frac{g_a'(|\vec{r}_j - \vec{r}_i|)}{|\vec{r}_j - \vec{r}_i|} (\vec{r}_j - \vec{r}_i)$$

$$- q_i \sum_k e_{h,k} \nabla \phi_k(\vec{r}_i), \quad i = 1, 2, \ldots, N.$$

In the expressions above, note that for each of U^{el} and $\vec{F}_i^{\mathrm{el}}$ the contribution from the smooth part of the potential (the last term) depends on e_h. Hence, it is important to do the computation $e_h = G_h q_h$ efficiently. We discuss below how this is done. Also note that it is more efficient not actually to compute the product Gq at the particle level.

The *direct calculation*

$$\frac{1}{2} \sum_i \sum_{j \neq \chi(i)} q_i q_j f_a(|\vec{r}_j - \vec{r}_i|) \quad \text{and} \quad q_i \sum_{j \neq \chi(i)} q_j \frac{f_a'(|\vec{r}_j - \vec{r}_i|)}{|\vec{r}_j - \vec{r}_i|} (\vec{r}_j - \vec{r}_i),$$

$$i = 1, 2, \ldots, N$$

can be done in $\mathbb{O}(N)$ time using the grid cell + linked list technique (ref. 15, Chapter 8), termed *geometric hashing* in ref. 30. The idea is to partition the computational box into an array of grid cells each containing a subset of the particles. This is implemented in a computer program as an array of pointers to linked lists of particle indices, one linked list for each cell. This makes it easy to avoid processing pairs of particles in grid cells separated by a distance of at least a.

The matrix G_h has special structure that can be exploited, for example, by using a 3D FFT to form the product $e_h = G_h q_h$.[31] An alternative fast way to multiply by G_h is to use a multilevel method. Just as we approximate the smooth part of G on a grid with spacing h, we approximate the smooth part of G_h on a coarser grid with spacing $2h$. The hierarchical structure of the multigrid algorithm lends itself better to a parallel implementation than a method based on the FFT and accommodates a multiplicity of time steps for the longer range electrostatics.

It is convenient to change notation and use superscripts 1, 2, 3, ... to denote quantities on grids $\Omega^1 = \Omega_h$, $\Omega^2 = \Omega_{2h}$, $\Omega^3 = \Omega_{4h}$, ... Based on the splitting $g_a(r) = (g_a(r) - g_{2a}(r)) + g_{2a}(r)$, write

$$G^1 = \hat{G}^1 + \tilde{G}^1$$

where $\hat{G}^1$ is sparse and $\tilde{G}^1$ is sufficiently smooth that it can be approximated on a coarser grid Ω^2. The result is then

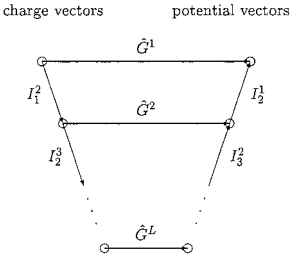

Figure 11. Multiple grid V-cycle.

Table 4. Smooth Part Only: C^1 Piecewise Cubic With C^2 Smoothing.

h	Avg FE	Max FE	EE
4.36	32.2	148.7	0.014
3.39	21.9	84.9	0.0034
2.77	18.5	78.2	0.0018
2.03	15.7	63.7	0.0075
1.33	9.8	32.7	0.0008
0.98	5.7	23.2	0.0022

$$G^1 \approx \hat{G}^1 + I_2^1 G^2 I_1^2,$$

where the matrix I_1^2 is sparse but G^2 is dense. The matrix I_1^2 is a tabulation of grid Ω^2 basis functions on grid Ω^1. This is repeated at higher levels in going from a finer grid to a coarser grid:

$$G^l \approx \hat{G}^l + I_{l+1}^l G^{l+1} I_l^{l+1}, \quad l = 1, 2, \ldots, L - 1,$$

where L is the number of grid levels. Thus, a dense matrix of N_l^2 elements is reduced to one of $N_{l+1}^2 = \frac{1}{64} N_l^2$ elements. The actual computation is as follows:

$$q^{l+1} = I_l^{l+1} q^l \quad l = 1, 2, \ldots, L - 1,$$

$$e^L = G^L q^L,$$

$$e^l \approx \hat{G}^l q^l + I_{l+1}^l e^{l+1}, \quad l = L - 1, L - 2, \ldots, 1.$$

This can be represented as a V-cycle with "rungs" as shown in Figure 11. In this diagram each circle represents a vector of values, each horizontal arrow represents multiplication by a sparse matrix, each diagonal arrow represents multiplication by a very sparse matrix, and the juncture of two arrows is the sum of two vectors.

The choice of the piecewise cubic basis function proposed earlier leads to $4 \times 4 \times 4$ stencils for transfer operations. More generally, the size of the stencils is proportional to the order p of the approximation, and the cost of applying I_*^1 and I_1^* is $O(p^3 N)$. The coefficients of operators I_l^{l+1} and I_{l+1}^l are independent of l and can be precomputed. Also, the cost of applying these is only $O(pN)$.

Because

$$g_{2a}(\sqrt{(2ih)^2 + (2jh)^2 + (2kh)^2}) = \frac{1}{2} g_a(\sqrt{(ih)^2 + (jh)^2 + (kh)^2}),$$

Table 5. Accuracy of Simple Cutoff With Various Radii vs. Multigrid.

a	Avg FE	Max FE	EE
8	4.04	14.23	1.98
10	3.16	11.90	1.58
12	2.63	9.53	2.81
8 (MG)	0.17	0.67	0.0024

Table 6. Simulation Results for Uniformly Charged Particles.

h	Avg FE	Max FE	EE
4.36	0.77	5.09	0.012
3.39	0.43	3.10	0.008
2.77	0.30	2.02	0.002
2.03	0.16	1.09	0.0007
1.33	0.06	0.45	0.0001

the $\hat{G}^l$ are independent of l except for a scale factor and can be precomputed; in particular, one can tabulate $g_a(h\sqrt{i^2 + j^2 + k^2})$ for all required values of $i^2 + j^2 + k^2$. Also, the coefficients of G^L can be precomputed.

It is of interest to consider how the mesh is generated for each level of computation. Let ℓ be the length of a cube containing all particles. Suppose that h_d is the desired grid size for the finest grid. Choose the finest grid to have $M_1 \approx \ell/h_d + 2$ subdivisions in each direction where $M_1 - 4$ is a modest integer times some power of 2. Then, the grid spacing for the finest grid is chosen to be $h = \ell/(M_1 - 2)$. The reason for subtracting two is that the computational box is one grid spacing bigger in each direction than the original cube—if piecewise cubic basis functions are used for interpolation. (We would subtract 4 for quintics.) Then, for each coarser level $l + 1$, $M_{l+1} = \frac{1}{2}M_l + 2$, and $h_{l+1} = 2h_l$. If, for example, there are 24 subdivisions and three levels, then the number of subdivisions for each level are 24, 14, and 9, respectively. Because of the padding of coarser grids, it is counterproductive to go to too coarse a grid. Also, an $O(N)$ operation count requires only that $M_L = O(M_1^{1/2})$.

Additional details are given in Appendix B.

Additional Experimental Analysis

Here we examine the (interpolation) errors in the smooth part of the forces relative not to the total forces but only to the smooth parts. We use C^1 piecewise cubic interpolation with C^2 splitting. The cutoff radius is 7.5 Å, and the depth of the grid hierarchy is 1. The results are in Table 4. The error is larger than would be expected from the previous results (and indicates a need for a more penetrating error analysis).

These unexpectedly poor results suggest a comparison with the use of simple cutoffs. Table 5 shows the accuracy of simple cutoff for cutoff radii 8, 10, and 12 Å vs. multigrid. The grid size is fixed at 2.77. It is evident that we do considerably better in terms of accuracy using the multigrid method.

For further insight, we also include results pertaining to the smooth part of the potential for the case where all atoms are positively charged (i.e., oxygen atoms carry a positive charge). The numbers in Table 6 indicate a much better error convergence. The convergence here is second-order for forces and fourth-order for energy. The discrepancy between the two sets of results is attributed to massive cancellation.

Acknowledgement

This work was completed while R. D. Skeel was visiting the Mathematics Department, University of California, San Diego.

Appendix A: Proof of the Proposition

Choose $h = 1$ and grid points $\ldots, -\frac{3}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{3}{2}, \ldots$, and consider the interval $-\frac{1}{2} \le x \le \frac{1}{2}$:

$$\tilde{f}(x) = f\left(-\frac{3}{2}\right)\Phi\left(x + \frac{3}{2}\right) + f\left(-\frac{1}{2}\right)\Phi\left(x + \frac{1}{2}\right) + f\left(\frac{1}{2}\right)\Phi\left(x - \frac{1}{2}\right) + f\left(\frac{3}{2}\right)\Phi\left(x - \frac{3}{2}\right). \quad (4)$$

We can also express this as a linear combination of averaged centered differences of $f(x)$ at $x = 0$ of orders 0, 1, 2, 3. Exactness of the first derivative for quadratic $f(x)$ implies

$$\tilde{f}'(x) = \left(f\left(\frac{1}{2}\right) - f\left(-\left(\frac{1}{2}\right)\right) + \frac{1}{2}x\left(f\left(\frac{3}{2}\right) - f\left(\frac{1}{2}\right) - f\left(-\frac{1}{2}\right) + f\left(-\frac{3}{2}\right)\right) + Q(x)\left(f\left(\frac{3}{2}\right) - 3f\left(\frac{1}{2}\right) + 3f\left(-\frac{1}{2}\right) - f\left(-\frac{3}{2}\right)\right) \quad (5)$$

for some arbitrary quadratic $Q(x)$. For this to be part of a continuous interpolant, the contribution of $f(-\frac{3}{2})$ to $\tilde{f}(x)$ should vanish at $x = \frac{1}{2}$ and that of $f(\frac{3}{2})$ should vanish at $x = -\frac{1}{2}$:

$$\left(\frac{1}{2}x - Q(x)\right)\Big|_{x=1/2} = 0, \quad \left(\frac{1}{2}x + Q(x)\right)\Big|_{x=-1/2} = 0.$$

This implies

$$Q(x) = \frac{1}{4} + \alpha\left(x^2 - \frac{1}{4}\right)$$

for some arbitrary constant α. Comparing eqs. (4) and (5) gives

$$\Phi'(x) = \begin{cases} \frac{1}{2}x + 1 + \alpha(x+1)(x+2), & -2 \le x \le -1, \\ -\frac{1}{2}x - 3\alpha x(x+1), & -1 \le x \le 0, \\ -\frac{1}{2}x + 3\alpha x(x-1), & 0 \le x \le 1, \\ \frac{1}{2}x - 1 - \alpha(x-1)(x-2), & 1 \le x \le 2, \\ 0, & \text{otherwise.} \end{cases}$$

Integrating this yields the stated result.

Appendix B: Additional Algorithmic Details

We revert back to the h subscripting and give details for the operations for the grid Ω_{2h}:

$$q_{2h} = I_h^{2h} q_h,$$

$$e_{2h} = \hat{G}_{2h} q_{2h} + \text{contribution from } \Omega_{4h},$$

$$e_h = \hat{G}_h q_h + I_{2h}^h e_{2h}.$$

Operations on coarser grids are just the same except for a smaller range of indices.

The transfer of charge from Ω_h to Ω_{2h} is computed as

$$q_{2h,k} = ((I_{2h}^h)^T q_h)_k = \sum_m I_{2h,mk}^h q_{h,m} = \sum_m \phi_{2h,k}(\vec{r}_{h,m}) q_{h,m}.$$

Assume for notational convenience that grid indices k are triplets $k = (k_x, k_y, k_z)$ (so that summations over k are triple summations) and that $\vec{r}_{2h,k} = \vec{r}_{h,2k}$. Expressing $m = 2k + \kappa$, we get

$$q_{2h,k} = \sum_\kappa \phi_{2h,k}(\vec{r}_{2h,k} + h\kappa) q_{h,2k+\kappa}$$

$$= \sum_{\kappa_x} \Phi\left(\frac{1}{2}\kappa_x\right) \sum_{\kappa_y} \Phi\left(\frac{1}{2}\kappa_y\right) \sum_{\kappa_z} \Phi\left(\frac{1}{2}\kappa_z\right) q_{h,2k_x+\kappa_x,2k_y+\kappa_y,2k_z+\kappa_z},$$

where the indices κ_x, κ_y, and κ_z each range from $-\frac{1}{2}p - 1$ to $\frac{1}{2}p + 1$.

The direct calculation on Ω_{2h} is

$$(\hat{G}_{2h} q_{2h})_k = \sum_m \hat{G}_{2h,km} q_{2h,m}$$

$$= \sum_\kappa \hat{G}_{2h,k,k+\kappa} q_{2h,k+\kappa}$$

$$= \sum_\kappa (g_{2a}(|2h\kappa|) - g_{4a}(|2h\kappa|)) q_{2h,k+\kappa}$$

$$= \sum_\kappa \left(\frac{1}{2} g_a(h|\kappa|) - \frac{1}{4} g_a\left(\frac{h}{2}\left|\kappa\right|\right)\right) q_{2h,k+\kappa},$$

where the summation for κ ranges over triplets κ such that $|\kappa| < 2a/h$.

Finally, the transfer of potential from Ω_{2h} to Ω_h is given by

$$(I_{2h}^h e_{2h})_k = \sum_m I_{2h,km}^h e_{2h,m} = \sum_m \phi_{2h,m}(\vec{r}_{h,k}) e_{2h,m}.$$

This is most easily implemented as an outer loop over each Ω_{2h} index m and an inner loop on k:

add $\phi_{2h,m}(\vec{r}_{h,k}) e_{2h,m}$ to $e_{h,k}$ for all $\vec{r}_{h,k}$ "near" $\vec{r}_{2h,m}$.

Changing to $k = 2m + \kappa$ this becomes

add $\phi_{2h,m}(\vec{r}_{2h,m} + h\kappa) e_{2h,m}$ to $e_{h,2m+\kappa}$ for all "small" κ.

The term that we add simplifies to

$$\Phi\left(\frac{1}{2}\kappa_x\right) \Phi\left(\frac{1}{2}\kappa_y\right) \Phi\left(\frac{1}{2}\kappa_z\right) e_{2h,m}$$

where the indices κ_x, κ_y, κ_z each range from $-\frac{1}{2}p - 1$ to $\frac{1}{2}p + 1$.

References

1. Greengard, L.; Rokhlin, V. J Comput Phys 1987, 73, 325.
2. Appel, A. W. SIAM J Sci Statist Comput 1985, 6, 85.
3. Barnes, J.; Hut, P. Nature 1986, 324, 446.
4. Rankin, W.; Board, J. 4th IEEE Symposium on High Performance Distributed Computing, Proceedings, IEEE Computer Society, 1995, p. 17.
5. URL: http://www.ee.duke.edu/Research/SciComp/Docs/Dpmta.
6. URL: http://www.ks.uiuc.edu/Research/namd.
7. Kalé, L.; Skeel, R.; Brunner, R.; Bhandarkar, M.; Gursoy, A.; Krawetz, N.; Phillips, J.; Shinozaki, A.; Varadarajan, K.; Schulten, K. J Comput Phys 1999, 151, 283.
8. Warming, R. F.; Hyett, B. J. J Comput Phys 1974, 14, 159.
9. Griffiths, D. F.; Sanz–Serna, J. M. SIAM J Sci Statist Comput 1986, 7, 994.
10. Skeel, R. D.; Hardy, D. J. Practical Construction of Modified Hamiltonians. SIAM J Sci Comput 2001, 23, 1172.
11. Bishop, T.; Skeel, R. D.; Schulten, K. J Comput Chem 1997, 18, 1785.
12. Sagui, C.; Darden, T. J Chem Phys 2001, 114, 6578.
13. Brandt, A.; Lubrecht, A. A. J Comput Phys 1990, 90, 348.
14. Skeel, R. D. In Multiscale Computational Methods in Chemistry and Physics, Brandt, A.; Bernholc, J.; Binder, K., Eds.; volume 177 of NATO Science Series: Series III Computer and Systems Sciences; IOS Press: Amsterdam, 2001, p. 3.
15. Hockney, R. W.; Eastwood, J. W. Computer Simulation Using Particles; McGraw-Hill: New York, 1981.
16. Sandak, B. J Comput Chem 2001, 22, 717.
17. Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pederson, L. J Chem Phys 1995, 103, 8577.
18. Cheng, H.; Greengard, L.; Rokhlin, V. J Comput Phys 1999, 155, 468.
19. Brandt, A.; Venner, C. H. SIAM J Sci Comput 1998, 19, 468.
20. Kahaner, D.; Moler, C.; Nash, S. Numerical Methods and Software; Prentice-Hall; Englewood Cliffs, NJ, 1989.
21. Duan, Z.-H.; Krasny, R. J Comput Chem 2001, 22, 184.
22. Zaslavsky, L. Y.; Schlick, T. Appl Math Comp 1998, 97, 237.
23. Anderson, C. R. SIAM J Sci Stat Comp 1992, 13, 923.
24. Makino, J. J Comput Phys 1999, 151, 910.
25. Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. J Chem Phys 1983, 79, 926.
26. Leach, A. R. Molecular Modelling: Principles and Applications; Addison-Wesley Longman: Reading, MA, 1996.
27. Grubmüller, H.; Heller, H.; Windemuth, A.; Schulten, K. Mol Simulat 1991, 6, 121.
28. Tuckerman, M.; Berne, B. J.; Martyna, G. J. J Chem Phys 1992, 97, 1990.
29. Skeel, R. D.; Biesiadecki, J. J. Ann Numer Math 1994, 1, 191.
30. Fox, G. C.; Johnson, M. A.; Lyzenga, G. A.; Otto, S. W.; Salmon, J. K.; Walker, D. W. Solving Problems on Concurrent Processors, Volume I: General Techniques and Regular Problems; Prentice-Hall: Englewood Cliffs, NJ, 1988.
31. Brooks, B. April 2000. Personal communication.