Parallel multigrid summation for the N-body problem

Jesús A. Izaguirre *, Scott S. Hampton

Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA

Thierry Matthey

Parallab, Bergen Center for Computational Science, University of Bergen, Norway

Abstract

An $\Theta(n)$ parallel multigrid summation method (MG) for the N-body problem is presented. The method was originally devised for vacuum boundary conditions. Here it is extended to periodic boundary conditions and implemented in parallel using force decomposition and MPI. MG is based on a hierarchical decomposition of computational kernels on multiple grids. For low accuracy calculations, appropriate for molecular dynamics, a sequential implementation is as fast or faster than Particle Mesh Ewald (PME). Our parallel implementation is more scalable than PME. The method can be combined with multiple time stepping integrators to produce a powerful simulation protocol for simulation of biological molecules and other materials. The parallel implementation is tested on both a Linux cluster with Myrinet interconnect and a shared memory computer. It is available as open-source at `http://protomol.sourceforge.net`. An auxiliary tool allows the automatic selection of optimal parameters for MG, and is available at `http://mdsimaid.cse.nd.edu`.

Key words: parallel N-body solvers, multigrid summation, fast electrostatic solvers, particle mesh-Ewald

* Corresponding author.
Email addresses: `izaguirr@cse.nd.edu` (Jesús A. Izaguirre), `shampton@cse.nd.edu` (Scott S. Hampton), `matthey@ii.uib.no` (Thierry Matthey).

1 Introduction

We present a method for fast parallel evaluation of the N-body problem. The core of this problem is the sum of slowly decaying pair-wise interactions of N particles. The motion of planets and galaxies, the folding of proteins, and the determination of the electronic structures of materials are examples of applications that need to solve the N-body problem.

Often, the N-body problem is solved by using a cutoff at some distance from the origin. However, in some cases the use of a cutoff affects the results of interest significantly. For example, in gravitational problems, cutoffs may exclude important gravitational effects of large bodies. The same is true of molecular systems, particularly those of biological relevance such as DNA and proteins: Simulations with cutoff lead to artifacts in simulations of peptides and proteins, and fail for interfacial and membrane simulations [1].

We consider isolated and periodically replicated systems. For isolated systems, a straightforward computation of all pairwise interactions leads to an $\Theta(N^2)$ algorithm. A hierarchical decomposition of space leads to the Barnes-Hut [2] $\Theta(N \log N)$ algorithm. A clever interpolation of harmonics leads to the $\Theta(N)$ fast multipole method (FMM) [3]. For periodically replicated systems, a decomposition into real and Fourier terms leads to the $\Theta(N^{3/2})$ Ewald algorithm. The solution of the Fourier part on a grid using FFT leads to the $\Theta(N \log N)$ Particle Mesh Ewald (PME) [4, 5] algorithm. Extensions of the $\Theta(N)$ FMM exist for periodic systems.

We extend an $\Theta(N)$ multigrid summation technique (MG) [6–9] to periodic boundary conditions and present a scalable and portable parallelization of the method. We have implemented and tested MG in the software framework ProtoMol [10], which is available as an open source project [1]. Tests on atomic and molecular systems ranging from 1,000 to 1,000,000 atoms show that MG scales linearly with system size and compares favorably to a highly optimized PME implementation in NAMD 2.5 [11]. Parallel tests on a benchmark apoA-I solvated protein of roughly 92,000 atoms show the superior scalability of MG vs. PME. Parallel tests on ionic systems of up to 1,000,000 atoms [12] show the potential for massive simulations enabled by the multigrid summation technique.

MG is highly parallelizable. Our implementation uses MPI, and is efficient when tested in both shared and distributed memory computers. Concretely, on an IBM p690 Regatta turbo, MG exhibits a parallel efficiency of over 90% when using 16 processors, whereas PME using the highly optimized parallel library

[1] `http://protomol.sourceforge.net`

FFTW[2] has a parallel efficiency of only a little over 60%. Similarly, on a Beowulf cluster using Myrinet, a force decomposition version of MG has a 50% parallel efficiency when using 66 processors, whereas a force decomposition version of PME has about 10% using 55 processors. The method is not as scalable for smaller number of processors as the spatial decomposition version of PME in NAMD 2.5, but in our tests it continues to scale, whereas NAMD 2.5 shows a slow down using 66 processors.

To make MG easier to use, a recommender system called MDSimAid, chooses optimal parameters, based on governing rules and run-time fine tuning[3] [13].

The rest of the paper is organized as follows: Sections 1.1 and 1.2 give a mathematical description of the N-body problem; Section 2 describes the MG method; Section 3 gives the parallel implementation of MG; Section 4 reports computational experiments on MG and other fast electrostatic solvers; Section 5 discusses related work; and Section 6 contains a discussion of the results.

1.1 N-body problem for isolated systems

The N-body problem for an isolated molecular system with N particles consists of computing the electrostatic energy

$$U^{\text{electrostatic}}(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j \notin \chi(i)} \frac{q_i q_j}{4\pi\varepsilon_0 \left|\vec{r}_j - \vec{r}_i\right|}, \tag{1}$$

where the position and partial charge of the i^{th} atom are $\vec{r}_i$ and q_i, and the dielectric coefficient is ε_0. $\chi(i)$ is a set of pairwise interactions that are excluded for the i^{th} atom.

The gradient of this potential energy is the force. This is used to compute the molecular dynamics using Newton's equations of motion, $\vec{F} = m\vec{a}$. The core of the computation is spent in evaluating the long-range electrostatic forces for many time steps.

1.2 N-body problem for replicated systems

When modeling molecular systems it is common to consider the system to be infinitely replicated in space (periodic boundary conditions, or PBC). One advantage of PBC is that the system size can be reduced, when compared to

[2] `http://fftw.org`
[3] `http://mdsimaid.cse.nd.edu`

systems with other boundary conditions. Periodicity may introduce spurious results and requires some careful study. In liquid simulations, periodicity effects are minimal. Charged or polar systems in high dielectric medium show minimal periodicity effects as well. More careful studies of low dielectric and biomolecular systems are still needed.

In the case of a replicated system under periodic boundary conditions, the problem to be solved is

$$U^{\text{electrostatic}}(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j \notin \chi(i)} \sum_{\vec{m} \in \mathbb{Z}^3}{}' \frac{q_i q_j}{4\pi\varepsilon_0 \left| \vec{r}_j - \vec{r}_i + \vec{m}L \right|}, \qquad (2)$$

where the sum is over all periodic cells with index $\vec{m}$, with self-interactions excluded, and L is the length of a periodic box cell with dimensions $L \times L \times L$. This is a conditionally convergent sum; a physical interpretation of the process is given in [14].

Ewald [15] splits this sum into two rapidly convergent sums, cf. Eq. (8). There are real and reciprocal space parts, the latter in the Fourier domain. Ewald chooses a softening function

$$g(r_{ij,m} \equiv ||\vec{r}_j - \vec{r}_i + \vec{m}||) = \frac{\operatorname{erf}(\alpha r_{ij,m})}{r_{ij,m}} = \frac{2}{\sqrt{\pi} r_{ij,m}} \int_0^{\alpha r_{ij,m}} \exp(-s^2) \mathrm{d}s. \qquad (3)$$

The real part is short-ranged,

$$\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} q_i q_j \sum_{\vec{n}}{}' \left(\frac{1}{r_{ij,n}} - \frac{\operatorname{erf}(\alpha r_{ij,n})}{r_{ij,n}} \right), \qquad (4)$$

and is solved directly, whereas the reciprocal space part is solved by a Fourier series:

$$\frac{1}{2\pi\varepsilon_0 L^3} \sum_{\vec{m} \neq 0} \frac{1}{\vec{m}^2} \exp\left(\frac{-\pi^2 \vec{m}^2}{\alpha^2} \right) \hat{\rho}(\vec{m}) \hat{\rho}(-\vec{m}), \qquad (5)$$

where

$$\hat{\rho}(\vec{m}) = \sum_j q_j \exp(2\pi i \vec{m} \cdot \vec{r}_j). \qquad (6)$$

The parameter α controls how much computation is done in the real space part. The optimal value for Ewald summation is obtained by varying the cutoff with the square root of the periodic cell length. In this case, the complexity of the Ewald sum is $\Theta(N^{3/2})$ [16, 17].

There are two approaches to solving the Ewald summation: as the solution of

Poisson's equations in PBC, and as a large but finite array of copies of the simulation cell immersed in a dielectric medium.

The particle-mesh (PM) method of Hockney and Eastwood [18] evaluates the Coulombic potential at particles by interpolating the charges to a regular grid. Then the FFT is used to obtain a solution to a discretized Poisson's equation. In PM the interactions between nearby particles are poorly represented. The particle-particle particle-mesh (P3M) improves upon PM by splitting the contributions into short-range and long-range, and solving the short-range part directly.

The particle mesh Ewald (PME) method [4] chooses the splitting parameter α such that the short-ranged part is $\Theta(N)$, and interpolates the Fourier series onto a mesh, thus allowing the use of $\Theta(N \log N)$ FFT for approximating the smooth part, Eq. (6):

$$\hat{\rho}(\vec{m}) \approx \sum_n \exp(2\pi i \vec{m} \cdot \vec{r}_n^{\,h}) \sum_j \phi_n(\vec{r}_j) q_j, \tag{7}$$

where $\vec{r}_n^{\,h}$ are grid positions, and $\phi(\vec{r})q$ does an adjoint interpolation (also called anterpolation) of the particle charges to the grid.

2 Multilevel summation methods

Multilevel methods recursively separate the length scales present in the problem and approximate the slower or longer-range scales with a coarser representation to achieve fast computation. There are two kinds of multilevel methods: (i) cell methods, such as the fast multipole algorithm [3], based on an oct-tree decomposition of space, and (ii) grid methods, such as the Brandt-Lubrecht fast summation method [6,9], which is based on a multiple grid hierarchy. The steps of these methods are as follows:

(1) *Separation of length scales* into short-range and smooth. The short range calculations are computed directly. In cell (or tree) methods, two cells are considered to be smooth or slowly-varying if they are well separated according to some mathematical criterion. A multiple grid method splits the computational kernel into short-range and smooth parts at a cutoff r_c using switching functions, for example:

$$\frac{1}{r} = \underbrace{\left(\frac{1}{r} - g^0_{\text{smooth}}(r)\right)}_{g^0_{\text{local}},\ \text{short-ranged}} + \underbrace{g^0_{\text{smooth}}(r)}_{\text{smooth}}, \tag{8}$$

where $g^0_{\text{local}} \equiv \left(\frac{1}{r} - g^0_{\text{smooth}}(r)\right)$ vanishes for $r > r_c$.

(2) *Coarsening.* This involves approximating the smooth part with a coarser grid. Among cell-methods, Barnes-Hut [2], which is an $\Theta(N \log N)$ algorithm, approximates at the source only, while the multipole $\Theta(N)$ algorithms approximate at both the source and destination. The source is the atom that we are calculating the interactions for, while the destination refers to each interacting atom, cf. Eq. (9). Coarsening normally involves interpolation: multipole algorithms use a truncated Taylor interpolation that exploits harmonicity of the $1/r$ potential to permit the use of an inexpensive basis of spherical harmonic potentials. Coarsening for multigrid algorithms involves interpolation using basis functions on the grids. With the latter approach, it is easy to produce forces and energies that are continuous as a function of positions, as in [9], whereas it is more difficult for multipole methods, cf. [19]. Continuous forces (derivatives of the potential energy) are necessary for stability of molecular dynamics (MD) integrators.

(3) *Hierarchical decomposition,* which involves a recursive separation of the length scales and coarsening of the problem at each scale.

In what follows, each one of these steps is described in more detail for the MG method. Henceforth it is assumed that vacuum boundary conditions are being used. Our presentation of MG follows [20, pp. 6-31] and [9] using the notation of [21]. Our changes to make the algorithm work with PBC are summarized in Section 2.5.

2.1 Separation of length scales

The separation of the length scales is done with a switching function that brings the computational kernel smoothly to zero at a cutoff distance r_c. These switching functions have varying degrees of smoothness and computational cost, cf. Figure 1 for examples.

2.2 Coarsening

The softened kernel g^0_{smooth} is approximated at the source $\vec{r}'$:

$$g^0_{\text{smooth}}(||\vec{r} - \vec{r}'||) \approx \sum_k g^0_{\text{smooth}}(||\vec{r} - \vec{r}_{h,k}||)\phi_k(\vec{r}'), \tag{9}$$

where $\vec{r}_{h,k}$ are points on a 3-d grid with grid point separation h, and ϕ_k are piecewise polynomials with local support on a few grid cells. The coefficients of the basis functions $g^0_{\text{smooth}}(||\vec{r} - \vec{r}_{h,k}||)$ are approximated at the destination $\vec{r}$:

$$g^0_{\text{smooth}}(||\vec{r}-\vec{r}_{h,k}||) \approx \sum_m g^0_{\text{smooth}}(\|\vec{r}_{h,m}-\vec{r}_{h,k}\|)\phi_m(\vec{r}), \tag{10}$$

resulting in a double sum over the grid cells necessary for the interpolation using ϕ_m and ϕ_k.

$$g^0_{\text{smooth}}(||\vec{r}-\vec{r}'||) \approx \sum_k \sum_m \phi_m(\vec{r}) g^0_{\text{smooth}}(\|\vec{r}_{h,m}-\vec{r}_{h,k}\|)\phi_k(\vec{r}'). \tag{11}$$

The charges at the grid point are defined as

$$q_{h,k} = \sum_{i=1}^{N} q_i \phi_k(\vec{r}_i). \tag{12}$$

Using these definitions, the smooth part of the electrostatic energy can be written simply as

$$\sum_k \sum_m q_{h,m} q_{h,k} \left\|\vec{r}_{h,m}-\vec{r}_{h,k}\right\|. \tag{13}$$

The sum over particle pairs has been reduced to a sum over grid point pairs.

2.3 Hierarchical decomposition

The *smoothed kernel* at the particle level, $g^0_{\text{smooth}}(r)$, is approximated on a fine level-1 grid, as $g^1_{\text{smooth}}(r)$. The superscript indicates the grid level. We redo the splitting of Eq. (8) for $g^1_{\text{smooth}}(r)$:

$$g^1_{\text{smooth}}(r) = \underbrace{\left(g^1_{\text{smooth}}(r) - g^2_{\text{smooth}}(r)\right)}_{g^1_{\text{local}}} + g^2_{\text{smooth}}(r), \tag{14}$$

where $g^1_{\text{local}} \equiv g^1_{\text{smooth}}(r) - g^2_{\text{smooth}}(r)$ is zero for $r > 2r_{\text{c}}$. This process can be applied recursively: in general, for grid level $k \in \{1, 2, \ldots, l\}$, the smoothed kernels are defined as

$$g^k_{\text{smooth}}(\vec{r}_i, \vec{r}_j) = \begin{cases} g_{s_k}(||\vec{r}_j - \vec{r}_i||) : ||\vec{r}_j - \vec{r}_i|| \leqslant s_k \\ 1/||\vec{r}_j - \vec{r}_i|| \quad : \text{otherwise.} \end{cases}. \tag{15}$$

$g_{s_k}(||\vec{r}_j - \vec{r}_i||)$ is a softening function, a polynomial parameterized by s_k, such that $g_{s_k}(s_k) = 1/s_k$. The softening distance at level k is given by $s_k = 2^{k-1} r_{\text{c}}$, cf. Figure 1. $g^k_{\text{smooth}}(||\vec{r}_j - \vec{r}_i||)$ is a piecewise polynomial with continuity C^n, e.g., a piecewise cubic or a quintic.

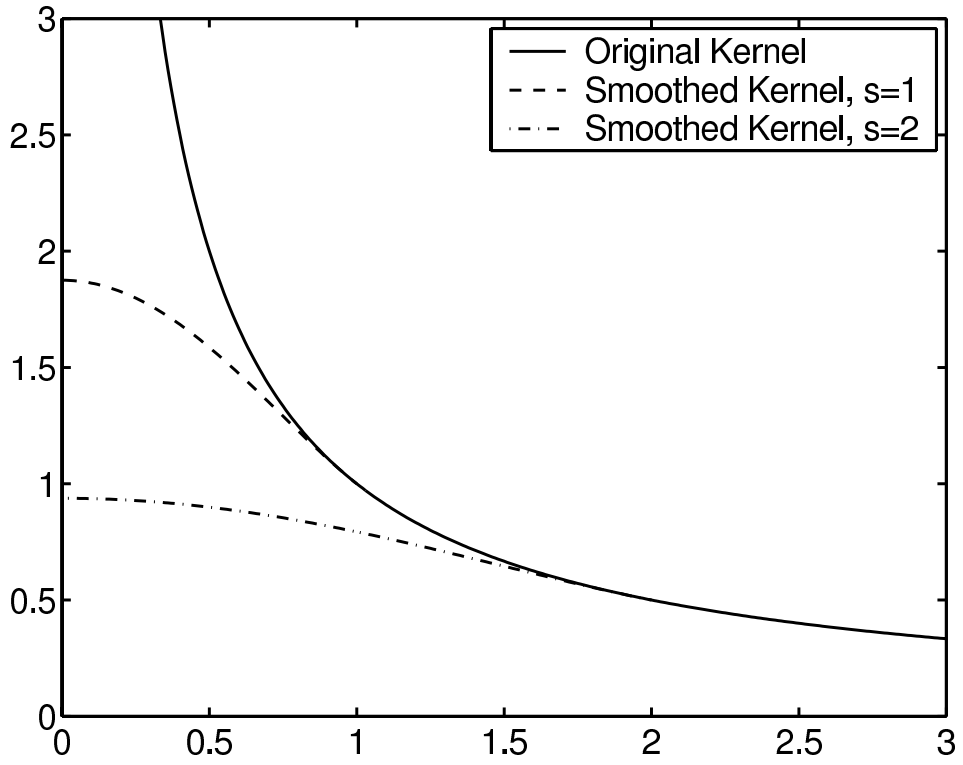


Fig. 1. A plot of the original kernel r^{-1} and two smoothed kernels $G^k_{\text{smooth}}(\vec{r}_i, \vec{r}_j)$ with softening distances $s = 1$ and $s = 2$. $G_s(r) = \frac{1}{s}\left(\frac{15}{8} - \frac{5}{4}\left(\frac{r}{s}\right)^2 + \frac{3}{8}\left(\frac{r}{s}\right)^4\right)$, $G_s(r)$ is C^2-continuous.

The local part of the smoothed kernels is defined as follows:

$$g^0_{\text{local}}(\vec{r}_i, \vec{r}_j) = \begin{cases} 1/||\vec{r}_j - \vec{r}_i|| - g^1_{\text{smooth}}(||\vec{r}_j - \vec{r}_i||) : ||\vec{r}_j - \vec{r}_i|| \leqslant r_{\text{c}} \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\quad : \text{otherwise} \end{cases} \tag{16}$$

$$g^k_{\text{local}}(\vec{r}_i, \vec{r}_j) = \begin{cases} g^k_{\text{smooth}}(||\vec{r}_j - \vec{r}_i||) - g_{s_k}(||\vec{r}_j - \vec{r}_i||) : ||\vec{r}_j - \vec{r}_i|| \leqslant s_k \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\quad : \text{otherwise} \end{cases} . \tag{17}$$

Note that $g^k_{\text{local}}(\vec{r}_i, \vec{r}_j)$, $k > 1$, can be pre-computed and represented by a single table with the corresponding pre-factor $2^{-(k-1)}$ imposing a constant coarsening ratio $1 : 2$.

2.4 Linear algebra view

It is informative to cast Eq. (1) as the vector-matrix-vector product

$$\frac{1}{8\pi\varepsilon_0} q^{\text{T}} G q, \tag{18}$$

where q is a vector of particle partial charges, and G is the electrostatic potential *kernel* defined by $G_{ij} = ||\vec{r}_j - \vec{r}_i||^{-1}$, for included i, j interactions. $G(r)$ is not bounded for small r. The multigrid formulation consists in the approximation of the matrix G as a sum of sparse matrices.

MG approximates G as $(G - G^0_{\text{smooth}}) + G^0_{\text{smooth}}$, where $G - G^0_{\text{smooth}}$ is a sparse matrix with a number of nonzeros proportional to $r_c^3 N$. The matrix G^0_{smooth} has slowly varying elements, $G^0_{\text{smooth},ij} = g^0_{\text{smooth}}(||\vec{r}_j - \vec{r}_i||)$, and thus one can further approximate it as a sparse matrix G^1_{smooth} in the level 1 grid (finest grid). The particles are considered as the level 0 "grid" in our notation. MG uses a sparse interpolation matrix I^{k+1}_k, such that $I^k_{k+1} V^{k+1} : V^{k+1} \to V^k$, and a sparse adjoint interpolation or anterpolation matrix A^{k+1}_k, such that $Q^k A^{k+1}_k : Q^k \to Q^{k+1}$. Using these operators,

$$G \approx (G - G^0_{\text{smooth}}) + I^0_1 G^1_{\text{smooth}} A^1_0. \tag{19}$$

Note that we choose $A^1_0 = (I^0_1)^{\mathrm{T}}$, and $G^k_{\text{smooth},rm} = g^k_{\text{smooth}}(||\vec{r}_{h,m} - \vec{r}_{h,k}||)$.

Let Q^k be the charges at grid level k and Q^0 the particle charges. Similarly, $V^k = G^k Q^k$ represents the electrostatic energy values at grid level k, and V^0 the values at the particles. Finally, R^k are the lattice or particle positions ($R^0 = \{\vec{r}_i\}$). The recursive MG scheme with l levels is given by

$$Q^{k+1} = A^{k+1}_k(Q^k), \ k = 1, 2, \ldots, l-1, \tag{20}$$

$$V^l = G^k_{\text{smooth}}(R^k, R^k) \cdot Q(R^k), \tag{21}$$

$$V^k = G^k_{\text{local}}(R^k, R^k) \cdot Q(R^k) + I^k_{k+1}(V^{k+1}), \ k = l-1, l-2, \ldots, 1, \tag{22}$$

$$U = \frac{1}{2} V^1 \cdot Q^1, \tag{23}$$

$$\vec{F}_i = \sum_{R^1_j \in \mathcal{X}(\vec{r}_i)} q_i \nabla_j V^1(R^1_j). \tag{24}$$

$\mathcal{X}(\vec{r}_i)$ denotes the lattice points where $\vec{r}_i$ has local support from I^1.

By reformulating (20-24), the MG scheme can be described as in Algorithm 1, defining a *V-cycle*. The *V-cycle* reflects the order in which the grids are used; the algorithm telescopes down to the coarsest grid, and then works its way back to the finest grid, describing a V. The pseudo-code for **main** handles the aggregation of charges from particles to the coarsest grid, and the interpolation of the kernel values from the coarsest grid to the particles. Lastly, it computes the force contributions and the total energy. **multiscale** recursively performs the aggregation of charges, the interpolation of potential values and the local correction as depicted in Figure 2. Due to the uniformity of the grids, the interpolation coefficients can be pre-calculated and represented by one 1-dimensional set of coefficients. The same holds for the local correction, since the softening distance is proportional to its corresponding mesh size.

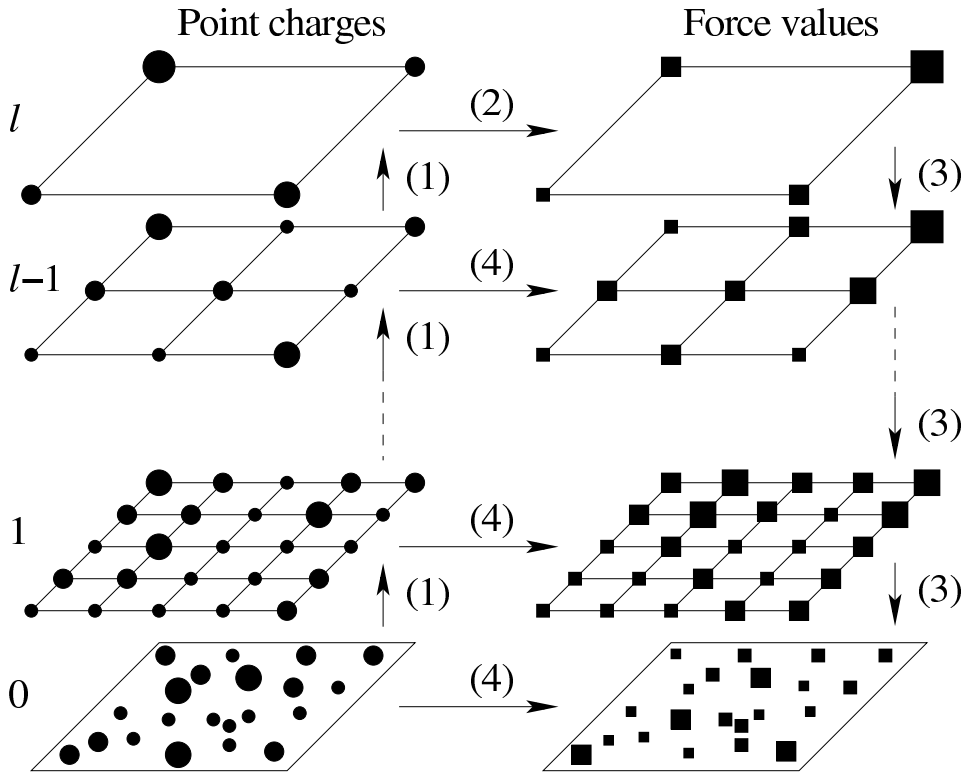


Fig. 2. The multilevel scheme of the multi-grid algorithm. (1) Aggregate to coarser grids; (2) Compute potential induced by the coarsest grid; (3) Interpolate potential values from coarser grids; (4) Local corrections.

2.5 Multigrid Summation for Periodic Boundary Conditions

We present our extension of MG to periodic boundary conditions. Eq. (2) can be cast as

$$\frac{1}{8\pi\varepsilon_0}\sum_{\vec{m}} q^{\mathrm{T}} G_{\vec{m}} q, \tag{25}$$

where $G_{\vec{m},ij} = ||\vec{r}_j - \vec{r}_i + \vec{m}L||^{-1}$, for included i, j interactions and lattice cells $\vec{m}$. This summation is approximated with a finite number of terms corresponding to $\vec{m}$ in the neighborhood of the periodic cell. In this case, the algorithm handles PBC by doing the following:

- Coordinates are wrapped around in each dimension that is periodically extended. Thus, the interpolation and adjoint interpolation always have enough grid points (or support). An exception is G^l_{local} for the coarsest grid, which does not wrap around.
- Multiple copies of the periodic cell are included in the summations. At the very least, in 3-d, the support includes the closest copy for a boundary grid point. Running over all grid points in 3-d, boundary grid points from 26 neighbor copies are considered.
- As a consequence of the above, for PBC the grids cover the same area at

each level, but for vacuum the area covered increases for increasing level, i.e., the coarsest grid covers all other grids: Points of a finer grid will have points outside its area of support when doing adjoint interpolation to the coarser grid.

In our implementation of MG, the order of the interpolation schemes can be chosen from among several generic interpolation routines, which operate from particles to grid, grid to particles, and from grid to grid. Internally, the algorithm keeps a multi-grid structure that contains a hierarchy of grids. The interpolation from particles to grid and the grid data structures are reused by PME. MG is very competitive and has already enabled material science simulations with millions of atoms that would be otherwise intractable, cf. [21, pp. 49 ff.], [12].

2.6 Time Complexity and Error Estimation

Assuming uniform grids, a constant coarsening ratio of $1:2$, a p-order interpolation, h the grid point separation for the finest grid and an average particle density $\rho^3 = \frac{N}{V}$ bounded by some constant, the total work of MG is given by

$$\mathcal{O}\left(N\left(\left(\frac{s}{h}\right)^3 + p^3\right)\right) \ . \tag{26}$$

Accuracy is governed by the size of the interpolation error of the smooth part. A detailed error estimation of MG is given in [9]. Assuming a C^p-continuous kernel G and at least p-order accurate interpolation, the relative force error of the smooth part is $\mathcal{O}\left(\frac{h^p}{s^p}\right)$ and the total relative force error is

$$\mathcal{O}\left(\frac{h^p}{\rho^2 s^{p+2}}\right) \ . \tag{27}$$

In [20, pp. 6-31], MG's performance for vacuum systems is compared with the direct method for a 2-dimensional system of charges and dipoles. In [9], MG is compared against the fast multi-pole method that is implemented in the parallel program DPMTA [22] for water systems. As expected, experiments from [9] show that for $s \to \infty$, MG converges to the direct method and the error drops to zero monotonously. The work increases monotonously with s, until s is large enough to encompass all pairs and remains constant. Furthermore, it was indicated in [9] that MG produces stable simulations when used for molecular dynamics for much lower accuracy than multi-pole methods.

3 Parallel multigrid summation

The most scalable parallelization of the multigrid summation would use domain decomposition and spatial data distribution: each node works on its local domain and propagates its results in a tree fashion way. Each local domain consists of the assigned domain and some overlap or ghost points to ensure correct interpolation. The overlap depends on the interpolation order and also on the softening distance when computing the local corrections. In order to achieve high scalability one needs to minimize the overlap and assign other work to idle nodes, for example, computing the direct part.

We implemented instead both atom and force decompositions versions of the algorithm (cf. [23]). For each step of the MG algorithm, in each grid, the work is distributed among all nodes. The local contributions are propagated to the next level, and then the MG algorithm proceeds. This involves synchronization among nodes when going from one grid to the next.

3.1 Direct sum

The direct sum consists of all pair wise interactions within the softening distance and the intra-molecular correction, which adds the effects of covalent bond interactions. The latter one can easily be distributed among the nodes since the pairs are statically given by the topology of the whole system. The pair-wise interactions can not be determined statically since the number and location of interactions for a given softening distance vary as a function of time. Nevertheless, using a cell-algorithm, also called *geometric hashing,* one can split the sum of interactions into small parts and solve the problem in $\Theta(N)$ time. Each part consists of all interactions of one cell and the neighboring cells within the softening distance.

For the atom decomposition, a simple distribution based on a modulo function gives good load balancing. For the force decomposition, we need another cell list, the transpose of the original cell list, to give us a force matrix view of the system.

3.2 Interpolation

The parallelization of interpolation and adjoint interpolation is based on an atom decomposition. Adjoint interpolation and interpolation require local support from the previous grid or particle level. For adjoint anterpolation of the charges to the coarsest grid, it is more efficient to perform a global sum over

the grid values to proceed to the next step in the MG algorithm than to recompute values. For interpolation of the force contributions there is no global summation, since the force and energy contributions are updated in a lazy manner at the end of the time step, when they are needed.

3.3 Smooth part of the sum

The computation of the smooth part of MG consists of interpolation steps between grids (Steps 1 and 3, Figure 2), local corrections between the potential and charge grids (Step 4, Figure 2), and a direct part on the the coarsest grid (Step 2, Figure 2).

For the atom decomposition, given the dimensions of the grids and the softening distance we can exactly predict the work for each single MG step. This enables us to distribute the work evenly among all nodes without any communication or scheduler. Our approach requires a global reduction after each MG step, but we avoid idle nodes through the even work distribution. Also, the total work over all nodes is the same as in a sequential run, which is not the case for a distributed implementation requiring ghost grid points. In the latter, for coarse grids, the amount of work increases significantly (since the grid size is comparable to the number of ghost points).

For the force decomposition, the communication is $O(N/\sqrt{P})$, where P is the number of processors at the end of the MG computation. This produces the greatest savings in communication and makes this approach more scalable than atom decomposition, and for the systems we tested, nearly as scalable as the dynamically load balanced hybrid force-domain decomposition of NAMD 2.5.

4 Results

We implemented MG, plain Ewald summation, and PME in ProtoMol version 2.0.2. All the methods have been parallelized using force decomposition (FD) or atom decomposition (AD). Their sequential versions are reasonably optimized, and commonalities among these algorithms are exploited. For example, the interpolation routines are common to both MG and PME. The algorithms for computing the direct sums using cell-lists are also common to all the fast electrostatics methods. ProtoMol 2.0.2 has been ported to AIX, IRIX, HP-UX, Solaris, Linux and Windows using vendor specific compilers if possible or GNU's g++.

We compare the running times of MG against a highly optimized parallel MD program, NAMD 2.5 [11], which uses PME for fast electrostatics and a hybrid spatial-force decomposition with dynamic load balancing. The results are presented for two different computer systems:

(1) IBMp690 Regatta Turbo (`http://tre.ii.uib.no`) with 3 nodes of 32 Power4 1.3 GHz processors per node. There is a total of 320 Gigabytes of memory. It runs AIX 5.1. ProtoMol 2.0.2 was compiled by configuring using the option ``` ``configure --with-aix-xlc-mpi'' ```.
(2) Atipa Linux cluster (`http://iss.cse.nd.edu`) with 44 dual Xeon 2.4 GHz processors with Myrinet interconnect. There is a total of 90 Gigabytes of memory. This cluster runs Linux Red Hat Linux 8.0 3.2-7, kernel 2.4.18-18.8.0smp. ProtoMol 2.0.2 was configured using ``` ``configure --with-gcc-mpich'' ``` and uses Myrinet MPI libraries. NAMD 2.5 was configured using ``` ``config tcl fftw plugins Linux-i686-MPI'' ```. It uses the parallel runtime system `Charm++`. This was built using ``` ``build charm++ mpi-linux gm2 -O -DCMK_OPTIMIZE=1'' ``` to include Myrinet MPI libraries and to optimize the library.

Our test cases consist of three types of systems. First, Coulomb crystals in vacuum, only consisting of ions and an outer field holding them together. Second, periodically-replicated flexible TIP3P water boxes of different sizes. Finally, a periodically-replicated protein solvated in flexible TIP3P water.

Both time and accuracy are measured (e.g., relative average force error). The Ewald method is assumed to be the standard for comparison when the experiments are done using PBC while the direct method is used for comparison when the experiments are done in vacuum. We give details of all the experimental conditions in Section 4.5.

4.1 Coulomb Crystals

Coulomb crystal systems [12, 24] are defined by a computationally dominant electrostatic part and an electric field with linear work complexity. We run tests from 1,000 to 1,000,000 atoms using MG in vacuum. The simulations were performed on the IBM p690 Regatta Turbo described above. Because this is a shared memory machine, there is not much difference between the atom decomposition and force decomposition versions of MG.

Figure 3 shows the parallel speedup of an atom-decomposition version of MG in vacuum implemented in ProtoMol 2.0.2. MG exhibits excellent scaling, even when the integration step (propagation of positions and velocities) is redundantly performed on all nodes. The scaling is close to linear, with a slight slow down due to cache and memory effects and the fact that accuracy

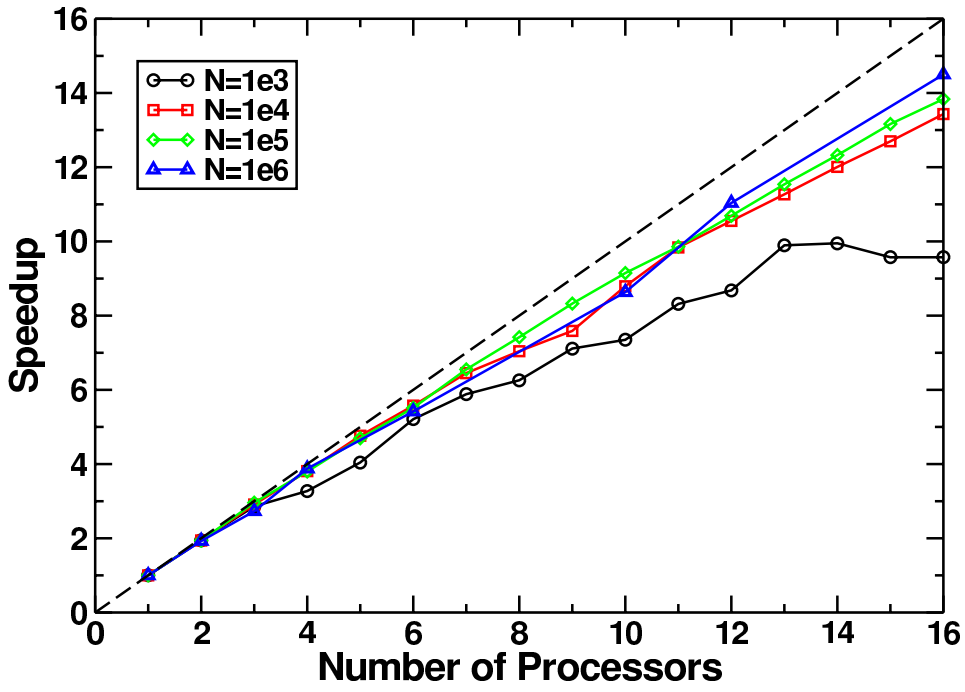


Fig. 3. Parallel speedup of MG electrostatic solver applied on Coulomb Crystal systems with relative error of order 10^{-5} or less; performed on an IBM p690 Regatta Turbo. N is the number of atoms in the system.

increases slightly with the system size when distance between grid points is kept the same for all systems. N is the number of atoms in the system. Note that the sequential speedup for $N = 10^6$ is of order 10^2 or more compared to the direct method, and for lower accuracy a speedup of order 10^3 was observed.

4.2 Different Size Water Boxes in Periodic Boundary Conditions

We run tests of different size water boxes. The experiments are based on a flexible variant of the TIP3P water model (cf. [25]). The system sizes are varied from 3,240 to 400,002 atoms. These systems use periodic boundary conditions.

Figure 4 compares the sequential run-time of MG and PME in ProtoMol 2.0.2 and PME implemented in NAMD 2.5. MG and PME are tested in periodic boundary conditions. The relative error in the forces is $\epsilon = 10^{-3}$. The same C^2-continuous switching function is used for all tests. This illustrates the linear scaling of MG. Note that MG is faster than PME in ProtoMol 2.0.2, but PME is faster in NAMD 2.5. The latter has many levels of optimization that are not present in ProtoMol 2.0.2.

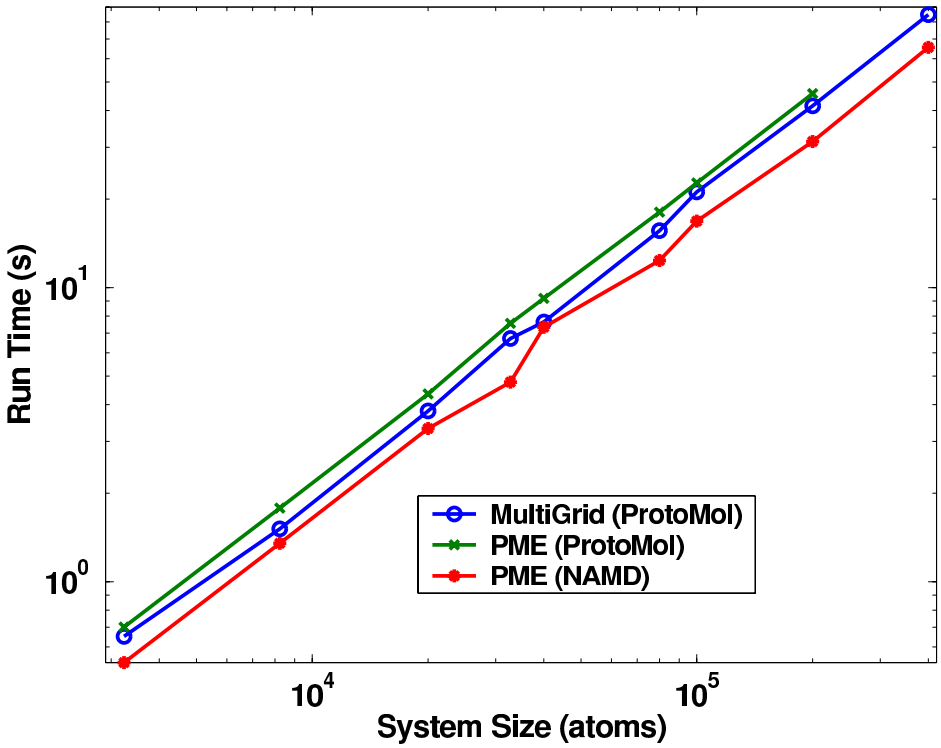


Fig. 4. Single processor run time for 8 MD steps for N-body solvers implemented in ProtoMol 2.0.2 and NAMD 2.5 with relative force error of order 10^{-3}. Runs performed on Linux cluster with 44 dual-processor Xeon 2.4GHz with Myrinet interconnect.

4.3 Solvated Protein in Periodic Boundary Conditions

Apolipoprotein A-I (apoA-I) is the primary protein constituent of high density lipoprotein (HDL), which circulates in the bloodstream, extracts cholesterol from body tissues and transports it to the liver for excretion or recycling [26, 27]. NAMD 2.5 uses a solvated structure of apoA-I with 92,224 atoms as a performance benchmark. We use the same benchmark to measure the efficiency of MG.

Figures 5 and 6 show the parallel speedup and run time of the benchmark MD simulations in ProtoMol 2.0.2, using atom (AD) and force (FD) decomposition versions of MG and PME. Also shown is the run time of NAMD 2.5 using spatial decomposition (SD) PME. Method parameters are chosen to give roughly the same accuracy and run time in 1 processor.

Both AD and FD versions of MG scale up to 64 processors. As expected, the FD version scales better, and indeed for 66 processors is more than 2 times faster than the best time for the AD version of MG, with 64 processors. The best run time of FD MG, 2.53 ± 0.01 seconds with 66 processors, is 5.6 times faster than the best run time of FD PME, which is 14.12 ± 0.02 seconds with

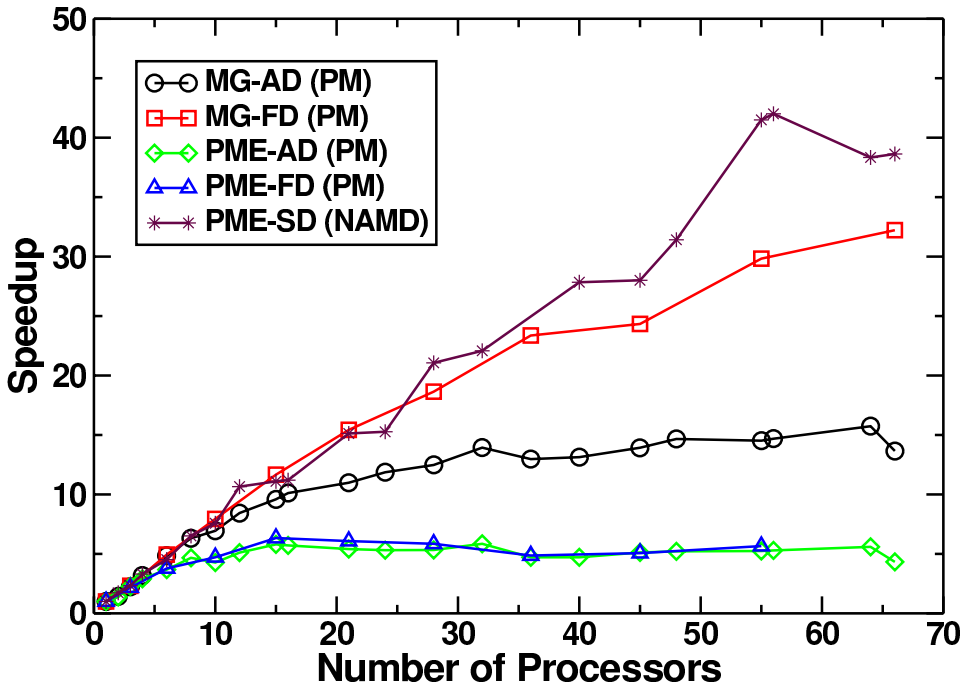


Fig. 5. Speedup comparison of MG using atom (AD) or force decomposition (FD) vs. PME using either decomposition in ProtoMol 2.0.2 and spatial decomposition (SD) in NAMD 2.5. Data is for apoA-I, 92,224 atoms. PME uses the parallel FFT library `FFTW` 2.1.5; performed on a Linux cluster of Xeon 2.4 GHz with Myrinet interconnection network.

15 processors. The best run time of FD MG is slightly faster than the best run time of PME in NAMD 2.5. More interestingly, the FD MG continues to scale for more than 64 processors, whereas NAMD 2.5 starts to slow down at that point.

NAMD 2.5 has a much more sophisticated parallel implementation of PME for molecular dynamics than ProtoMol 2.0.2. It performs a hybrid spatial data decomposition and force decomposition of the computation. It also performs dynamic load balancing. To improve scaling when using PME, they restrict the number of processors that run FFT. Thus, it shows better scaling than either version of MG in ProtoMol 2.0.2. For simulations on larger number of processors, the FFT becomes the bottleneck to scalability in NAMD 2.5, as starts to be seen in our tests for 64 or more processors. MG would scale much better than PME if implemented within NAMD 2.5 or implemented in ProtoMol 2.0.2 using spatial distribution and dynamic load balancing.

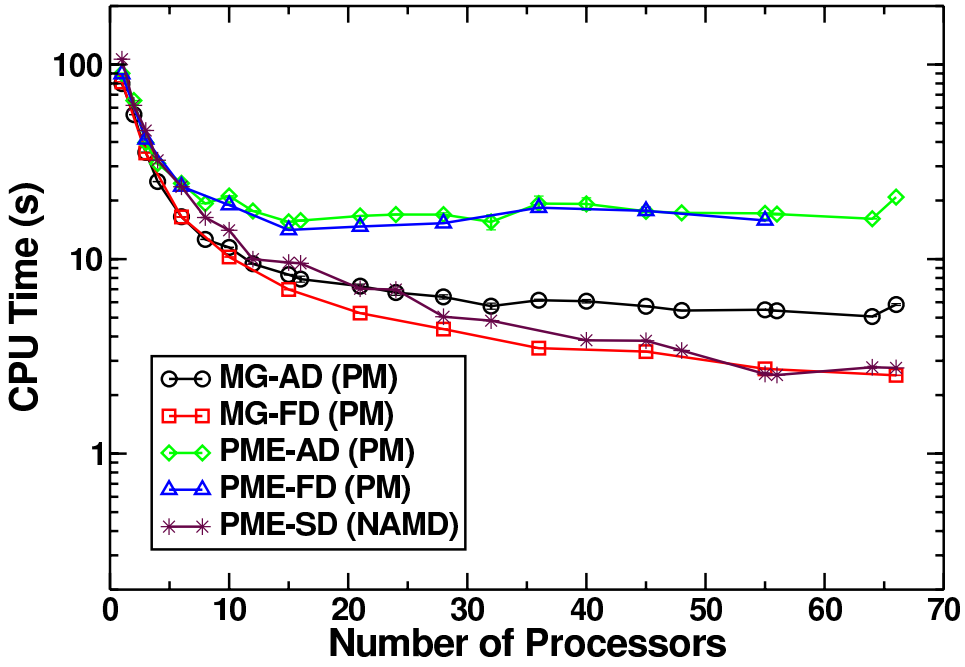
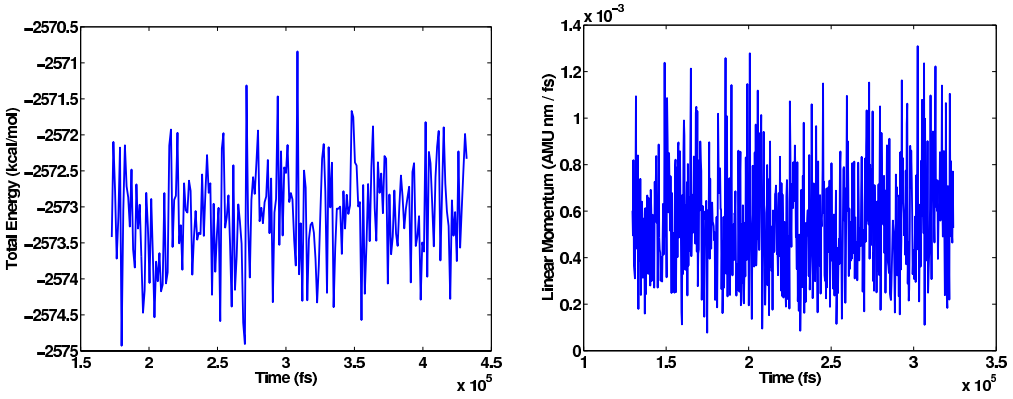


Fig. 6. CPU time MG using atom (AD) or force decomposition (FD) vs. PME using either decomposition in ProtoMol 2.0.2 and spatial decomposition (SD) in NAMD 2.5.. Data is for apoA-I, 92,224 atoms. PME uses the FFT library `FFTW`; performed on a Linux cluster of Xeon 2.4 GHz with Myrinet interconnection network.

4.4 Conservation Properties

Neither linear momentum nor angular momentum are theoretically preserved by MG. The introduction of a gridding of space perturbs the value of the potential energy function so that it is not invariant to rigid body rotation or translation. The momenta fluctuate around a constant value and do not suffer from overheating or freezing. Note that for MD simulations using PBC, angular momentum is not conserved anyway.

To validate that the momentum oscillates around a constant value with MG, we ran a simulation using crambin, a small protein with PDB id 1EJG. We solvated it in flexible TIP3P water for a total of 2,277 atoms. We minimized and equilibrated it using NAMD 2.5 using an identical procedure to the apoA-I. Then, we performed a simulation for 500 ps using a triple time stepping r-RESPA, with innermost time step of 0.75 fs for bonds, angles, Lennard–Jones and direct part of MG; 1.5 fs time step for impropers, dihedrals, and the correction for MG; and 3 fs time step for the smooth part of MG. We remove every few steps the linear and angular momentum of the center of mass of the system. This is necessary in vacuum simulations in general to prevent movement of the molecule. We confirmed this is necessary by running simulations using plain cutoffs.



(a) (b)

Fig. 7. (a). Total energy for solvated crambin in vacuum with spherical boundary conditions using MultiGrid summation. (b). Linear momentum for solvated crambin in vacuum with spherical boundary conditions using MultiGrid summation.

Figures 7(a) and 7(b) show the total energy and linear momentum of the system. We discard the first 150 ps of simulation and plot the next 350 ps. It can be seen that despite the long time steps, the energy and momentum are conserved in practice.

4.5 Methods and Simulations

The Coulomb crystals were generated using the Ion Crystal generator tool of ProtoMol 1.8.3. For example, for 1000 atoms the command used was `ioncrystalGenerator -3d -nvt -temp 1e-3 -d 1e5 -w 2.5e-9 1000 CA 40.08 1.0 1000 A 80.16 2.0`. The simulations use a single time stepping Nosé-Hoover NVT integrator. The forces used include a multigrid Coulomb force with quintic interpolation, 2 levels, a C^3 kernel, and a Paul Trap force with $\omega_r = \omega_z = 2.5e-9$. Simulation data and configuration files are available upon request.

The water boxes were generated using VMD [28]. For each water box, 1000 minimization steps were run with NAMD 2.5. Then, starting from a temperature of 0 K, 1000 steps were run between increments of 25 K until a temperature of 300 K was reached. For equilibration, additional 8000 steps were run at 300 K. All these equilibration runs used a Verlet/leapfrog integrator with time step of 1 fs, and cutoff for Lennard–Jones and Coulomb of 12 Å. The switching distance for a C^2 switching function for Lennard–Jones was 8 Å. To run the experiments comparing MG and PME in ProtoMol 2.0.2 and PME in NAMD 2.5, a short cutoff of Lennard–Jones of 6 Å and a switching distance of 3 Åwas used for all methods. The reason is that NAMD 2.5 only

reports run time for all the MD integration and force calculation. Thus, we tried to minimize the time spent in non-Coulomb calculations. For PME, the separation between grid points was set to 1 Å, a standard practice. MG was run with 2 grid levels, a C^2 kernel, cubic interpolation, a smoothing distance of 12 Å in the finest grid, and between 1000 and 3000 grid points in the finest grid. These tests use a multiple time stepping r-RESPA scheme where every 1 fs all bonded forces and the direct and correction parts of MG or PME are evaluated, and every 4 fs the smooth or reciprocal parts of MG or PME are evaluated. Both ProtoMol 2.0.2 and NAMD 2.5 are compiled without MPI for these uni-processor runs.

For the apoA-I simulations, the same parameters used by the NAMD 2.5 performance benchmark were used: a switching distance of 8 Å, a cutoff of 10.5 Å for the Lennard–Jones and direct part of MG, and r-RESPA with all bonded forces and short range Lennard–Jones and Coulomb evaluated every 1 fs, and the smooth or reciprocal part of MG or PME evaluated additionally every 4 fs. Three runs were performed for each data point in the results. The error bars were smaller than the symbols. Both ProtoMol 2.0.2 and NAMD 2.5 are compiled for using the MPI library with Myrinet support. We noted that the Myrinet MPI version of NAMD 2.5 consumes significantly more memory than the uni-processor version, and also than ProtoMol 2.0.2.

5 Related Work

5.1 Multigrid summation techniques

The earliest reference that uses multilevel matrix multiplication in the context of the fast solution of integral equations is [6]. There, an $\Theta(Np^3)$ method, where p is the order of the interpolation, is applied to the kernel $G(x, y) = \ln|x - y|$. An early vectorized version of MG summation for this kernel is in [29]. The method is extended to oscillatory kernels in [30] at a cost of $\Theta(p^3 N \log N)$.

A low accuracy adaptive multigrid summation is developed in [8]. For low accuracy computations this method competes favorably against the fast multipole implementation of Board and collaborators [31], but not for high accuracy. Speaking about parallelism, the authors of Ref. [8] note that

> The structure of the adaptive multigrid algorithm (...) is highly parallel, i.e., many levels of parallelism can be exploited. These include calculation of the force field at the different points in the scheme in tandem, parallel evaluation of the summation for each point and at each level using parallel reduction,

evaluation of the sum at different levels (atomic level and sequence of grids) in parallel using the additive property of the sum and, finally, instruction-level parallelism. (p. 249)

A more recent implementation of the MG method in the context of MD is that of Skeel and collaborators [9]. Using smooth interpolation they show that MG is better than the fast multipole when used in MD, since it has better smoothness properties that make the numerical integrator of the equations of motion stable. They achieve a considerably faster implementation of MG than [8].

Our version of MG summation extends the work of [8, 9] to also work in periodic boundary conditions. We have also parallelized it using both a simple master/slave decomposition and a force decomposition. MG's advantage over PME is clearly seen in its scalability. For MD codes that do not have the sophisticated dynamic load balancing of NAMD 2, MG will scale better because of its lower communication costs.

The iterative multigrid method of [32] uses the particle-mesh (PM) approach, but then transforms the problem in the mesh to the solution of an elliptic PDE, and uses an $\Theta(N)$ iterative multigrid solver, achieving an $\Theta(N)$ solver. This is a high-accuracy scheme, unlike our algorithm and those of [8, 9]. This method is expected to be more expensive and less scalable in parallel due to the iterative solver, and to the fewer levels of parallelism available in the method. It is slower than PME for a single processor, whereas our method is faster than PME for a single processor.

5.2 Ewald and multipole methods

An important comparison of FMM, particle-particle particle-mesh (P3M), and Ewald is [33]. They empirically test reasonably efficient implementations of these methods. P3M is a method similar in spirit to PME. Their conclusion is that P3M is both faster than the FMM and easier to implement efficiently as it relies on commonly available software (FFT subroutines). Both the Ewald and P3M method are easily implemented on parallel architectures with the P3M method the clear choice for large systems.

The similarities between P3M, PME, and smooth PME, as well as the influence of methods parameters on accuracy are described in [34]. They conclude that P3M is the most flexible approach, capable of achieving the largest accuracy by using a force interpolation variant of the method.

Similar conclusions are reached by [35]. They point out that in cases in which it is less expensive to use a higher-order interpolation than to perform two extra

FFT's, the smooth PME is preferable. In particular, this holds for parallel implementations, for which FFT is not scalable. They also discuss another variant on PM methods, the fast Fourier Poisson method of York and Yang [36]. This avoids errors on the interpolation by sampling Gaussian sources directly at the grid point, using clever mathematical identities. It has much higher accuracy than PME or P3M, but it is also more costly.

The advantages of multipole methods, particularly in the context of parallel processing, are reviewed in [37]. Other reviews of fast summation techniques are in [38–40]. Cell methods have continued to improve: multipole has been extended to PBC [41]; a hierarchical $\mathcal{O}(N)$ cell method has been developed that conserves momentum through the use of symmetric Cartesian Taylor approximations [42]; the Tree Particle-Mesh algorithm has been experimentally shown to be faster than P3M [43]; and an elegant adaptive treecode has been implemented in vacuum and PBC, and shown to be easier to implement than multipole [44, 45].

5.3 Parallel N-body solvers

One of the most influential parallel N-body solvers is the parallel cell code of [46, 47]. The work of Board and collaborators has produced robust parallel software using the multipole method [22, 48, 49]. The book [50] has early references of work in this area. Recent work by [51] exploits the hierarchical decomposition in space of cell methods to provide high compression ratios (about 6:1 larger than gzipped raw data), which enables faster storage and retrieval of simulation data.

Force-decomposition, which is the basis of this implementation of parallel MG, is described in [23]. It has also been adopted by the IBM Bluegene project for their petaflop computer. Its advantage is that it can be simply load-balanced and performs well for irregular geometries. It improves the scaling due to replicated data techniques, and is simpler than full spatial-decomposition.

Comparisons of static and dynamic decomposition techniques appear in [52, 53]. For more adaptive applications than molecular dynamics, such as astrophysics simulations, combinations of static and dynamic load balance have proven useful: For example, a combination of static decomposition for the grid and dynamic balance for particles in a P3M parallel code scales to thousands of processors [54]. Similar ideas appear in [55].

Parallel iterative multigrid PDE solvers are presented in [56–58]. These solvers are different than the MG summation presented here, but share the multiple grid structure and similar parallelization issues. The latter two papers deal with clever partitionings for adaptive grids. A comparison of techniques that

try to compensate deficiency of parallelism on coarser grids in multigrid solvers is in [59]. They evaluate a multiple coarse grid technique advocated by [60], and additive MG that allows computation on all grids simultaneously [61]. Their analysis suggests that standard algorithms are substantially more efficient than either method, except for highly impractical number of processors.

6 Discussion

We have shown that the MG method in periodic boundary conditions is competitive with PME, being slightly faster in most of our tests. More importantly, it is more scalable than the FFT that is part of PME, and thus can scale to larger number of processors. We have shown this through a force decomposition implementation of MG that is faster than a spatial decomposition of PME in NAMD 2.5. A spatial decomposition of MG would scale even better. For scaling to more nodes, one might benefit from dynamic balancing for the direct summation on particle space, such as NAMD 2.5 does.

One of the major advantages of MG is that it produces stable dynamics with much lower accuracy than other methods [9] and converges to the correct solution if the softening distance goes to infinity. Furthermore, MG is an excellent candidate for multiple time stepping, since each grid level can be associated with its own time step.

A limitation of MG is that there are more parameters to determine than in simpler methods such as Ewald. We have developed a tool called MDSimAid that assists potential users in the determination of optimal parameters for PME and MG for a given system size and accuracy [13].

Besides doing a spatial decomposition for MG, other areas for improvement to the method include the following: Development of a higher accuracy version: this requires a more careful analysis of the interpolation errors and use of better interpolation functions, cf. [62]; optimization in the context of multiple time stepping algorithms such as r-RESPA [63], cf. work on optimization of PME coupled to r-RESPA in [17]; formulas for computing the virial using MG; and most importantly, a rigorous justification of MG summation in PBC.

Acknowledgments

Special thanks to David Hardy and Robert D. Skeel for sharing details of their vacuum multigrid summation implementation, which allowed cross-verification of our implementation. Chengbang Huang generated many of the plots in this

paper. This research was partially funded by National Science Foundation grants ACI-0135195 and IBN-0083653; an Equipment Renovation Grant to JI by the University of Notre Dame, Indiana; the Research Council of Norway and the Norwegian High Performance Computing Consortium (NOTUR). SH was partially supported by an Arthur J. Schmitt fellowship.

References

[1] T. E. Cheatham, J. L. Miller, T. I. Spector, P. Cieplak, P. A. Kollman, Molecular dynamics simulations on nucleic acid systems using the Cornell et al force field and particle mesh Ewald electrostatics, in: ACS Symposium Series: Molecular Modeling and Structure Determination of Nucleic Acids, Am. Chem. Soc., 1997.

[2] J. Barnes, P. Hut, A hierarchical $O(N \log N)$ force-calculation algorithm, Nature 324 (1986) 446–449.

[3] L. Greengard, V. Rokhlin, A fast algorithm for particle simulation, J. Comput. Phys. 73 (1987) 325–348.

[4] T. Darden, D. York, L. Pedersen, Particle mesh Ewald: An N log(N) method for Ewald sums in large systems, J. Chem. Phys. 98 (12) (1993) 10089–10092.

[5] U. Essmann, L. Perera, M. L. Berkowitz, A smooth particle mesh Ewald method, J. Chem. Phys. 103 (19) (1995) 8577–8593.

[6] A. Brandt, A. A. Lubrecht, Multilevel matrix multiplication and fast solution of integral equations, J. Comput. Phys. 90 (1990) 348–370.

[7] B. Sandak, Multiscale fast summation of long-range charge and dipolar interactions, J. Comp. Chem. 22 (7) (2001) 717–731.

[8] L. Y. Zaslavsky, T. Schlick, An adaptive multigrid technique for evaluating long-range forces in biomolecular simulations, Appl. Math. Comput. 97 (1998) 237–250.

[9] R. D. Skeel, I. Tezcan, D. J. Hardy, Multiple grid methods for classical molecular dynamics, J. Comp. Chem. 23 (6) (2002) 673–684.

[10] T. Matthey, T. Cickovski, S. S. Hampton, A. Ko, Q. Ma, M. Nyerges, T. Raeder, T. Slabach, J. A. Izaguirre, ProtoMol: An object-oriented framework for prototyping novel algorithms for molecular dynamics, ACM Trans. Math. Softw. 30 (3) (2004) 237–265.

[11] L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, K. Schulten, NAMD2: Greater scalability for parallel molecular dynamics, J. Comput. Phys. 151 (1999) 283–312.

[12] T. Matthey, J. P. Hansen, M. Drewsen, Coulomb bi-crystals of species with identical charge-to-mass ratios, Phys. Rev. Lett. 91 (16) (2003) 165001.

[13] M. S. Crocker, S. S. Hampton, T. Matthey, J. A. Izaguirre, MDSimAid: Automatic parameter optimization in fast electrostatic algorithms, J. Comp. Chem.In press. Manuscript available at `http://www.nd.edu/~izaguirr/papers/CHMI05.pdf`.

[14] S. W. de Leeuw, J. W. Perram, E. R. Smith, Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constants, Proc. R. Soc. Lond. A 373 (1980) 27–56.

[15] P. Ewald, Die Berechnung optischer und elektrostatischer Gitterpotentiale, Ann. Phys. 64 (1921) 253–287.

[16] D. Fincham, Optimisation of the Ewald sum for large systems, Mol. Sim. 13 (1994) 1–9.

[17] P. F. Batcho, D. A. Case, T. Schlick, Optimized particle-mesh Ewald/multiple-time step integration for molecular dynamics simulations, J. Chem. Phys. 115 (9) (2001) 4003–4018.

[18] R. W. Hockney, J. W. Eastwood, Computer Simulation Using Particles, McGraw-Hill, New York, 1981.

[19] T. Bishop, R. D. Skeel, K. Schulten, Difficulties with multiple timestepping and the fast multipole algorithm in molecular dynamics, J. Comp. Chem. 18 (14) (1997) 1785–1791.

[20] A. Brandt, J. Bernholc, K. Binder (Eds.), Multiscale Computational Methods in Chemistry and Physics, Vol. 177 of NATO Science Series: Series III Computer and Systems Sciences, IOS Press, Amsterdam, Netherlands, 2001.

[21] T. Matthey, Framework design, parallelization and force computation in molecular dynamics, Ph.D. thesis, University of Bergen, Bergen, Norway (2002).

[22] W. Rankin, J. Board, A portable distributed implementation of the parallel multipole tree algorithm, IEEE Symposium on High Performance Distributed Computing[Duke University Technical Report 95-002].

[23] S. Plimpton, B. Hendrickson, A new parallel method for molecular dynamics simulation of macromolecular systems, J. Comp. Chem. 17 (3) (1996) 326.

[24] R. H. Hasse, V. V. Avilov, Structure and Mandelung energy of spherical Coulomb crystals, Phys. Rev. A 44 (7) (1991) 4506–4515.

[25] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, M. L. Klein, Comparison of simple potential functions for simulating liquid water, J. Chem. Phys. 79 (1983) 926–935.

[26] J. H. Wald, E. S. Krul, A. Jonas, Structure of apolipoprotein A-I in three homogeneous reconstituted high density lipoprotein particles, J. Biol. Chem. 265 (1990) 20037–20043.

[27] A. Jonas, K. E. Kezdy, J. H. Wald, Defined apolipoprotein A-I conformations in reconstituted high density lipoprotein discs, J. Biol. Chem. 264 (1989) 4818–4824.

[28] W. F. Humphrey, A. Dalke, K. Schulten, VMD – Visual Molecular Dynamics, J. Mol. Graphics 14 (1996) 33–38.

[29] D. S. Balsara, A. Brandt, Multilevel methods for fast solution of N-body and hybrid systems, in: International Series of Numerical Mathematics, Vol. 98, Birkhäuser Verlag, Basel, 1991, pp. 131–142.

[30] A. Brandt, Multilevel computations of integral transforms and particle interactions with oscillatory kernels, Comput. Phys. Commun. 65 (1-3) (1991) 24–38.

[31] J. A. Board, Jr., J. W. Causey, J. F. Leathrum, Jr., A. Windemuth, K. Schulten, Accelerated molecular dynamics simulation with the parallel fast multipole algorithm, Chem. Phys. Lett. 198 (1992) 89–94.

[32] C. Sagui, T. Darden, Multigrid methods for classical molecular dynamics simulations of biomolecules, J. Chem. Phys. 114 (15) (2001) 6578–6591.

[33] E. L. Pollock, J. Glosli, Comments on PPPM, FMM, and the Ewald method for large periodic Coulombic systems, Comput. Phys. Commun. 95 (1996) 93–110.

[34] M. Deserno, C. Holm, How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines, J. Chem. Phys. 109 (18) (1998) 7678–7693.

[35] C. Sagui, T. A. Darden, Molecular dynamics simulations of biomolecules: Long-range electrostatic effects, Ann. Rev. Biophys. Biomol. Struct. 28 (1999) 155–179.

[36] D. York, W. T. Yang, The fast Fourier–Poisson method for calculating Ewald sums, J. Chem. Phys. 101 (1994) 3298–3300.

[37] A. Y. Toukmaji, J. A. Board, Ewald summation techniques in perspective: A survey, Comput. Phys. Commun. 95 (1996) 73–92.

[38] L. Greengard, Science 265 (1994) 903–914.

[39] C. L. Berman, Grid-multipole calculations, SIAM J. Sci. Comput. 16 (1995) 1082–1091.

[40] T. Darden, A. Toukmaji, L. Pedersen, J. Chem. Phys. 94 (1997) 1346.

[41] C. G. Lambert, T. A. Darden, J. A. Board, A multipole-based algorithm for efficient calculation of forces and potentials in macroscopic periodic assemblies of particles, J. Comput. Phys. 126 (2) (1996) 274–287.

[42] W. Dehnen, A hierarchical O(N) force calculation algorithm, J. Chem. Phys. 179 (1) (2002) 27–42.

[43] P. Bode, J. P. Ostriker, Tree-Particle Mesh: An adaptive, efficient, and parallel code for collisionless cosmological simulation, Astrophysical J. Supplement Series 145 (1) (2003) 1–13.

[44] Z. H. Duan, R. Krasny, An adaptive treecode for computing nonbonded potential energy in classical molecular systems, J. Comp. Chem. 22 (2) (2001) 184–195.

[45] Z. H. Duan, R. Krasny, An Ewald summation based multipole method, J. Chem. Phys. 113 (9) (2000) 3492–3495.

[46] M. S. Warren, J. K. Salmon, Astrophysical N-body simulations using hierarchical tree data structures, in: Supercomputing '92, IEEE Computer Society Press, 1992, pp. 570–576.

[47] M. S. Warren, J. K. Salmon, A portable parallel particle program, Comput. Phys. Commun. 87 (1-2) (1995) 266–290.

[48] A. Toukmaji, C. Sagui, J. Board, T. Darden, Efficient particle-mesh Ewald based approach to fixed and induced dipolar interactions, J. Chem. Phys. 113 (24) (2000) 10913–10927.

[49] J. Board, K. Schulten, The fast multipole algorithm, IEEE Comp. Sci. & Eng. 2 (2000) 56–59.

[50] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, D. W. Walker, Solving Problems on Concurrent Processors, Vol. 1, Prentice Hall, Englewood Cliffs, NJ, 1988.

[51] D. Y. Yang, A. Grama, V. Sarin, K. Ramakrishnan, Compression of particle data from hierarchical approximate methods, ACM Trans. Math. Softw. 27 (3) (2001) 317–339.

[52] J. P. Singh, C. Holta, T. Totsuka, A. Gupta, J. Hennessy, Load balancing and data locality in adaptive hierarchical N-body methods - Barnes-Hut, Fast Multipole and Radiosity, J. Paral. Distrib. Comp. 27 (2) (1995) 118–141.

[53] A. Grama, V. Kumar, A. Sameh, Scalable parallel formulations of the Barnes-Hut method for N-body simulations, J. of Parallel Computation 24 (5-6) (1998) 797–822.

[54] S. J. Plimpton, D. B. Seidel, M. F. Pasik, R. S. Coats, G. R. Montry, A load-balancing algorithm for a parallel electromagnetic particle-in-cell code, Comput. Phys. Commun. 152 (2) (2003) 227–241.

[55] S. Plimpton, S. Attaway, B. Hendrickson, J. Swegle, C. Vaughan, D. Gardner, Parallel transient dynamics simulations: Algorithms for contact detection and smoothed particle hydrodynamics, J. Paral. Distrib. Comp. 50 (1-2) (1998) 104–122.

[56] P. N. Brown, R. D. Falgout, J. E. Jones, Semicoarsening multigrid on distributed memory machines, SIAM J. Sci. Comput. 21 (5) (2000) 1823–1834.

[57] W. F. Mitchell, The full domain partition approach to distributing adaptive grids, Applied Numerical Mathematics 26 (8) (1998) 265–275.

[58] M. Griebel, G. Zumbusch, Parallel multigrid in an adaptive PDE solver based on hashing and space-filled curves, J. of Parallel Computation 25 (7) (1999) 827–843.

[59] L. R. Matheson, R. E. Tarjan, Parallelism in multigrid methods: How much is too much?, International J. of Parallel Programming 24 (5) (1996) 397–432.

[60] P. O. Frederickson, O. A. McBryan, Normalized convergence rates for the PSMG method, SIAM J. Sci. Stat. Comput. 12 (1) (1991) 221–229.

[61] D. Gannon, J. Vanrosendale, On the structure of parallelism in a highly concurrent PDE solver, J. Paral. Distrib. Comp. 3 (1) (1986) 106–135.

[62] A. Grama, V. Sarin, A. Sameh, Improving error bounds for multipole-based treecodes, SIAM J. Sci. Comput. 21 (5) (2000) 1790–1803.

[63] D. D. Humphreys, R. A. Friesner, B. J. Berne, A multiple-time-step molecular dynamics algorithm for macromolecules, J. Phys. Chem. 98 (27) (1994) 6885–6892.

Algorithm 1 Pseudo-code of a recursive multi-grid scheme with V-cycle. Code is shown for one processor. N is the number of particles, i the $i^{\text{th}} \in \{0, \ldots, p-1\}$ processors, the grid dimensions $n_x \times n_y \times n_z$, and M denotes the size of the spatial hashing.

main:

(1) anterpolate charges from particles $\left[\lfloor \frac{Ni}{p} \rfloor, \lfloor \frac{N(i+1)}{p} \rfloor\right]$ to the finest charge grid(1), and do local sum over the grid(1) values – step (1);

(2) call **multiscale**(maxLevel, level 1);

(3) interpolate forces in finest grid(1) to particles number $\left[\lfloor \frac{Ni}{p} \rfloor, \lfloor \frac{N(i+1)}{p} \rfloor\right]$ – step (3), Eq. (24);

(4) correct kernel in particles $\left[\lfloor \frac{Mi}{p} \rfloor, \lfloor \frac{M(i+1)}{p} \rfloor\right]$ – step (4), Eq. (16);

(5) compute total energy due to grid points number $\left[\lfloor \frac{n_x n_y n_z i}{p} \rfloor, \lfloor \frac{n_x n_y n_z (i+1)}{p} \rfloor\right]$ – Eq. (23);

multiscale(maxLevel, level k):

(1) if maxLevel = k then

  (a) compute kernel on coarsest grid(maxLevel) points $\left[\lfloor \frac{n_x n_y n_z i}{p} \rfloor, \lfloor \frac{n_x n_y n_z (i+1)}{p} \rfloor\right]$, and do local sum over the grid(maxLevel) values – step (2), Eq. (15);

(2) otherwise

  (a) anterpolate charge grid(k) $\left[\lfloor \frac{n_x n_y n_z i}{p} \rfloor, \lfloor \frac{n_x n_y n_z (i+1)}{p} \rfloor\right]$ to coarser charge grid(k+1), and do local sum over the grid(k+1) values – step (1);
  (b) call **multiscale**(maxLevel, k+1);
  (c) interpolate coarser kernel grid(k+1) $\left[\lfloor \frac{n_x n_y n_z i}{p} \rfloor, \lfloor \frac{n_x n_y n_z (i+1)}{p} \rfloor\right]$ to kernel grid(k) – step (3);
  (d) correct kernel grid(k) $\left[\lfloor \frac{n_x n_y n_z i}{p} \rfloor, \lfloor \frac{n_x n_y n_z (i+1)}{p} \rfloor\right]$, and do global sum over the grid(k) values – step (4), Eq. (17);