Multiscale Fast Summation of Long-Range Charge and Dipolar Interactions

BILHA SANDAK

Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel

Received 10 April 2000; accepted 11 December 2000

ABSTRACT: Here we present a linear order multiscale method for the fast summation of long range forces in a system consisting of a large number of charge and dipolar particles. For a N-body system, our algorithm requires an order of work that is proportional to $O(N)$, in comparison to order $O(N^2)$ of the direct pairwise computation. Our method is demonstrated on two-dimensional homogeneous point-charge and dipolar systems, and a combined heterogeneous particle system, for the calculation of the induced electrostatic potential and energy. The electrostatic interaction is decomposed into a local part and a smooth part. The method thus, has several potential advantages over other $O(N \log N)$ or $O(N)$ techniques, especially for calculation with moving particles or implicit charges locations. This approach is beneficial to large-scale problems such as molecular statics, molecular dynamics, equilibrium statistics (Monte-Carlo simulations), molecular docking, and in areas such as magnetism and astrophysics. J Comput Chem 22: 717–731, 2001

Keywords: multiscale methods; fast summation; dipolar system; long-range forces; electrostatic potential and energy

Introduction

We present an approach for summing long-range *many-body* interactions in linear time. Examples where many-body computations are essential, are in the research fields of particle and biomolecular simulations. There, the pairwise summation of interactions is required for the evaluation of the electrostatic potential, force, and total energy. In a system of N particles, all interactions should be taken into account for evaluating Coulombic or gravitational potentials, because these potentials have a slow decay. The direct computation therefore involves an order of $O(N^2)$ computer operations. Systems where N is huge may even manifest the question of feasibility of the N-body computation. This computational drawback arises for example, in molecular statics where the most stable conformation of the molecular structure is sought by

Correspondence to: B. Sandak; e-mail: billie@wisdom.weizmann.ac.il

Contract/grant sponsor: Israel Science Foundation; contract/grant number: 696/97

Contract/grant sponsor: Carl F. Gauss Minerva Center for Scientific Computation (Weizmann Institute of Science)

calculating the lowest energy. In molecular dynamics, the force, obtained from the gradient of the potential function, is calculated for governing the particle's trajectory. In equilibrium statistics, Monte-Carlo simulations are carried out for obtaining configurations of the particles and their observed average properties at equilibrium, with transition probabilities depending on the total energy change in the system. In biomolecular systems, such as in molecular and protein docking,[1,2] one may require to calculate the electrostatic potential of the interacting molecules to facilitate the notion of electrostatic docking. We thus propose our *multiscale method* for reducing the complexity of this long-range many-body computation to a linear complexity. That is, for heterogeneous N-body particle system composed of point-charges and dipolar particles, our computation is carried out in the order of $O(N)$ computer operations.

The general approach for our *multiscale* computations in the context of integral transforms, many-body problems and dense-matrix multiplication has been initially devised by Brandt.[3–5] This is part of the general multiscale paradigm in solving problems in the physical space, which yield various types of fast algorithms in many areas of science and engineering (see the recent survey[6]).

Previous methods to reduce the computational cost of the N-body to $O(N \log N)$ are presented in refs. 7–9, and 10. The fast multipole expansion method (FMM) of ref. 11 achieves the complexity of $O(N)$, for the fast summation of Coulombic fields in a system of point-charges. In ref. 12, the cell multipole method, an adaptation of FMM, is implemented for dipolar domains. Additional implementations of fast algorithms for solving N-body problems can be found in survey.[10]

We have designed and implemented our multiscale algorithm to account for heterogeneous systems, i.e., involving both dipolar particles as well as point charges. Furthermore, our "mathematical engines" and algorithm's architecture are general to allow the fast evaluation and summation of physical functions other than potential-type ones. The algorithm allows efficient multiscale particle movements, which can facilitate the acceleration of Monte-Carlo simulations and energy minimization processes: see the Discussion section.

The Computational Problem

Our basic computational task is the fast calculation of the electrostatic potential induced by a system composed of point charges and dipolar particles, arbitrarily positioned in space. The point charges may represent atoms, whereas the dipoles represent polar particles, for example, water molecules. For the simplicity of the exposition, our method is demonstrated for the two-dimensional (2D) case. Assume $\mathbf{u} = (u^X, u^Y)$ is a point in the plane. $V(\mathbf{u})$ denotes the potential at point-$\mathbf{u}$, and is computed as the sum of potentials induced by the dipoles and point charges in the system. For a system composed of n point charges and m dipoles, the potential at point $\mathbf{u}$ is

$$V(\mathbf{u}) = \sum_{i=1}^{n} -\log\big(|\mathbf{u} - \mathbf{x}_i|\big) q_i + \sum_{j=1}^{m} \boldsymbol{\mu}_j \cdot \nabla_j \big(-\log\big(|\mathbf{u} - \mathbf{y}_j|\big)\big), \quad (1)$$

where $\mathbf{x}_i$ is the position vector of point charge numbered i, $\mathbf{x}_i = (x_i^X, x_i^Y)$; q_i is the ith charge; $|\mathbf{u} - \mathbf{x}_i|$ is the distance between point charge-i and point-$\mathbf{u}$, $\mathbf{y}_j$ is the position vector of dipole numbered j, $\mathbf{y}_j = (y_j^X, y_j^Y)$; $\boldsymbol{\mu}_j = (\mu_j^X, \mu_j^Y) = \mu_j(\cos\phi_j, \sin\phi_j)$ is its moment vector, where $\mu_j = |\boldsymbol{\mu}_j|$ is its moment magnitude and ϕ_j is its orientation; $|\mathbf{u} - \mathbf{y}_j|$ is the distance between dipole-j and point-$\mathbf{u}$ (see Fig. 1).

Examples of computational tasks that stem from the fast calculation of the potential and are motivated by the many-body problems previously mentioned, are the computation of potential differences and force (potential gradient) calculations. Here we consider the computational task of computing E, the total electrostatic energy of the heterogeneous

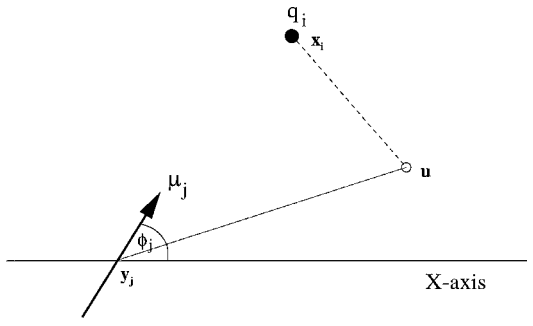

FIGURE 1. A schematic illustration of a point-charge numbered i and a dipole number j located at positions $\mathbf{x}_i$ and $\mathbf{y}_j$, respectively. The distance between point charge-i and point P positioned at $\mathbf{u}$ is $|\mathbf{u} - \mathbf{x}_i|$; the distance between dipole-j and point P is $|\mathbf{u} - \mathbf{y}_j|$. q_i is the charge of point i. ϕ_j is the orientation of dipole-j and μ_j is its moment magnitude, and $\boldsymbol{\mu}_j$ is its moment vector.

point-charge and dipolar system:

$$E = \frac{1}{2}\left(\sum_{k=1}^{n} q_k V_k + \sum_{l=1}^{m} \boldsymbol{\mu}_l \cdot \nabla V_l\right), \qquad (2)$$

where V_k and ∇V_l are the potential at the location of point-charge-k and the gradient of the potential at the position of dipole-l, respectively, *excluding their self contribution*. In our heterogeneous system, we, therefore, have three types of interactions, charge–charge, charge–dipole, and dipole–dipole giving,

$$\begin{aligned} E = \frac{1}{2}\Bigg(\sum_{k=1}^{n} q_k \Bigg[& \sum_{i=1,\, i\neq k}^{n} \big(-\log(|\mathbf{x}_k - \mathbf{x}_i|)\big) q_i \\ & + \sum_{j=1}^{m} \boldsymbol{\mu}_j \cdot \nabla_j \big(-\log(|\mathbf{x}_k - \mathbf{y}_j|)\big) \Bigg] \\ & + \sum_{l=1}^{m} \boldsymbol{\mu}_l \cdot \Bigg[\sum_{i=1}^{n} \nabla_l \big(-\log(|\mathbf{y}_l - \mathbf{x}_i|)\big) q_i \\ & + \sum_{j=1,\, j\neq l}^{m} \nabla_l \boldsymbol{\mu}_j \cdot \nabla_j \big(-\log(|\mathbf{y}_l - \mathbf{y}_j|)\big) \Bigg] \Bigg). \end{aligned} \qquad (3)$$

We achieve this task by an $O(m+n)$ calculation in comparison to the order of $(m+n)^2$, which is required by the direct computation. The evaluation is done to a certain degree of accuracy, ϵ, i.e., an error of up to ϵ is allowed in each of the summed terms. Taking this accuracy into account, our complexity is actually an $O((m+n)(\log \frac{1}{\epsilon})^d)$ computation, where d is the dimension of the space (here $d = 2$).

The Algorithm

OVERVIEW

General Description

The first stage of our algorithm is to reduce the number of charges and dipoles to a manageable small set of *super-charges* and *super-dipoles*, yielding a fast potential calculation. On the plane at which the particles are located, a rectangular grid is placed. The grid is uniform, i.e., its mesh size h is constant in each grid direction and its "density" is comparable to the particle density. The super-charges (dipoles) are created by aggregating the arbitrarily located charges (dipoles) into collections positioned at the gridpoints. This procedure can be carried out recursively for increasingly coarser grids, as depicted in Figure 2, stage (1). Each coarser grid is obtained by omitting every other gridline from the finer grid. Henceforth, this stage is referred to as the *fine-to-coarse* stage. The recursion proceeds until

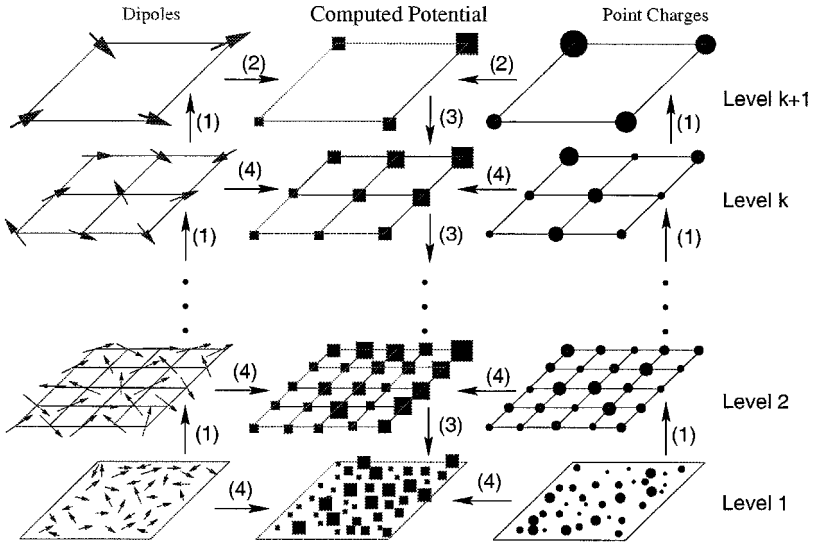

FIGURE 2. A multilevel scheme of the multigrid algorithm for the fast summation of long-range forces exemplified by the computation of the potential induced by arbitrarily positioned point-charges and dipoles.

the number of super-charges (dipoles) is so small that the calculation of their potential costs little compared with the whole algorithm. The potential induced by the super-charges and super-dipoles is computed directly at the gridpoints on the "potential grid" [Fig. 2, stage (2)]. The result of the coarse level potential evaluation is propagated recursively by interpolation via intermediate finer grids (the *coarse-to fine* stage), adding local corrections at each grid (as will be explained hereafter). This is carried out until reaching the finest level, that is, the locations at which the potential values are required. This is displayed in Figure 2, stages (3) and (4).

Potential Kernels—Problem and Remedy

Assume first that each of the potential *kernels* $G(\mathbf{u},\mathbf{z})$ (like $\log|\mathbf{u}-\mathbf{z}|$ or its gradients) were smooth throughout (i.e., having continuous first derivatives) as a function of both the particle positions $\mathbf{z}$, and the location $\mathbf{u}$ at which the potential is to be computed. The super-particles can then be defined so that their potential field agree with that of the original particles, except for a small *aggregation error*, its smallness depending only on the smoothness of $G(\mathbf{u},\mathbf{z})$, as a function of $\mathbf{z}$. Likewise, the values of the potential computed at the lattice points can be interpolated to any other position, with small *interpolation error*, its smallness depending only on the smoothness of $G(\mathbf{u},\mathbf{z})$ as a function of $\mathbf{u}$.

However, this is not quite the case here, because the potential kernels are *not* smooth throughout. They have unbounded derivatives as $\mathbf{u}$ tends to $\mathbf{z}$. Hence, the error caused by aggregation/interpolation is unbounded. Nevertheless, the smoothness of the kernels indefinitely increases with $|\mathbf{u}-\mathbf{z}|$, that is, the kernel becomes smoother as the distance between the two points grows; such kernels are called *asymptotically smooth*. They can be turned into sufficiently smooth kernels $G_{\text{smooth}}(\mathbf{u},\mathbf{z},s)$ by "softening," i.e., by being modified only in a local range $|\mathbf{u}-\mathbf{z}| \le s$, where s is a suitably chosen "softening radius (distance)" (see Fig. 3).

Assume a two-level system, i.e., the original plane on which the particles are scattered and a single grid level placed on it as depicted in Figure 2 (composed of Level 1 and Level 2). Defining a "potential grid" with mesh size similar to that of the aggregation grids, the potential values are first computed at its gridpoints, using the *softened* kernel [see Fig. 2, stage (2)], and then they are interpolated to the desired particle, locations [Fig. 2, stage (3)]. However, *corrections* of the potential values are required because softened potential has been used instead of the exact one. The correction of the kernel at any point $\mathbf{u}$ is simply $\sum_i [G(\mathbf{u},\mathbf{z}_i) - G_{\text{smooth}}(\mathbf{u},\mathbf{z}_i,s)]$, summed only over all particles i such that $|\mathbf{u}-\mathbf{z}_i| \le s$ [Fig. 2, stage (4)]. The original kernel is, therefore, said to be decomposed into a *smooth part* $G_{\text{smooth}}(\mathbf{u},\mathbf{z},s)$ and *local part* $[G(\mathbf{u},\mathbf{z}) - G_{\text{smooth}}(\mathbf{u},\mathbf{z},s)]$. The smooth part of the potential is obtained by interpolating from the grid level, while the local part is then added as a correction.

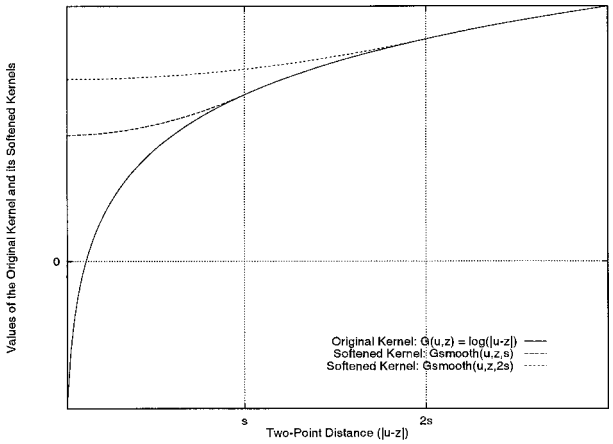

FIGURE 3. A plot of the original kernel $\log|\mathbf{u}-\mathbf{z}|$ and two of its softening kernels $G_{\text{smooth}}(\mathbf{u},\mathbf{z},s)$ and $G_{\text{smooth}}(\mathbf{u},\mathbf{z},2s)$ having the softening distance of s and of $2s$, respectively. For $|\mathbf{u}-\mathbf{z}| > s$ and $|\mathbf{u}-\mathbf{z}| > 2s$ the softened kernels coincide with the original kernel $\log|\mathbf{u}-\mathbf{z}|$. For $|\mathbf{u}-\mathbf{z}| < s$ and $|\mathbf{u}-\mathbf{z}| < 2s$, two softening *polynomials* of degree-2 are defined $G_s(|\mathbf{u}-\mathbf{z}|) = \log s + A_0 + A_1(|\mathbf{u}-\mathbf{z}|/s)^2$ having the softening distance of s, and $G_{2s}(|\mathbf{u}-\mathbf{z}|) = \log 2s + A_0 + A_1(|\mathbf{u}-\mathbf{z}|)^2/(2s)^2$ with the softening distance of $2s$. The softening polynomials coefficients are $A_0 = -0.5$ and $A_1 = 0.5$.

Recursion Considerations

If we wish to decrease the aggregation and interpolation errors, a smoother function should be defined for the softeners by enlarging the softening distance s, or by using a higher interpolation order (as will be explained in detail in the next section), or both. The mesh size h can also be decreased, however, this will result in increasing the number of gridpoints and thus enlarging the computational work required. If h is increased, the interpolation/anterpolation errors will rise. Thus, the mesh size h is chosen such that it is comparable to the particles density. If s is increased, however, to be too large, there would be too many local corrections to be done as the number of particles involved

in this process grows. A remedy for this, which keeps the particle density and the computational cost intact, is to define an additional coarser grid level by doubling the mesh size h, on which s is enlarged, usually doubled (see Fig. 3). A softened kernel $G_{\text{smooth}}(\mathbf{u}, \mathbf{z}, 2s)$ is thus obtained. Although the local part at which the corrections should be carried out is larger, the number of corrections remains the same, because the corrections themselves are carried out on a coarser level, involving the same number of gridpoints. The entire algorithm for calculating the potential of the system can thus be recursively defined. As such, the particle aggregation can be carried out recursively for increasingly coarser grids, as depicted in Figure 2, stage (1). At each coarser grid a smoother kernel is defined by enlarging the softening vicinity (usually it is doubled). Thus, the many-times softened potential induced by the super-charges and super-dipoles is computed directly at the gridpoints [Fig. 2, stage (2)]. The coarse level potential evaluation is propagated recursively by interpolation to levels of intermediate finer grids [Fig. 2, stage (3)]. The appropriate corrections at each grid are then added. The kernel correction at any gridpoint $\mathbf{U}$ of level k is done by adding $\sum_i [G_{\text{smooth}}(\mathbf{U}, \mathbf{Z}_i, 2^k s) - G_{\text{smooth}}(\mathbf{U}, \mathbf{Z}_i, 2^{k+1} s)]$, summed only over all super-particles i such that $|\mathbf{U} - \mathbf{Z}_i| \leq 2^{k+1} s$ [Fig. 2, stage (4), level k]. This is carried out until reaching the finest level, that is, interpolating and correcting the potential onto the locations at which these values are required.

Note that the amount of work is decreased fourfold on each of the coarsened grid levels. Because the work decreases geometrically on increasingly coarser (uniform) levels, the total cost of this algorithm is comparable to the work being done at the finest levels, which is $O(m + n)$, assuming the number of locations where the potential values are needed is comparable to the number of particles.

Interpolation and aggregation are the underlying processes of the algorithm, the latter being performed by *anterpolation,* i.e., the adjoint of interpolation. These processes are outlined next, followed by a detailed description of the two main stages of the method, the fine-to-coarse (anterpolation) and the coarse-to-fine (interpolation) stages.

INTERPOLATION AND ANTERPOLATION

Interpolation is employed wherever a value of a suitably smooth function V is to be approximated, given a set of points where the values of the function are known. In one dimension, the number of points to interpolate from defined as the *interpolation order* and denoted by p. A polynomial of degree $p - 1$ is constructed to approximate the function at the required target points (the general form of a polynomial is $f(x) = \sum_{i=0}^{p-1} a_i x^i$, a_i are constants, $a_i \neq 0$). For higher dimensions, the interpolation is done one direction at a time. Hence, in 2D, the number of of points to interpolate from is p^2. An example of linear interpolation ($p = 2$) is depicted in Figure 4a where at each gridpoint the value of the function is known, for example, the potential V. Figure 4b shows how the interpolation is done, that is, by using the two interpolation points in each grid direction. First, on the gridlines around the target point, two intermediate points are interpolated to by the appropriate gridpoints at the X position of the target point on the gridlines [steps (1) and (2) in Figure 4b]. Next, the two intermediate points are used as the two interpolation points for the target point, as depicted by step (3). Note, that this technique is symmetrical in any grid direction (X or Y). We employ central interpolation, i.e., the interpolation order is even (e.g., $p = 2$, $p = 6$) and the interpolation points are chosen symmetrically around the target point. The requirement that uniform grids should be used, stems mainly for the purpose of having simple inter-

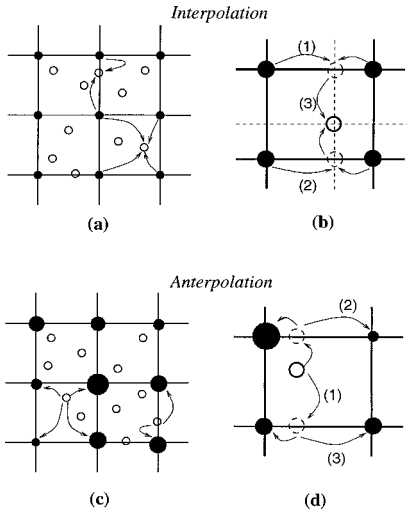

FIGURE 4. An illustration of 2D linear interpolation and anterpolation processes of order 2 ($p = 2$). (a) Each target point (white circles) is interpolated from four gridpoints (black circles). If the target point is along a gridline, it is interpolated from two gridpoints. (b) Efficient interpolation is done by using the p interpolation points in each grid direction, denoted as steps (1) to (3). (c) Each arbitrary point (white circles) is aggregated to four gridpoints (black circles) or less (depending on its location). (d) Anterpolation of an arbitrary point is done in each grid direction (steps (1) to (3)).

polation formulas. Here, we use the Lagrange form for the interpolation polynomial. In one dimension, the interpolated value at a target point x, that is, the polynomial value at the point, is denoted by $P(x)$ and computed by

$$P(x) = \sum_{k=0}^{p} V(x_k) l_k(x), \qquad l_k(x) = \prod_{i=0,\ i\neq k}^{p} \frac{(x - x_i)}{(x_k - x_i)}, \tag{4}$$

where p is the interpolation order; V is the interpolated function; $x_0, \ldots, x_p$ are the points to interpolate from (at which V is known, e.g., the potential at the gridpoints), and l_k's are the Lagrange coefficients referred to as the weights of the interpolation (their sum for the target point x_k is 1). The approximation itself has an $O(h^p V^{(p)})$ error, where h is the grid's mesh size and $V^{(p)}$ is the maximal p-order derivative of V. At the grid boundaries, the p-order interpolation is carried out by defining appropriate number of "ghost cells," i.e., external neighboring points to the grid that create a frame around it. These enable sound interpolation to all points required within the grid boundaries.

Interpolation of a gradient of a function is also required for target points x, for example, the interpolation of the gradient $\nabla V(x)$ of the potential V [appearing in eq. (2)]. This is done by using the derivative of the interpolation polynomial, denoted by $P'(x)$, and computed by

$$P'(x) = \sum_{k=0}^{p} V(x_k) l'_k(x), \qquad l'_k(x) = \left[\prod_{i=0,\ i\neq k}^{p} \frac{(x - x_i)}{(x_k - x_i)}\right]\left[\sum_{i=0,\ i\neq k}^{p} \frac{1}{(x - x_i)}\right]. \tag{5}$$

Note that for any given x, both $P(x)$ and $P'(x)$ can be calculated in just $O(p)$ operations. In 2D, the two components of ∇V are $(\partial V/\partial X, \partial V/\partial Y)$. $\partial V/\partial X$ is approximated by employing P' for the interpolation in the X direction and P for the subsequent Y direction interpolation. $\partial V/\partial Y$ is approximated by first using P in the X direction interpolation, and then P' in the interpolation of the Y direction.

Anterpolation is the procedure carried out for aggregating the particles to super-particles. It is defined as the adjoint of interpolation, having weights of interpolation and an order that is also defined by p. Every arbitrary target point distributes its value, for example, its charge, to $p \times p$ gridpoints surrounding it (p points in each grid direction). In each direction, the anterpolation (aggregation) is done according to the weights (coefficients) $l_k(x)$ computed in eq. (4). This is shown in Figure 4c and d. The anterpolation "justification" stems from the following: assume we wish to *directly* compute the potential at point $\mathbf{u}$ induced by particles arbitrarily positioned at $\mathbf{z}_i$. We use the notation $G(\mathbf{u}, \mathbf{z}_i)$ for the exact kernel, $G_{\text{smooth}}(\mathbf{u}, \mathbf{z}_i)$ for the *softened* kernel, and $c(\mathbf{z}_i)$ for denoting the charge at the particle's location $\mathbf{z}_i$. The potential value is $V(\mathbf{u}) = \sum_i G(\mathbf{u}, \mathbf{z}_i) c(\mathbf{z}_i)$. Now, assume we wish to obtain $V(\mathbf{u})$ by *interpolation* using the smoothness properties of G_{smooth} both as a function of $\mathbf{u}$ and $\mathbf{z}_i$. This means that G_{smooth} can be interpolated from gridpoints $\mathbf{Z}_k$ with a bounded error, that is, $V(\mathbf{u}) \approx \sum_i \sum_k l_{ik} G_{\text{smooth}}(\mathbf{u}, \mathbf{Z}_k) c(\mathbf{z}_i)$, where the l_{ik}s are the weights of interpolation. By *changing* the order of summation we obtain that $V(\mathbf{u}) \approx \sum_k G_{\text{smooth}}(\mathbf{u}, \mathbf{Z}_k) \sum_i l_{ik} c(\mathbf{z}_i)$. We denote $C(\mathbf{Z}_k) = \sum_i l_{ik}(\mathbf{z}_i)$, that is, this is the interpolation required for transferring particles to super-particles, i.e., anterpolating them to their positions on the grid. We thus obtain $V(\mathbf{u}) \approx \sum_k G_{\text{smooth}}(\mathbf{u}, \mathbf{Z}_k) C(\mathbf{Z}_k)$.

In view of the symmetry in the smoothness of the softened kernels both as a function of the particles locations and the locations onto where the potential is to be evaluated, the aggregation and potential grids have the same mesh sizes. Also, the interpolation and anterpolation are of same order p. Note that in 2D for any point x, both interpolation (either for the function values or for its gradient) and anterpolation can be performed in $O(p^2)$. In case the points to anterpolate from, or interpolate to, themselves reside on a uniform grid (as at coarser levels of the algorithm), the interpolation and anterpolation can be executed in $O(p)$ operations per point. Thus, the number of operations required by the anterpolation and interpolation processes on the finest level at which the particles are scattered is $O(mp^2 + np^2)$. At each coarser level these processes cost $O(Np)$, where N is the number of gridpoints on that level. For the interpolation to be done with an $O(\epsilon)$ error, p should be in the order of $O(\log \frac{1}{\epsilon})$. Thus, the complexity of the algorithm, as determined by the degree of accuracy is an $O((m + n)(\log \frac{1}{\epsilon})^2)$ computation.

FINE-TO-COARSE STAGE

At each level, we have three uniform grids: one grid contains the aggregates of charges (the super-charges); the other two grids are assigned to the aggregates of the two components of the dipoles (the super-dipoles). Recall that dipole-j moment can be represented as a vector $\boldsymbol{\mu}_j = (\mu_j^X, \mu_j^Y) = (\mu_j \cos\phi_j, \mu_j \sin\phi_j)$. The dipole moments in the

X-direction, μ_j^X, are aggregated on one grid, and the dipole moments in the Y-direction, μ_j^Y, on the other. The X and Y-components of each super-dipole $\boldsymbol{\mu}_D$ are the summation of the X and Y-components of the dipoles that were aggregated to the corresponding coarse gridpoint. That is,

$$\boldsymbol{\mu}_D = \left(\sum_{j=1}^{k} \omega_j \mu_j \cos\phi_j, \sum_{j=1}^{k} \omega_j \mu_j \sin\phi_j\right) = \left(\mu_D^X, \mu_D^Y\right), \tag{6}$$

where k is the number of contributing dipoles and ω_js are the anterpolation weights. The super-dipole's magnitude is, $|\boldsymbol{\mu}_D| = \sqrt{(\mu_D^X)^2 + (\mu_D^Y)^2}$ and its orientation is $\Phi_D = \arctan(\mu_D^Y/\mu_D^X)$.

COARSE-TO-FINE STAGE

To recover the potential values from a coarse grid to a fine grid (or to their final locations), interpolation is required. As mentioned in the Overview section, the potential kernels we use are asymptotically smooth, i.e., they are not suitably smooth for direct interpolation throughout. Assume we wish to interpolate the potential at point $\mathbf{U}$, to obtain its value as induced by the super-charge and dipolar particles positioned at gridpoints $\mathbf{X}_i$ $(i = 1, \ldots, n)$ and $\mathbf{Y}_j$ $(j = 1, \ldots, m)$, respectively. The interpolation can still be done with an $O(\epsilon)$ error, with interpolation order $p = O(\log \frac{1}{\epsilon})$, for distances $|\mathbf{u} - \mathbf{X}_i| \geq O(h)$, $|\mathbf{u} - \mathbf{Y}_j| \geq O(h)$, where h is the interpolation-grid mesh size. For the distances $|\mathbf{u} - \mathbf{X}_i|$ and $|\mathbf{u} - \mathbf{Y}_j|$, which are smaller than $O(h)$, the interpolation error is unbounded for the potential computed with the exact kernel. Therefore, a softened kernel is used instead to sustain the bounded interpolation error $O(\epsilon)$. Thus, corrections should be made to the interpolated potential value at $\mathbf{u}$, to account for the erroneous contributions of the particles in this softened vicinity.

To keep the computational cost of the corrections comparable to the cost of the anterpolation and interpolation processes, which are in the order $(m + n)p^2$, two requirements should be fulfilled: (i) In the vicinity of the point $\mathbf{u}$ where the interpolated value should be corrected, the number of particles participating in the correction should be on the average $O(p^2)$. Once achieved, this is automatically sustained by the algorithm, because we have a sequence of levels with bounded coarsening ratios, for example, 1:2. (ii) The correction should cost $O(1)$ operations for each of the particles in the above vicinity.

We tackle this issue by using the *softened kernels* as described above. A softened kernel is defined to be equal to the original kernel except in a neighborhood of radius s, $s = O(h)$, around the singularity. The softened kernel (*softener*) for $\log|\mathbf{x} - \mathbf{y}|$, $\mathbf{x} = (x^X, x^Y)$, $\mathbf{y} = (y^X, y^Y)$ is (adopted from ref. 13):

$$G_{\text{smooth}}(\mathbf{x}, \mathbf{y}, s) = \begin{cases} G_s(|\mathbf{x} - \mathbf{y}|) & \text{if } |\mathbf{x} - \mathbf{y}| \leq s \\ \log|\mathbf{x} - \mathbf{y}| & \text{otherwise,} \end{cases}$$
$$G_s(|\mathbf{x} - \mathbf{y}|) = \log s + \sum_{j=0}^{p-1} A_j \left(\frac{|\mathbf{x} - \mathbf{y}|}{s}\right)^{2j}, \tag{7}$$

where p is the interpolation order, $G_s(|\mathbf{x} - \mathbf{y}|)$ is a $(p-1)$-degree polynomial of $|\mathbf{x} - \mathbf{y}|^2$, and its coefficients A_js are independent of s. They are determined by requiring G_{smooth} to be continuous and have continuous derivatives up to order $p-1$ (hence, that $A_0 = -\sum_{j=1}^{p-1} A_j$). Figure 3 depicts the softened kernel $G_{\text{smooth}}(\mathbf{x}, \mathbf{y}, s)$ for $\log|\mathbf{x} - \mathbf{y}|$, and its softening scale s. The softened kernel, $G_{\text{smooth}}(\mathbf{x}, \mathbf{y}, s)$ is constructed such that it is everywhere suitably smooth on scale-s (in both X and Y directions). This means that G_{smooth} can be approximated up to an error smaller than ϵ by a p-order interpolation from G_{smooth}s values on any uniform grid with mesh size h (or smaller). That is, $(\gamma h)^p |G_{\text{smooth}}^{(p)}(\mathbf{x}, \mathbf{y}, s)| \leq O(\epsilon)$ for some $p = O(\log \frac{1}{\epsilon})$, where γ depends on the geometry of the interpolation (here, $\gamma = \frac{1}{2}$ for central interpolation), and $|G_{\text{smooth}}^{(p)}(\mathbf{x}, \mathbf{y}, s)|$ is the maximal p-order derivative of G_{smooth} with respect to any $\mathbf{x}$ or $\mathbf{y}$ component.

Next, we construct the softener for $\nabla_x \log|\mathbf{x} - \mathbf{y}|$, used in the calculation of the potential induced by the dipolar particles. Our construction is based on the observation that a softener for the derivative of a function can be defined as the derivative of a softener for that function. Hence $G'_{\text{smooth}}(\mathbf{x}, \mathbf{y}, s)$, the softener for $\nabla_x \log|\mathbf{x} - \mathbf{y}|$ is the gradient of $G_{\text{smooth}}(\mathbf{x}, \mathbf{y}, s)$, i.e., $G'_{\text{smooth}}(\mathbf{x}, \mathbf{y}, s) = \nabla_x G_{\text{smooth}}(\mathbf{x}, \mathbf{y}, s)$. It is constructed as follows,

$$\begin{aligned} &G'_{\text{smooth}}(\mathbf{x}, \mathbf{y}, s) \\ &= \begin{cases} \left(\dfrac{\partial G_s(|\mathbf{x} - \mathbf{y}|)}{\partial x^X}, \dfrac{\partial G_s(|\mathbf{x} - \mathbf{y}|)}{\partial x^Y}\right) & \text{if } |\mathbf{x} - \mathbf{y}| \leq s \\ \left(\dfrac{\partial \log|\mathbf{x} - \mathbf{y}|}{\partial x^X}, \dfrac{\partial \log|\mathbf{x} - \mathbf{y}|}{\partial x^Y}\right) & \text{otherwise,} \end{cases} \\ &= \begin{cases} G'_s(|\mathbf{x} - \mathbf{y}|)|\mathbf{x} - \mathbf{y}|^{-1} & \\ \quad \times \left(x^X - y^X, x^Y - y^Y\right) & \text{if } |\mathbf{x} - \mathbf{y}| \leq s \\ |\mathbf{x} - \mathbf{y}|^{-2}\left(x^X - y^X, x^Y - y^Y\right) & \text{otherwise,} \end{cases} \\ &G'_s(|\mathbf{x} - \mathbf{y}|) = \frac{1}{s}\sum_{j=1}^{p-1} A_j(2j)\left(\frac{|\mathbf{x} - \mathbf{y}|}{s}\right)^{2j-1}. \end{aligned} \tag{8}$$

Note that the *softening distance* s in $G'_{\text{smooth}}(\mathbf{x},\mathbf{y},s)$ does not have to be the same s as in $G_{\text{smooth}}(\mathbf{x},\mathbf{y},s)$ (in our current implementation it is).

Assume the number of levels we have in our algorithm is two: the fine level where the particles are arbitrarily located, and a single grid "above" it (see Fig. 2 lower two levels). Having aggregated the charges and the dipoles at the gridpoints, we directly compute the potential at the gridpoints using the softened kernels (7) and (8) for the potential of the aggregated charges and dipoles, respectively. The potential values are then interpolated to the fine level, for example, to an arbitrary point $\mathbf{u}$. A correction is required to the interpolated value $\tilde{V}(\mathbf{u})$, to account for the difference between the exact potential and the softened potential of each particle at distance less then s from $\mathbf{u}$. The corrected interpolated value $V(\mathbf{u})$ is therefore obtained by

$$\begin{aligned} V(\mathbf{u}) = \tilde{V}(\mathbf{u}) &+ \sum_{0<|\mathbf{u}-\mathbf{x}_i|<s} -\big[\log\big(|\mathbf{u}-\mathbf{x}_i|\big) - G_s\big(|\mathbf{u}-\mathbf{x}_i|\big)\big]q_i \\ &+ \sum_{0<|\mathbf{u}-\mathbf{y}_j|<s} -\left[\frac{1}{|\mathbf{u}-\mathbf{y}_j|} - G'_s\big(|\mathbf{u}-\mathbf{y}_j|\big)\right] \\ &\quad \times \boldsymbol{\mu}_j \cdot \frac{(\mathbf{u}-\mathbf{y}_j)}{|\mathbf{u}-\mathbf{y}_j|}. \end{aligned} \tag{9}$$

Every charge is aggregated from its arbitrary location onto the super-charges grid. These aggregates participate in the potential computation. Subsequently, the potential value is interpolated back to the required particle position. However, there is an erroneous contribution of the particle's aggregates to the potential value at the particle's location, because its own charge should not be included in the computation. This self-contribution is, therefore, removed from the potential value of every particle. This is done by computing the potential induced by charge-i on a $p \times p$ grid having the same mesh size as the original grid, i.e., h. The self-potential value is interpolated to the location of charge-i and then subtracted from the system's potential at that point. That is, for $\mathbf{x}_i = \mathbf{u}$ [of eq. (9)], the interpolated value of the potential induced by charge-i is subtracted from $V(\mathbf{u})$. We call this procedure *self-correction*.

A sequence of levels of increasingly coarser uniform grids as illustrated in Figure 2 is built for recursively applying the algorithm. To construct suitably smooth kernels at all coarsened grids so that the number of corrections will be kept the same, the softening distance s is doubled in accordance with the doubling of the mesh size (see Fig. 3). Equation (7) is rewritten considering the level k we are in, by multiplying s by 2^k ($k = 0, 1, \ldots$) to obtain,

$$\begin{aligned} G^k_{\text{smooth}}\big(\mathbf{x},\mathbf{y},2^k s\big) &= \begin{cases} G_{2^k s}(|\mathbf{x}-\mathbf{y}|) & \text{if } |\mathbf{x}-\mathbf{y}| \le 2^k s \\ \log|\mathbf{x}-\mathbf{y}| & \text{otherwise,} \end{cases} \\ &= G_{\text{smooth}}\big(\mathbf{x},\mathbf{y},2^k s\big). \end{aligned} \tag{10}$$

Similarly, the kernel softener for $\nabla_x \log|\mathbf{x}-\mathbf{y}|$ at the level number k, i.e.,

$$\begin{aligned} &G'^k_{\text{smooth}}\big(\mathbf{x},\mathbf{y},2^k s\big) \\ &= \begin{cases} G'_{2^k s}(|\mathbf{x}-\mathbf{y}|)|\mathbf{x}-\mathbf{y}|^{-1} \\ \quad \times \big(x^X - y^X, x^Y - y^Y\big) & \text{if } |\mathbf{x}-\mathbf{y}| \le 2^k s \\ |\mathbf{x}-\mathbf{y}|^{-2}\big(x^X - y^X, x^Y - y^Y\big) & \text{otherwise,} \end{cases} \\ &= G'_{\text{smooth}}\big(\mathbf{x},\mathbf{y},2^k s\big). \end{aligned} \tag{11}$$

The correction at level k is done by adding the difference between the kernel softener of the current level k and that of the coarser level above it (level $k+1$) to the value $\tilde{V}_k(\mathbf{u})$ interpolated from that coarser level, computed for all points within radius $2^{k+1}s$:

$$\begin{aligned} V_k(\mathbf{u}) = \tilde{V}_k(\mathbf{u}) &+ \sum_{0<|\mathbf{u}-\mathbf{x}_i|<2^{k+1}s} -\big[G^k_{\text{smooth}}\big(\mathbf{x}_i,\mathbf{u},2^k s\big) \\ &\quad - G_{2^{k+1}s}\big(|\mathbf{u}-\mathbf{x}_i|\big)\big]q_i \\ &+ \sum_{0<|\mathbf{u}-\mathbf{y}_j|<2^{k+1}s} -\big[G'^k_{\text{smooth}}\big(\mathbf{y}_j,\mathbf{u},2^k s\big) \\ &\quad - G'_{2^{k+1}s}\big(|\mathbf{u}-\mathbf{y}_j|\big)\big]\boldsymbol{\mu}_j \cdot \frac{(\mathbf{u}-\mathbf{y}_j)}{|\mathbf{u}-\mathbf{y}_j|}. \end{aligned} \tag{12}$$

For carrying out the computation of the total energy of the system [eq. (2)], the gradient of the potential should also be evaluated, for example, at point $\mathbf{u}$. This should be done only at the finest grid, that is, from the finest grid to the required location of $\mathbf{u}$. We first compute $\nabla\tilde{V}(\mathbf{u})$ by evaluating the derivative of the interpolation polynomial at point $\mathbf{u}$ as explained above. Because the finest grid "holds" the smooth part of the potential evaluation, respective corrections should be done here for obtaining $\nabla V(\mathbf{u})$. That is,

$$\begin{aligned} \nabla V(\mathbf{u}) = \nabla\tilde{V}(\mathbf{u}) &+ \sum_{0<|\mathbf{u}-\mathbf{x}_i|<s} -\left(\frac{1}{|\mathbf{u}-\mathbf{x}_i|} - G'_s\big(|\mathbf{u}-\mathbf{x}_i|\big)\right) \\ &\quad \times q_i \frac{(\mathbf{u}-\mathbf{y}_j)}{|\mathbf{u}-\mathbf{y}_j|} \\ &+ \sum_{0<|\mathbf{u}-\mathbf{y}_j|<s} \left[-\left(\frac{1}{|\mathbf{u}-\mathbf{y}_j|} - G'_s\big(|\mathbf{u}-\mathbf{y}_j|\big)\right)\right. \\ &\quad \times \nabla\boldsymbol{\mu}_j \cdot \frac{(\mathbf{u}-\mathbf{y}_j)}{|\mathbf{u}-\mathbf{y}_j|} \end{aligned}$$

$$- \left(-\frac{1}{|\mathbf{u}-\mathbf{y}_j|^2} - G_s''(|\mathbf{u}-\mathbf{y}_j|) \right) \times \boldsymbol{\mu}_j \cdot \frac{(\mathbf{u}-\mathbf{y}_j)^2}{|\mathbf{u}-\mathbf{y}_j|^2} \Bigg] \quad (13)$$

where $G_s''(|\mathbf{x}-\mathbf{y}|)$ is defined as (for $\mathbf{x} = (x^X, x^Y)$, $\mathbf{y} = (y^X, y^Y)$):

$$G_s'' = \left(|\mathbf{x}-\mathbf{y}|\right) = \frac{1}{s^2} \sum_{j=1}^{p-1} A_j(2j)(2j-1) \left(\frac{|\mathbf{x}-\mathbf{y}|}{s} \right)^{2(j-1)}. \quad (14)$$

Similarly to the self-correction procedure of the charges [as described below eq. (9)], for $\mathbf{y}_j = \mathbf{u}$, the interpolated value of the gradient of the potential induced by dipole-j on a $p \times p$ grid, is subtracted from $\nabla V(\mathbf{u})$ to exclude the self-contribution of the dipole's aggregates to the potential.

Note that the term $2^k s$ in eqs. (10) and (11) can be substituted by the more general expression $2^k s_k$ where s_k is a level-dependent value of the softening distance. Similarly, different values for p can be used on different levels. See the Results section where this is exemplified and discussed. To date, we employ similar sets of ps and ss for the two types of particles. It may be of interest (which is application dependent) to employ different sets of these parameters for dipolar and charge particles.

Results

SYSTEM TYPES AND RUNNING TIMES

We have run our fast summation algorithm for computing the potential, potential gradient, and the total electrostatic energy of the system [eq. (3)]. The three system types tested are composed of arbitrarily positioned point-charges, dipolar particles, and their combination. Figure 5 depicts the typical random scattering of the particles in the heterogeneous particle system. The charges of the points are randomly chosen between -1 and 1. The dipoles orientations are random between 0 and 360 degrees, and their strength (moment magnitude) is of fixed value 0.2. We have calculated the potential and energy for increasing number of particle systems (from 81 to 8,396,802 particles) to measure the effect of the system size and type on the running times and measured errors. We have conducted our fast computation on a sequence of coarsened grids (the finest having mesh size of $h = a$, where a is the average distance between the particles), such that the coarsest grid for each of the particle systems is a 9×9 grid of super-particles. On this coarse grid the direct computation of the electrostatic potential is done. Subsequently, the potential is recursively interpolated (and corrected) to the finest grid, and from there, to the required arbitrary positions of the particles. We call this type of run a *multilevel* run. The notation used throughout this section is summarized in Table I, and more precisely defined

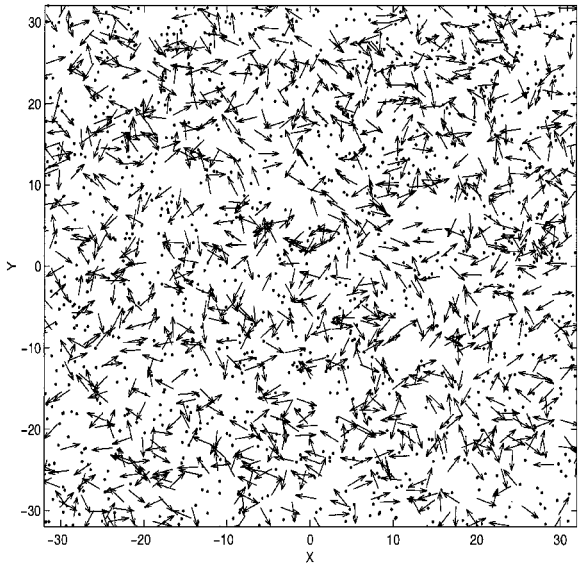

FIGURE 5. A system with typical distribution of arbitrarily located 578 particles, half of which are dipoles and half are point-charges.

TABLE I.
Nomenclature.

Notation	Meaning
n	Number of point charges
m	Number of dipoles
h	Mesh size
a	Average distance between particles
p	Interpolation (anterpolation) order
s	Softening distance (scale)
γ	Constant equal to 0.5 for central interpolation
ϵ	Degree of accuracy
l	Number of multigrid levels
T_d	Running time of the direct (naive) computation (seconds)
T_f	Running time of the fast multiscale algorithm (seconds)
E_{err}	Average energy calculation error per particle interaction
V_{err}	Average potential calculation error per charge interaction
∇V_{err}	Average potential gradient calculation error per dipole interaction

below. The results for the three types of systems are exhibited in Tables II–IV, respectively. The program has been executed on a SGI-R10000 machine (194 MHZ processor) with 1024 Mbytes of memory. m dipoles and n point charges constitute the system, where in the heterogeneous case, there are equal number of dipoles and charges ($m = n$). In all three systems, as the total number of the particles is increased by about four-fold, the running times of the direct (naive) computation is enlarged by about 16-fold (T_d), which is in the order of $O(m^2)$ for the dipolar system, $O(n^2)$ for the charge system and $O((m+n)^2)$ for the heterogeneous system. Whereas, our fast calculation time (T_f) grows only linearly with problem size, i.e., it in the order of $O(m)$, $O(n)$, and $O(m+n)$ with respect to the three systems. The memory requirement of the algorithm is also in the order of the number of particles. This many-body computation is therefore made feasible. For example, the computation of a 8,396,802 heterogeneous particle system is plausible (Table IV), because we achieve this within 21.5 min, where the direct approach would require almost a year.

ERROR CLASSIFICATION AND ERROR MEASUREMENTS

In general, three controllable types of errors affect the accuracy of the calculations. The first two are the anterpolation and interpolation errors that are determined by p, s, and h. For the interpolation/anterpolation to be done with bounded error, s needs to be at least several mesh sizes. The interpolation error is thus dominated by the smoothness properties of the softeners in the local region s of where softening is defined. The third error type is produced from the self-charge contribution of the particles to the potential computation at their positions. The error is removed using the self-correction procedure. The errors are a function of the number of particles in the system, as will be shown hereby.

To evaluate the accuracy of our fast computation in comparison with the direct computation, we define various error measures. The average error of the potential calculation per charge interaction is computed as $V_{\text{err}} = \frac{1}{n(n+m)} \sum_{i=1}^{n} |V_{f_i} - V_{d_i}|$, where V_{f_i} is the potential computed by our fast algorithm at point charge-i, and V_{d_i} is the potential computed directly at that point. Similarly, the average calculation

TABLE II.
***Multilevel* Performance Table for Point-Charge Systems of Varying Number of Charges.**

					Without Self-Correction			With Self-Correction		
n	p	s	l	T_d[a]	T_f[a]	E_{err}	V_{err}	T_f[a]	E_{err}	V_{err}
81	2	4	2	0.00[b]	0.01	$2.18e^{-3}$	$6.54e^{-3}$	0.01	$9.29e^{-6}$	$1.72e^{-3}$
289	2	4	3	0.03	0.01	$5.72e^{-4}$	$1.81e^{-3}$	0.02	$1.81e^{-5}$	$7.71e^{-4}$
1,089	2	4	4	0.39	0.03	$1.58e^{-4}$	$5.42e^{-4}$	0.06	$9.97e^{-6}$	$2.88e^{-4}$
4,225	2	4	5	5.79	0.12	$4.28e^{-5}$	$2.38e^{-4}$	0.23	$4.84e^{-6}$	$2.06e^{-4}$
16,641	2	4	6	90.14	0.47	$1.14e^{-5}$	$9.66e^{-5}$	0.96	$1.81e^{-6}$	$9.18e^{-5}$
66,049	2	4	7	2,056.82	2.36	$3.16e^{-6}$	$5.93e^{-5}$	4.40	$7.16e^{-7}$	$5.86e^{-5}$
263,169	2	4	8	33,086.24	14.82	$7.96e^{-7}$	$2.85e^{-5}$	20.64	$1.83e^{-7}$	$2.84e^{-5}$
1,050,625	2	4	9	529,380.00[c]	71.38	—	—	103.05	—	—
4,198,401	2	4	10	8,470,080.00[c]	327.24	—	—	510.82	—	—
81	4	6	2	0.00[b]	0.01	$1.99e^{-3}$	$5.74e^{-3}$	0.04	$4.88e^{-6}$	$1.35e^{-4}$
289	4	6	3	0.03	0.02	$5.14e^{-4}$	$1.53e^{-3}$	0.10	$6.30e^{-6}$	$1.01e^{-4}$
1,089	4	6	4	0.39	0.07	$1.38e^{-4}$	$4.16e^{-4}$	0.38	$3.23e^{-6}$	$5.47e^{-5}$
4,225	4	6	5	5.79	0.27	$3.57e^{-5}$	$1.09e^{-4}$	1.48	$8.98e^{-7}$	$2.20e^{-5}$
16,641	4	6	6	90.14	1.10	$9.20e^{-5}$	$2.96e^{-5}$	5.86	$3.39e^{-7}$	$1.14e^{-5}$
66,049	4	6	7	2,056.82	5.45	$2.35e^{-6}$	$9.93e^{-6}$	24.80	$7.14e^{-8}$	$7.27e^{-6}$
263,169	4	6	8	33,086.24	29.95	$5.91e^{-7}$	$3.15e^{-6}$	106.30	$2.95e^{-8}$	$2.62e^{-6}$
1,050,625	4	6	9	529,380.00[c]	151.19	—	—	454.83	—	—
4,198,401	4	6	10	8,470,080.00[c]	685.87	—	—	1905.15	—	—

[a] The running times are in seconds (resolution of 10 ms).
[b] Under 10 ms.
[c] Estimated time of the direct computation as an $O(n^2)$ computation.

TABLE III.
***Multilevel* Performance Table for Dipolar Systems of Varying Number of Dipoles.**

					Without Self-Correction			With Self-Correction		
m	p	s	l	T_d[a]	T_f[a]	E_{err}	∇V_{err}	T_f[a]	E_{err}	∇V_{err}
81	2	4	2	0.01	0.01	$1.57e^{-5}$	$1.69e^{-4}$	0.01	$3.14e^{-7}$	$1.42e^{-4}$
289	2	4	3	0.03	0.02	$3.70e^{-6}$	$5.41e^{-5}$	0.03	$6.24e^{-7}$	$4.94e^{-5}$
1,089	2	4	4	0.42	0.07	$1.09e^{-6}$	$1.49e^{-5}$	0.09	$5.30e^{-8}$	$1.33e^{-5}$
4,225	2	4	5	6.23	0.28	$2.94e^{-7}$	$3.88e^{-6}$	0.34	$8.96e^{-10}$	$3.47e^{-6}$
16,641	2	4	6	104.58	0.89	$7.31e^{-8}$	$9.88e^{-7}$	1.41	$2.06e^{-9}$	$8.94e^{-7}$
66,049	2	4	7	2,059.87	4.55	$1.88e^{-8}$	$2.51e^{-7}$	6.70	$1.34e^{-10}$	$2.27e^{-7}$
263,169	2	4	8	33,252.61	26.37	$4.71e^{-9}$	$6.31e^{-8}$	31.95	$3.67e^{-11}$	$5.69e^{-8}$
1,050,625	2	4	9	532,042.00[b]	125.12	—	—	163.29	—	—
4,198,401	2	4	10	8,512,672.00[b]	518.48	—	—	664.99	—	—
81	4	6	2	0.01	0.01	$2.01e^{-5}$	$1.28e^{-4}$	0.04	$1.51e^{-7}$	$1.71e^{-5}$
289	4	6	3	0.03	0.05	$5.53e^{-6}$	$3.60e^{-5}$	0.15	$1.27e^{-7}$	$6.82e^{-6}$
1,089	4	6	4	0.42	0.16	$1.49e^{-6}$	$9.61e^{-5}$	0.52	$1.27e^{-8}$	$1.87e^{-6}$
4,225	4	6	5	6.23	0.62	$3.84e^{-7}$	$2.49e^{-6}$	2.03	$2.94e^{-9}$	$4.84e^{-7}$
16,641	4	6	6	104.58	2.52	$9.76e^{-8}$	$6.30e^{-7}$	8.23	$7.19e^{-10}$	$1.26e^{-7}$
66,049	4	6	7	2,059.87	12.20	$2.46e^{-8}$	$1.59e^{-7}$	35.12	$1.54e^{-10}$	$3.20e^{-8}$
263,169	4	6	8	33,252.61	62.06	$6.19e^{-9}$	$3.99e^{-8}$	147.00	$3.12e^{-12}$	$8.01e^{-9}$
1,050,625	4	6	9	532,042.00[b]	289.08	—	—	640.47	—	—

[a] The running times are in seconds (resolution of 10 ms).
[b] Estimated time of the direct computation as an $O(m^2)$ computation.

error of the potential's gradient per dipolar particle is defined as $\nabla V_{\text{err}} = \frac{1}{m(n+m)} \sum_{j=1}^{m} |\nabla V_{f_j} - \nabla V_{d_j}|$, where ∇V_{f_j} is the potential gradient computed by the fast algorithm at dipole-j, and ∇V_{d_j} is the direct computation of the potential gradient at that dipole. The energy calculation error per interaction $E_{\text{err}} = |E_f - E_d|/(m+n)^2$, where E_f is the fast computation of the total energy and E_d is the direct computation. In a pure charge system $m = 0$, whereas in a dipolar system $n = 0$. The measurements per interaction give an error estimate of each of the summed terms in eq. (3).

ACCURACY OF THE POTENTIAL, ENERGY, AND FORCES

The errors are measured for two types of runs. The runs applying self-correction (the column specified as "with self-correction"), account for the exclusion of the erroneous self-contribution of the particles aggregates to the potential and energy calculations. Whereas, the column named "without self-correction" presents the results for the cases where this type of error is not handled.

For each particle, as its number of interactions grows linearly with the linear increase in the total number of particles, the per-interaction errors decrease linearly by the same factor.

This is explained by the fact that far away particles have less influence than nearby local ones. Therefore, when increasing the number of interactions, the "constant" local-induced error (per particle) is averaged out. The local-induced errors introduced by the smoothness properties of the kernel softeners, and by the self-contribution, are yielded for every particle in the system. As seen by the results obtained for E_{err}, self-correction may be important for systems where high accuracy in the energy calculation is desired. In all systems and cases therein, E_{err} obtained is smaller than the relevant V_{err} and ∇V_{err}. The energy computation is based on the summation of the potentials obtained and the particles charges. The potential computation can be above or below its direct computed value, and hence, as a result the energy calculation error is averaged out. The measured errors for dipolar systems compared with charged systems show that the former errors are smaller than those of the latter. This is because the dipolar potential gradient kernel has a smaller long-range contribution than that of the charge potential kernel. We can thus approximate that the measured errors for the computation of forces will be in the order of those for the dipolar

TABLE IV.
***Multilevel* Performance Table for Heterogeneous (Charge and Dipolar) Particle Systems of Varying Number of Particles.**

					Without Self-Correction				With Self-Correction			
$m+n$[a]	p	s	l	T_d[b]	T_f[b]	E_{err}	V_{err}	∇V_{err}	T_f[b]	E_{err}	V_{err}	∇V_{err}
162	2	4	2	0.01	0.01	$4.68e^{-4}$	$3.02e^{-4}$	$4.53e^{-4}$	0.02	$4.93e^{-6}$	$9.10e^{-4}$	$4.55e^{-4}$
578	2	4	3	0.12	0.04	$1.25e^{-4}$	$8.95e^{-4}$	$1.42e^{-4}$	0.06	$1.51e^{-6}$	$3.76e^{-4}$	$1.41e^{-4}$
2,178	2	4	4	1.70	0.11	$4.11e^{-5}$	$3.25e^{-4}$	$4.32e^{-5}$	0.18	$3.93e^{-6}$	$2.28e^{-4}$	$4.31e^{-5}$
8,450	2	4	5	25.03	0.42	$1.12e^{-5}$	$1.47e^{-4}$	$1.43e^{-5}$	0.72	$1.88e^{-8}$	$1.34e^{-4}$	$1.43e^{-5}$
33,282	2	4	6	447.14	1.76	$2.85e^{-6}$	$5.24e^{-5}$	$3.75e^{-6}$	2.58	$4.47e^{-7}$	$5.02e^{-5}$	$3.74e^{-6}$
132,098	2	4	7	7,428.57	11.50	$7.67e^{-7}$	$2.24e^{-5}$	$1.05e^{-6}$	15.64	$1.58e^{-7}$	$2.19e^{-5}$	$1.05e^{-6}$
526,338	2	4	8	120,715.16	60.78	$2.03e^{-7}$	$1.43e^{-5}$	$2.93e^{-7}$	73.67	$5.11e^{-8}$	$1.42e^{-5}$	$2.92e^{-7}$
2,101,250	2	4	9	1,931,443.00[c]	281.31	—	—	—	347.17	—	—	—
8,396,802	2	4	10	30,903,088.00[c]	1290.86	—	—	—	1628.33	—	—	—
162	4	6	2	0.01	0.02	$4.25e^{-4}$	$2.61e^{-3}$	$9.17e^{-5}$	0.07	$1.65e^{-7}$	$8.70e^{-5}$	$6.23e^{-5}$
578	4	6	3	0.12	0.08	$1.23e^{-4}$	$7.62e^{-4}$	$2.99e^{-5}$	0.26	$1.14e^{-6}$	$4.92e^{-5}$	$2.52e^{-5}$
2,178	4	6	4	1.70	0.27	$3.44e^{-5}$	$2.10e^{-4}$	$8.23e^{-6}$	0.95	$5.33e^{-7}$	$2.48e^{-5}$	$6.95e^{-6}$
8,450	4	6	5	25.03	1.07	$8.79e^{-6}$	$5.48e^{-5}$	$2.48e^{-6}$	3.69	$2.77e^{-7}$	$1.34e^{-5}$	$2.20e^{-6}$
33,282	4	6	6	447.14	4.46	$2.27e^{-6}$	$1.47e^{-5}$	$6.53e^{-7}$	14.89	$7.67e^{-8}$	$5.81e^{-5}$	$5.93e^{-7}$
132,098	4	6	7	7,428.57	25.96	$5.83e^{-7}$	$4.42e^{-6}$	$1.83e^{-7}$	67.87	$2.73e^{-8}$	$2.83e^{-6}$	$1.68e^{-7}$
526,338	4	6	8	120,715.16	129.17	$1.47e^{-7}$	$1.83e^{-6}$	$4.98e^{-8}$	289.12	$8.36e^{-9}$	$1.61e^{-6}$	$4.66e^{-8}$
2,101,250	4	6	9	1,931,443.00[c]	605.34	—	—	—	1264.79	—	—	—

[a] Half the particles are dipoles and the other half are point-charges.
[b] The running times are in seconds (resolution of 10 ms).
[c] Estimated time of the direct computation as an $O((m+n)^2)$ computation.

systems, because the forces are the derivatives of the potential kernels. In addition, for dipolar systems and force calculations smaller values for ps and ss can be employed in comparison to those needed to obtain accurate potentials of a charge system.

We have tested the algorithm for studying the effect the dipoles magnitude and orientation, and charge characteristics, i.e., positive, negative or combined. Confined systems of homogeneous charges (positive or negative), or systems of restricted dipole orientation, yield slightly larger errors compared to the aforementioned unrestricted systems (results not shown here). This is because the errors are a function of the charges values (coupled with the interpolation/anterpolation errors). In a restricted positive (or negative) charge system, the fast computed potential values are consistently above (or below) the direct computed values. Therefore, relative error measurements can also be employed (e.g., $V_{\mathrm{err}}^{rlt} = \frac{1}{n}\sum_{i=1}^{n} |(V_{f_i} - V_{d_i})/V_{d_i}|$), because the variation of the difference between the fast and direct evaluation is small, yielding reliable "normalized" error values. In orientation-restricted dipolar systems the error estimates do not manifest themselves as in positively or negatively charged systems. A dipole, by definition is composed of a positive and a negative charge, and as such, the long-range dipoles charges tend to average out as previously has been explained for positive and negative charged systems.

THE EFFECT OF p AND s

In Tables II–IV, we also show the effect of the softening distance s and the interpolation order p on the measured errors and running times. For all three systems, all error measures improve when increasing s or p separately, or both (this was also successfully tested in the two-level system—not shown here). However, this influences the computational work invested as the complexity of the algorithm is $O((m+n)p^2)$. Because we want to maintain this complexity, the number of corrections carried out for every particle should be on average p^2. Therefore, $(s/a)^2$, the number of corrections, should be in the order of p^2. Taking into consideration p, s, and h, the error obtained from interpolating the softened potential kernel is of the order $O((\gamma h)^p(p!/s^p)) \sim O(((\gamma hp)/(es))^p)$, where $(p!/s^p)$ is an approximation of the p-order derivative of the kernel G_{smooth}; $p! \sim (p/e)^p$ by Sterling, where e is the base of the natural logarithm. Thus, we should choose $s = O(hp)$, that is s/h and p should be kept in the same order.

Next, we will consider the amount of work performed as a function of p, s, and h. The number of operations performed per gridpoint on the grid of size h for anterpolating it to grid $2h$, interpolating to it from grid $2h$, and the corrections related to it is $2p + 4(2s/h)$ (assuming the softening distance is also doubled). The number of operations performed per arbitrarily positioned particle point for anterpolating it to grid h, interpolating to it from grid h, and its required corrections (not including self correction) is $2p^2 + 4(s/a)^2$ (plus $2p^2$ if self-corrections are properly introduced). In the cases presented here, the execution times double when increasing p from 2 to 4 and s, from 4 to 6 (or multiplied by four times when the self-correction is accounted for, which is of order $O(p^4)$ per particle in this version of implementation). Nevertheless, the running times of the varying sizes of the systems still increase linearly, as a function of the system size.

THE EFFECT OF h

We have previously mentioned that the use of kernel softeners and the anterpolation/interpolation processes yields a bounded $O(h^p G_{\mathrm{smooth}}^{(p)})$ approximation error for the potential, where $G_{\mathrm{smooth}}^{(p)}$ is the maximum p-order derivative of the kernel softener G_{smooth}. The gradient of the potential approximation (the ∇V computation) has a bounded $O(h^{p-1} G_{\mathrm{smooth}}^{(p-1)})$ error. This is seen by the results obtained from runs of different values of h in a heterogeneous particle system (not shown here). For a fixed s, as the mesh size h is reduced by two-fold, V_{err} decreases by $O(2^p)$, whereas ∇V_{err} decreases $O(2^{p-1})$.

The anterpolation/interpolation errors rise as the grids get coarser and coarser, because the mesh size h is enlarged. Thus, to reduce the anterpolation/interpolation errors, the interpolation-order p and the softening distance s are chosen for every grid level currently processed. That is, for coarser levels, a higher interpolation order p and/or larger softening distance s are the remedy for reducing the interpolation errors obtained. The computational cost does not rise by much, as the larger p and s are chosen to be exercised on coarser grids.

THE EFFECT OF THE PARTICLES DISTRIBUTION

In real-life particle systems, the positions of the particles are not chosen randomly as has been done here. In the natural systems, the particles are dis-

tributed more uniformly and cannot come as near to each other as may randomly occur in our test systems. We have obtained bounded and small computations errors in our unrestricted "worst-case" test systems. Indeed, the natural relaxation on the particle location results in even better accuracy. This was checked by running the algorithm on additional test cases where the distance between the particles has been restricted to pairs of points that are $>a$ apart from each other (e.g., >2). The results obtained have shown that the computation errors even decreased by an order of magnitude in comparison to the unrestricted cases. In these latter restricted cases, each particle has less number of interactions with very close neighbors within its softening region. Hence, most of the computation is done with the exact potential, as the particles interactions are usually beyond the singularity region of the kernel.

The system was also tested for its performance on a line-like distribution of particles (e.g., a circle) to emulate anisotropic particle distribution. Both the running times and accuracy are comparable to the the cases where the particles are arbitrarily distributed. As an example, for a charge system composed of 263,169 particles distributed on a circle, a fine grid of 1025×1025 gridpoints and a multilevel run have yielded $V_{\text{err}} = 4.16e^{-5}$ and $E_{\text{err}} = 2.6e^{-7}$ (the radius of the circle is $1025 \times h$, i.e., around 85 particles on average reside in the $h \times h$ bin). The running time obtained is 109.11 s compared with the direct computation of 32,390 s (almost 9 h). When using a 513×513 grid (the radius is $513 \times h$, i.e., around 165 particles on average reside in a bin), we have obtained $V_{\text{err}} = 4.72e^{-5}$, $E_{\text{err}} = 2.18e^{-7}$, and $T_f = 194$ s. Both the number of particles and the number of gridpoints on the finest grid level can be of the same order for obtaining accurate results and fast running times. For the particular handling of cases of variable particle density an extension of the algorithm can be found in ref. 5.

OPTIMIZATION ISSUES

Here we have described our algorithm and its options, and have demonstrated its capabilities. Optimal values can be obtained for the algorithm's parameters, i.e., the softening distance s, the interpolation order p, the mesh size h , and the number of grid levels. The computational work can be minimized subject to given constraints on the error measurements. Or, for a given work requirement, the errors can be minimized. Another possibility is to minimize the work and errors given the exchange rate of accuracy for work. All in all, the parameters are determined according to the application in which our method is employed. See ref. 13 for an example of a parameter optimization and control procedure.

Discussion

In this work we have demonstrated a multiscale method for the fast summation of long-range interactions, namely, the rapid evaluation of the potential and energy induced in homogeneous and heterogeneous point-charge and dipolar systems. We have achieved this using linear complexity. This means that although we are summing $O(N^2)$ interactions, the execution time grows only linearly with the number N of particles.

Here, the method is applied to the fast summation of the Coulombic potential of charges and dipoles in 2D. Our algorithm can be extended to any asymptotically smooth kernels, not only for potential-type kernels, and to any space dimension. Nevertheless, because our multiscale approach is generic, i.e., it is not restricted to this kind of functions and can be straightforwardly modified to handle other asymptotically smooth kernels (oscillatory kernel handling appears in ref. 5). As an example, we can consider the long-range dipolar system used to describe magnetic phenomenon such as complex ferrofluids and ultrathin magnetic films.[14, 15] The energy function of a system of magnetic dipoles on a square lattice represents also long-range dipolar interaction and is over all pairs of moments on the lattice. The long-range dipole–dipole interaction kernel can be softened using the compatible derivatives of the softening polynomials, and the fast summation can be carried out using our multiscale structure and mechanisms, i.e., the fine-to-coarse and coarse-to-fine procedures. An additional example where our algorithm can be utilized is the gravitational N-body problem is astrophysics. There, N mass points (of stars/galaxies) are moving under their mutual gravitational forces according to Newton's laws of motions.

Our method relies on decomposing the electrostatic potential into a smooth (or "softened") part and a local part, and describing the smooth part as the potential of aggregated charges and dipoles defined on a coarser grid. The long-range dipolar kernel is softened using the derivatives of the charge-kernel softening polynomials. The main advantage, and indeed a motivation for this method, is that it provides the structure for describing collective particle *motions* on larger scales that facilitates

the acceleration of Monte-Carlo (MC) simulations and energy minimization processes. For example, the direct fine-scale MC simulation of water and other fluids tend to be extremely inefficient due to the very slow change of various kind of clusters, at various scales, for example, clusters of aligned dipoles, whose size depend on the given temperature. The general multiscale approach to accelerate MC simulations and inexpensively average over many large-scale fluctuations (see refs. 6, 16, 17, and 18), is based on devising (Lagrangian or Eulerian) dynamics of aggregated quantities at increasingly larger scales. When electrostatic interactions are involved, their smooth part can directly be transferred to the larger scale dynamics by being described, as in the present work, through the fields of aggregated charges and dipoles. The local part, together with other local interaction, is separately transferred to the coarse levels by a different approach (iterative construction of local coarse-level potential by comparing local MC simulations at the coarse level with such simulations at the fine level).

Acknowledgments

I thank Prof. A. Brandt for advising and commenting the manuscript, and Drs. C. H. Venner, V. Ilyin, and V. Rozenbaum for helpful discussions, encouragement and interest.

References

1. Sandak, B.; Nussinov, R.; Wolfson, H. J Comput Biol 1998, 5, 631.
2. Sandak, B.; Wolfson, H. J.; Nussinov, R. Proteins Struct Funct Genet 1998, 32, 159.
3. Brandt, A. In Multigrid Methods; Hackbusch, W.; Trottenberg, V., Eds.; Springer Veralg, 1982, see §8.6.
4. Brandt, A.; Lubrecht, A. A. J Comput Phys 1990, 90, 348.
5. Brandt, A. Comput Phys Commun 1991, 65, 24.
6. Brandt, A. In Report WI/GC-12, Weizmann Institute of Science, 1999.
7. Hockney, R.; Eastwood, J. Computer Simulation Using Particles; McGraw-Hill: New York, 1981.
8. Appel, A. SIAM J Sci Stat Comput 1985, 6, 85.
9. Barnes, J.; Hut, P. Nature 1986, 324, 446.
10. Greengard, L. Science 1994, 265, 909.
11. Greengard, L.; Rokhlin, V. J Comput Phys 1987, 73, 325.
12. Kutteh, R.; Nicholas, J. Comput Phys Commun 1995, 86, 236.
13. Brandt, A.; Venner, C. SIAM J Sci Comput 1998, 19, 468.
14. Bruno, A.; Pisacane, F.; Rosato, V. Int J Mod Phys 1997, 8, 459.
15. MacIsaac, A.; De'Bell, K.; Whitehead, J. Phys Rev Lett 1998, 80, 616.
16. Brandt, A.; Ron, D. J Stat Phys 2001, 102, 231.
17. Shumulyian, S. PhD thesis, Weizmann Institute of Science, 1999.
18. Brandt, A.; Galun, M.; Ron, D. J Stat Phys 1994, 74, 313.