Mesh-free hierarchical clustering methods for fast evaluation of electrostatic interactions of point multipoles

H. A. Boateng

Citation: The Journal of Chemical Physics **147**, 164104 (2017); doi: 10.1063/1.4990552
View online: http://dx.doi.org/10.1063/1.4990552
View Table of Contents: http://aip.scitation.org/toc/jcp/147/16
Published by the American Institute of Physics

Mesh-free hierarchical clustering methods for fast evaluation of electrostatic interactions of point multipoles

H. A. Boateng[a)]
Department of Mathematics, Bates College, 2 Andrews Rd., Lewiston, Maine 04240, USA

(Received 15 June 2017; accepted 10 October 2017; published online 23 October 2017)

Electrostatic interactions involving point multipoles are being increasingly implemented to achieve higher accuracy in molecular simulations. A major drawback of multipolar electrostatics is the increased computational cost. Here we develop and compare two Cartesian tree algorithms which employ Taylor approximations and hierarchical clustering to speed up the evaluation of point multipole interactions. We present results from applying the algorithms to compute the free space Coulomb potential and forces of different sets of interacting point multipoles with different densities. The methods achieve high accuracy and speedup of more than an order of magnitude over direct sum calculations and scale well in parallel. *Published by AIP Publishing.* https://doi.org/10.1063/1.4990552

I. INTRODUCTION

It is self-evident that point charge electrostatic models have been immensely useful in the study of the dynamical and structural properties of condensed phase systems. However, the inability of the standard point charge models to capture the anisotropy inherent in electrostatic interactions limits the models' range of application.[1–8] Point multipolar electrostatic models have been shown to provide better accuracy than standard fixed point charge models[9–16] across a broad range of simulations. However, at present, the computational cost for simulating a system of point multipoles is significantly higher than for fixed point charge models and that has naturally limited the adoption of point-multipole models. As an example, a particle-mesh Ewald (PME) method for multipolar electrostatics up to hexadecapole-hexadecapole interactions, developed by Sagui and co-workers,[17] was found to be 8.5 times slower than the fixed point charge model. Prior to the work of Sagui and co-workers, theoretical development of Ewald sum based multipolar electrostatics had been limited to interactions with multipolar order up to one[18,19] and two[20,21] primarily because of the cumbersome nature of the Cartesian representation and the lack of availability of computing power. Sagui *et al.*[17] and Simmonett *et al.*[22] developed methods for computing the real space multipolar Ewald interactions in Cartesian and spherical coordinates, respectively. Boateng and Todorov[23] generalized the work of Sagui *et al.*[17] to other interaction potentials and to both the real and reciprocal space sums of PME. Lin[24] has provided a more efficient recurrence method for computing multipolar interactions due to pair potentials.

Several fast approximate methods have been developed over the years for fixed-point charge interactions in classical molecular simulations. These include smooth cutoff methods[25] and mesh-based methods such as particle mesh Ewald (PME)[26] and the particle-particle particle mesh (P3M) methods,[27] which interpolate particle positions and charges on a grid and employ fast Fourier transforms (FFTs) to evaluate the potential on the grid. An alternative mesh-based method that usually avoids the use of FFTs is the multilevel summation method[28,29] which interpolates particle positions and charges on a set of hierarchical nested meshes and uses a multigrid method[30–32] to evaluate the potential on the nested meshes. In addition to the mesh-based methods, a different class of methods based on a hierarchical tree clustering of particles has been developed for fixed-point charge interactions. These include the fast multipole method,[33] the cell multipole method,[34] and the Cartesian treecode methods.[35–38]

Despite the tremendous amount of method development for fast evaluation of fixed-point charge interactions in classical molecular simulations, there is a significant lack of similar developments for classical multipolar electrostatic interactions. Sagui, Pedersen, and Darden (SPD)[17] developed a particle mesh Ewald method for multipolar electrostatics up to hexadecapole-hexadecapole interactions. Recently a fast multipole method[39] for induced dipole-dipole interactions has also been developed. This work develops Cartesian treecode methods to provide fast and accurate approximations for classical permanent multipolar electrostatic interactions. The methods are specifically suited for simulations with free-space boundary conditions. Treecode methods are in spirit similar to fast multipole methods.[33,39–41] As such, analogous fast multipole methods can be developed for permanent Cartesian multipolar electrostatic interactions. For practitioners simulating systems with periodic boundary conditions, the particle-mesh Ewald method[17–23,42] is more suitable. Future work will focus on extending the methods developed in this paper to periodic boundary conditions to be compared to particle-mesh Ewald.

The rest of this paper is organized as follows: Sec. II introduces the formalism for multipolar interactions needed for our derivations in Sec. III. In Sec. III we explain the

[a)]Electronic mail: hboateng@bates.edu

procedure for constructing the hierarchical tree and introduce the particle-cluster and cluster-particle algorithms completely with pseudocode. In addition, we derive the expressions needed for approximating several electrostatic quantities using either particle-cluster or cluster-particle and provide other details for the implementation of the two algorithms. Section IV provides numerical results comparing the efficiency of the two algorithms based on the error in approximating point potentials and the CPU time for different user-specified parameters for the two methods. We end the paper with the summary and concluding remarks.

II. FORMALISM FOR MULTIPOLAR INTERACTIONS

We define a multipolar operator, $\hat{L}_i$, as

$$\hat{L}_i = \Big(q_i + \mathbf{p}_i \cdot \nabla_i + \mathbf{Q}_i : \nabla_i\nabla_i + \mathbf{O}_i\vdots\nabla_i\nabla_i\nabla_i + \mathbf{H}_i :: \nabla_i\nabla_i\nabla_i\nabla_i + \dots\Big), \tag{1}$$

where q_i, $\mathbf{p}_i$, $\mathbf{Q}_i$, $\mathbf{O}_i$, and $\mathbf{H}_i$ are the point charge, dipole, quadrupole, octupole, and hexadecapole tensors, respectively, of atom i, ∇_i refers to the three-dimensional gradient with respect to the position of atom i, and the "dot" products stand for tensor contraction.

Additionally, we define a unidimensional vector of independent multipole moments,

$$\mathcal{M} = \Big(q, p_x, p_y, p_z, Q_{xx}, Q_{xy}, Q_{xz}, Q_{yy}, Q_{yz}, Q_{zz}, O_{xxx}, \dots, H_{xxxx}, H_{xxxy}, \dots\Big), \tag{2}$$

based on the original multipole vector $\mathcal{M}'$ which has degenerate components for multipoles of order two or higher. $\mathcal{M}$ is indexed by a triplet. As an example, $\mathcal{M}^{000}$ refers to the zero-order multipole (monopole), i.e., $\mathcal{M}^{000} = q$, $\mathcal{M}^{100}$ refers to the x-component of the dipole, i.e., p_x, and $\mathcal{M}^{211}$ refers to the hexadecapole component $\mathbf{H}_{xxyz}$. In addition, individual components of $\mathcal{M}$ contain the sum of all original multipole components, $\mathcal{M}'$, related by symmetry. For instance, the octupole $\mathcal{M}^{111}$ is the sum of the corresponding six degenerate components in the original multipole vector, $\mathcal{M}'$. Thus $\mathcal{M}^{111} = O'_{xyz} + O'_{xzy} + O'_{yxz} + O'_{yzx} + O'_{zxy} + O'_{zyx} = 6O'_{xyz}$.

By defining a unidimensional vector of independent (non-degenerate) multipole moments, $\mathcal{M}_i$, for atom i, the corresponding multipolar operator to an arbitrary order p can be written in a more compact form as

$$\hat{L}_i = \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_i^{\mathbf{s}} \partial_i^{\mathbf{s}}. \tag{3}$$

Here $\mathbf{s} = (s_1, s_2, s_3)$ is the triplet that runs over all independent multipoles, $||\mathbf{s}|| = s_1 + s_2 + s_3$, $\mathcal{M}_i^{\mathbf{s}} = \mathcal{M}_i^{s_1 s_2 s_3}$, and $\partial_i^{\mathbf{s}} = \partial_{z_i}^{s_3} \partial_{y_i}^{s_2} \partial_{x_i}^{s_1}$ is the multidimensional derivative with respect to the position $\langle x_i, y_i, z_i \rangle$ of atom i with orders s_1, s_2, and s_3 in the x, y, and z directions, respectively. For pair potentials, it is often convenient to redefine the multipolar operator for atom j in terms of the derivatives with respect to the position of atom i to arrive at

$$\begin{aligned}\hat{L}_j &= \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_j^{\mathbf{s}} \partial_j^{\mathbf{s}} \\ &= \sum_{||\mathbf{s}||=0}^{p} (-1)^{||\mathbf{s}||} \mathcal{M}_j^{\mathbf{s}} \partial_i^{\mathbf{s}} \\ &= \sum_{s_3=0}^{p} \sum_{s_2=0}^{p-s_3} \sum_{s_1=0}^{p-s_3-s_2} (-1)^{s_1+s_2+s_3} \mathcal{M}_j^{s_1 s_2 s_3} \partial_{z_i}^{s_3} \partial_{y_i}^{s_2} \partial_{x_i}^{s_1}.\end{aligned} \tag{4}$$

III. TREE ALGORITHMS

A. Introduction

The treecode algorithm restructures a set of interacting particles into a hierarchical octree. In the simplest case,[35] an octree is formed by first considering the root of the tree as the smallest rectangular box that encloses the particles. The root is then divided into eight clusters, which forms the next level of the tree. Then each of these eight clusters is divided into eight. This process continues recursively on each branch until the number of particles in a cluster is less than or equal to a predetermined number, N_0. The last cluster at the end is called a leaf. This procedure generates a hierarchical tree of $\log N$ levels where N is the number of particles in the tree. An example clustering in 2-D is shown in Fig. 1. In the figure, the clustering stops when the number of particles in a leaf is less than or equal to 3. The root (level 0) has 12 particles.

The hierarchical tree is used to separate particle-particle interactions into near-field and far-field interactions by replacing particle-particle interactions with ones involving particles and clusters. Interactions between particles and clusters that are deemed to be near-field are computed directly, while those that are deemed to be far-field are approximated by a Taylor approximation about the center of the cluster. For a set of N interacting particles, the treecode reduces the $O(N^2)$ cost to $O(N\log N)$.[35,38,43,44] Cartesian treecode methods have been developed for varied kernels including the charge-charge real space Ewald sum,[36] power law potentials,[37] multi-quadric radial basis functions,[45,46] Matérn kernel,[47] and regularized Biot-Savart kernel.[44] Boateng and Krasny[38] studied two different formulations of the Cartesian treecode which they called *particle-cluster* and *cluster-particle* and identified the settings for which each is preferred in a charge-charge system with disjoint target and source particles. In the present work, we develop *particle-cluster* and *cluster-particle* treecode algorithms to speed up evaluations of multipolar electrostatic interactions. The target and sources are the same. The tree construction procedure outlined above is the same for both methods developed in this paper. The *particle-cluster* algorithm views the tree as a cluster of source particles, while the

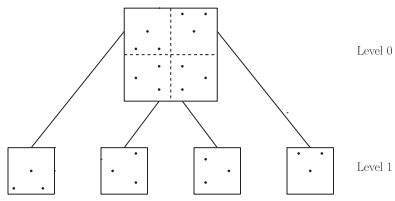

FIG. 1. A depiction of hierarchical tree clustering in 2-D. Here the root (level 0) is divided into four clusters which form level 1.

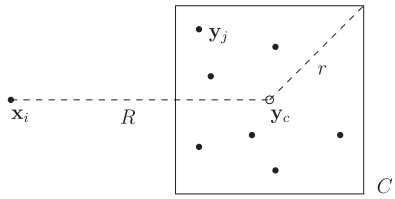

FIG. 2. A particle-cluster interaction between a particle at $\mathbf{x}_i$ and particles at $\mathbf{y}_j$ in cluster C. The cluster has center $\mathbf{y}_c$ and radius r, and the particle-cluster distance is R.

cluster-particle algorithm views the tree as a cluster of target particles.

B. Particle-cluster treecode

In a system of interacting particles, let $\mathbf{x}_i$ be the position of particle i with multipolar vector $\mathcal{M}_i$ and multipolar operator $\hat{L}_i$ interacting via a pair potential with a cluster, C, of M particles with multipolar vectors and multipolar operators $\mathcal{M}_j$ and $\hat{L}_j$, respectively, at position $\mathbf{y}_j$, $\{\mathbf{y}_j, \mathcal{M}_j, \hat{L}_j, j = 1 : M\}$, shown schematically in Fig. 2. Cluster C has radius r, center $\mathbf{y}_c$, and the distance from the $\mathbf{x}_i$ to $\mathbf{y}_c$ is R. Let ϕ be the pair potential kernel, then for a pth order multipole, the potential at $\mathbf{x}_i$ due to the particles in cluster C is exactly given as

$$V_{i,C} = \sum_{j=1}^{M} \hat{L}_j \phi(\mathbf{x}_i, \mathbf{y}_j) = \sum_{\mathbf{y}_j \in C} \hat{L}_j \phi(\mathbf{x}_i, \mathbf{y}_j) = \sum_{\mathbf{y}_j \in C} \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_j^{\mathbf{s}} \partial_j^{\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_j), \quad (5)$$

and the potential energy due to the interaction between particle i and cluster C is exactly evaluated as

$$U_{i,C} = \sum_{j=1}^{M} \hat{L}_i \hat{L}_j \phi(\mathbf{x}_i, \mathbf{y}_j) = \sum_{\mathbf{y}_j \in C} \sum_{||\mathbf{n}||=0}^{p} \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_i^{\mathbf{n}} \mathcal{M}_j^{\mathbf{s}} \partial_i^{\mathbf{n}} \partial_j^{\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_j). \quad (6)$$

Particle i and cluster C are well separated if $\frac{r}{R} \leq \theta$,[35,43] where $0 < \theta < 1$. This is called the multipole acceptance criterion (MAC).[35,43] We now derive the formulae for the Taylor approximations of (1) the potential at position $\mathbf{x}_i$ due to a cluster of particles, C, (2) the potential energy due to the interaction between particle i and cluster C, and (3) the force on particle i due to cluster C. Formulae for the electric field and torque are derived in Subsection 1 of the Appendix.

1. Approximation of the potential

If the particle and cluster are well separated, then the exact potential given in Eq. (5) is approximated by an ℓth order Taylor expansion about $\mathbf{y} = \mathbf{y}_c$,

$$\sum_{\mathbf{y}_j \in C} \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_j^{\mathbf{s}} \partial_j^{\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_j) \approx \sum_{\mathbf{y}_j \in C} \sum_{||\mathbf{k}||=0}^{\ell} \frac{1}{\mathbf{k}!} \partial_{\mathbf{y}}^{\mathbf{k}} \left(\sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_j^{\mathbf{s}} \partial_j^{\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_c) \right) (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} \quad (7a)$$

$$= \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{s}||=0}^{p} \frac{1}{\mathbf{k}!} \partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_c) \sum_{\mathbf{y}_j \in C} \mathcal{M}_j^{\mathbf{s}} (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} \quad (7b)$$

$$= \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{s}||=0}^{p} b_{\mathbf{k}}^{\mathbf{s}}(\mathbf{x}_i, \mathbf{y}_c) H_{\mathbf{k}}^{\mathbf{s}}(C). \quad (7c)$$

In Eqs. (7a) and (7b), $||\mathbf{k}|| = k_1 + k_2 + k_3$, $\mathbf{k}! = k_1! k_2! k_3!$, $\partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}} = \partial_{y_1}^{k_1+s_1} \partial_{y_2}^{k_2+s_2} \partial_{y_3}^{k_3+s_3}$, and $(\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} = (y_{j_1} - y_{c_1})^{k_1} (y_{j_2} - y_{c_2})^{k_2} (y_{j_3} - y_{c_3})^{k_3}$. In Eq. (7c), we define the $\mathbf{k}$th Taylor coefficient of the $\mathbf{s}$th order multipole kernel,

$$b_{\mathbf{k}}^{\mathbf{s}}(\mathbf{x}_i, \mathbf{y}_c) = \frac{1}{\mathbf{k}!} \partial_{\mathbf{y}}^{\mathbf{k}} \partial_{\mathbf{y}}^{\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_c) = \frac{1}{\mathbf{k}!} \partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_c), \quad (8)$$

and an $\mathbf{s}$th order cluster moment,

$$H_{\mathbf{k}}^{\mathbf{s}}(C) = \sum_{\mathbf{y}_j \in C} \mathcal{M}_j^{\mathbf{s}} (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}}, \quad (9)$$

to give the ℓth order Taylor approximation for the potential due to the particle-cluster interaction. The cluster moments depend exclusively on the particles in the cluster and as such can be stored and reused without having to recompute them.

2. Approximation of the energy

By using the definitions in Eqs. (3), (5), and (7b) and observing that for a pair potential kernel $\phi(\mathbf{x}_i, \mathbf{y}_j)$, $\partial_i^{\mathbf{n}} \phi = (-1)^{||\mathbf{n}||} \partial_j^{\mathbf{n}} \phi = \partial_{\mathbf{y}}^{\mathbf{n}} \phi$, we can approximate the energy due to a well-separated particle-cluster interaction to ℓth order as

$$\sum_{j=1}^{M} \hat{L}_i \hat{L}_j \phi(\mathbf{x}_i, \mathbf{y}_j) \approx \sum_{||\mathbf{n}||=0}^{p} \mathcal{M}_i^{\mathbf{n}} \partial_i^{\mathbf{n}} \left(\sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{s}||=0}^{p} \frac{1}{\mathbf{k}!} \partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_c) \sum_{\mathbf{y}_j \in C} \mathcal{M}_j^{\mathbf{s}} (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} \right)$$

$$= \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{n}||=0}^{p} (-1)^{||\mathbf{n}||} \mathcal{M}_i^{\mathbf{n}} \sum_{||\mathbf{s}||=0}^{p} \frac{1}{\mathbf{k}!} \partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}+\mathbf{n}} \phi(\mathbf{x}_i, \mathbf{y}_c) \sum_{\mathbf{y}_j \in C} \mathcal{M}_j^{\mathbf{s}} (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}}$$

$$= \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{n}||=0}^{p} (-1)^{||\mathbf{n}||} \mathcal{M}_i^{\mathbf{n}} \sum_{||\mathbf{s}||=0}^{p} b_{\mathbf{k}}^{\mathbf{s}+\mathbf{n}}(\mathbf{x}_i, \mathbf{y}_c) H_{\mathbf{k}}^{\mathbf{s}}(C). \quad (10)$$

3. Approximation of the force

The force on i due to the particles in cluster C, $\mathbf{f}_{i,C} = -\nabla_{\mathbf{x}_i} U_{i,C} = \nabla_{\mathbf{y}} U_{i,C}$, is approximated as

$$\mathbf{f}_{i,C} \approx \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{n}||=0}^{p} (-1)^{||\mathbf{n}||} \mathcal{M}_i^{\mathbf{n}} \sum_{||\mathbf{s}||=0}^{p} \nabla_{\mathbf{y}} \left(b_{\mathbf{k}}^{\mathbf{s}+\mathbf{n}} \right) H_{\mathbf{k}}^{\mathbf{s}}(C) \tag{11}$$

$$= \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{n}||=0}^{p} (-1)^{||\mathbf{n}||} \mathcal{M}_i^{\mathbf{n}} \sum_{||\mathbf{s}||=0}^{p} \begin{bmatrix} b_{\mathbf{k}}^{\mathbf{s}+\mathbf{n}+e_1} \\ b_{\mathbf{k}}^{\mathbf{s}+\mathbf{n}+e_2} \\ b_{\mathbf{k}}^{\mathbf{s}+\mathbf{n}+e_3} \end{bmatrix} H_{\mathbf{k}}^{\mathbf{s}}(C), \tag{12}$$

where $e_1 = \langle 1,0,0 \rangle$, $e_2 = \langle 0,1,0 \rangle$, and $e_3 = \langle 0,0,1 \rangle$.

4. The particle-cluster algorithm

The pseudo-code for particle-cluster[35,38] is given in Algorithm 1. The algorithm requires the user to specify the order of Taylor approximation, ℓ, the MAC, θ, and the maximum number of particles in a leaf, N_0. After the tree is built, the algorithm proceeds with a loop over all particles, $\mathbf{x}_i$, where each particle interacts with the root of the tree via a call to the subroutine **compute-pc** in **line 5** of the algorithm. Each particle-root interaction fails and the algorithm descends to the next level of the tree, i.e., the children of the current cluster. Here the MAC is tested again for each potential interaction between the particle and the clusters on this level of the tree. If the MAC is satisfied, the algorithm performs two tasks as follows:

1. It computes and stores the moments, Eq. (9), of the clusters, if the moments are not already available, to be used for interactions with other particles.
2. The interaction between the particle and the cluster is approximated by the Taylor series polynomial in Eq. (7c).

If the MAC is again not satisfied, the algorithm descends to the next level. This procedure will continue until the algorithm reaches a leaf at which point the interaction between particle i and the leaf is evaluated by direct summation if the MAC is not satisfied.

Algorithm 1. Particle-cluster treecode.

```
1  program pc-treecode
2    input: N particles x_i, multipoles M_i, parameters p, θ, N_0
3    output: potential, electric field, energies, forces, torques
4    Set {y_j}_{j=1}^N = {x_i}_{j=1}^N; construct tree of source particles y_j
5    for i = 1: N; compute-pc(x_i, root); end
6  end program
7  subroutine compute-pc (x, C)
8    if MAC is satisfied
9      compute and store moments of C (if not already available)
10     compute particle-cluster interaction by Taylor approximation
11   else if C is a leaf
12     compute particle-cluster interaction by direct summation
13   else for each child of C
14     compute-pc(x, child)
15 end subroutine
```

The treecode achieves speedup in two ways: (**1**) The exact particle-cluster interactions are replaced by a Taylor approximation that typically involves fewer arithmetic. (**2**) Once the moments of a cluster is computed, it is stored and used in interactions of the cluster with every other well separated particle without recomputing.

C. Cluster-particle treecode

While the particle-cluster algorithm looks at the effect of a cluster of source particles on a target particle, the cluster-particle algorithm focuses on the effect of a source particle on a cluster of target particles. Consider a cluster, C, of particles at position $\mathbf{x}_i$, $\{\mathbf{x}_i, \mathcal{M}_i, \hat{L}_i\}$, interacting with M particles outside the cluster at position $\mathbf{y}_j$ with multipolar vectors and operators $\mathcal{M}_j$ and $\hat{L}_j$, $\{\mathbf{y}_j, \mathcal{M}_j, \hat{L}_j, j = 1 : M\}$, respectively, as depicted in Fig. 3.

The cluster has center $\mathbf{x}_c$, radius r, and the distance from the center to $\mathbf{y}_j$ is R. The cluster-particle interaction is well separated if $\frac{r}{R} \leq \theta$ for $0 < \theta < 1$. In this case, the interaction of particles j with all the particles in cluster C is approximated by a Taylor polynomial of order ℓ about the center of cluster C. We now derive, for the cluster-particle algorithm, the formulae for the approximations of (1) the potential at position $\mathbf{x}_i$ in cluster C due to M source particles at position $\mathbf{y}_j$, (2) the potential energy due to the interactions between particle i in cluster C and M source particles at position $\mathbf{y}_j$, and (3) the force on particle i in cluster C due to M source particles at position $\mathbf{y}_j$. Formulae for the electric field and torque are derived in Subsection 2 of the Appendix.

1. Approximation of the potential

The exact potential at position $\mathbf{x}_i$ in cluster C due to the source particles at $\mathbf{y}_j$ is

$$V_{i,C} = \sum_{j=1}^{M} \hat{L}_j \phi(\mathbf{x}_i, \mathbf{y}_j) = \sum_{j=1}^{M} \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_j^{\mathbf{s}} \partial_j^{\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_j). \tag{13}$$

In the case of a well-separated cluster-particle interaction, this potential is approximated by a Taylor expansion of the kernel about $\mathbf{x} = \mathbf{x}_c$ to obtain

$$\sum_{j=1}^{M} \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_j^{\mathbf{s}} \partial_j^{\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_j) \approx \sum_{j=1}^{M} \sum_{||\mathbf{k}||=0}^{\ell} \frac{1}{\mathbf{k}!} \partial_{\mathbf{x}}^{\mathbf{k}} \left(\sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_j^{\mathbf{s}} \partial_j^{\mathbf{s}} \phi(\mathbf{x}_c, \mathbf{y}_j) \right) (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \tag{14a}$$

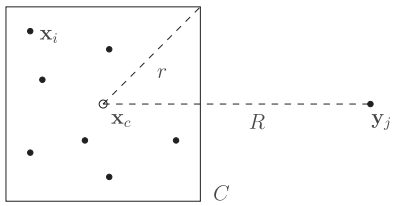

FIG. 3. A cluster-particle interaction between particles at $\mathbf{x}_i$ in cluster C and a source particle $\mathbf{y}_j$. The cluster has center $\mathbf{x}_c$ and radius r, and the cluster-particle distance is R.

$$= \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{s}||=0}^{p} \sum_{j=1}^{M} \frac{(-1)^{||\mathbf{k}||}}{\mathbf{k}!} \mathcal{M}_j^{\mathbf{s}} \partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}} \phi(\mathbf{x}_c, \mathbf{y}_j)(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \tag{14b}$$

$$= \sum_{||\mathbf{k}||=0}^{\ell} m_{\mathbf{k},C}^{\mathbf{s}} (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}}. \tag{14c}$$

Equation (14c) is a power series of degree ℓ in powers of $(\mathbf{x}_i - \mathbf{x}_c)$ with coefficients

$$m_{\mathbf{k},C}^{\mathbf{s}} = \frac{(-1)^{||\mathbf{k}||}}{\mathbf{k}!} \sum_{||\mathbf{s}||=0}^{p} \sum_{j=1}^{M} \mathcal{M}_j^{\mathbf{s}} \partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}} \phi(\mathbf{x}_c, \mathbf{y}_j)$$
$$= (-1)^{||\mathbf{k}||} \sum_{j=1}^{M} \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_j^{\mathbf{s}} b_{\mathbf{k}}^{\mathbf{s}}(\mathbf{x}_c, \mathbf{y}_j). \tag{15}$$

2. Approximation of the energy

Similarly, the potential energy due to the cluster-particle interaction is evaluated exactly as

$$U_{i,C} = \sum_{j=1}^{M} \hat{L}_i \hat{L}_j \phi(\mathbf{x}_i, \mathbf{y}_j)$$
$$= \sum_{j=1}^{M} \sum_{||\mathbf{n}||=0}^{p} \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_i^{\mathbf{n}} \mathcal{M}_j^{\mathbf{s}} \partial_i^{\mathbf{n}} \partial_j^{\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_j). \tag{16}$$

By using the definitions in Eqs. (3), (13), and (14b) and observing that for a pair potential kernel $\phi(\mathbf{x}_i, \mathbf{y}_j)$, $\partial_i^{\mathbf{n}} \phi = (-1)^{||\mathbf{n}||} \partial_j^{\mathbf{n}} \phi = \partial_{\mathbf{y}}^{\mathbf{n}} \phi$, we can approximate the energy due to a well-separated particle-cluster interaction to ℓth order as

$$\sum_{j=1}^{M} \sum_{||\mathbf{n}||=0}^{p} \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_i^{\mathbf{n}} \mathcal{M}_j^{\mathbf{s}} \partial_i^{\mathbf{n}} \partial_j^{\mathbf{s}} \phi(\mathbf{x}_i, \mathbf{y}_j)$$
$$\approx \sum_{||\mathbf{n}||=0}^{p} \mathcal{M}_i^{\mathbf{n}} \partial_i^{\mathbf{n}} \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{s}||=0}^{p} \sum_{j=1}^{M} \frac{(-1)^{||\mathbf{k}||}}{\mathbf{k}!}$$
$$\times \mathcal{M}_j^{\mathbf{s}} \partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}} \phi(\mathbf{x}_c, \mathbf{y}_j)(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}}$$
$$= \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{n}||=0}^{p} (-1)^{||\mathbf{n}||} \mathcal{M}_i^{\mathbf{n}} \frac{(-1)^{||\mathbf{k}||}}{\mathbf{k}!}$$
$$\times \sum_{||\mathbf{s}||=0}^{p} \sum_{j=1}^{M} \mathcal{M}_j^{\mathbf{s}} \partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}+\mathbf{n}} \phi(\mathbf{x}_c, \mathbf{y}_j)(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}}$$
$$= \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{n}||=0}^{p} (-1)^{||\mathbf{n}||} \mathcal{M}_i^{\mathbf{n}} m_{\mathbf{k},C}^{\mathbf{s}+\mathbf{n}} (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}}. \tag{17}$$

3. Approximation of the force

The force on particle i in cluster C due to source particles j is $\mathbf{f}_{i,C} = -\nabla_{\mathbf{x}_i} U_{i,C}$, which is approximated as

$$\mathbf{f}_{i,C} \approx - \sum_{||\mathbf{k}||=0}^{\ell} \sum_{||\mathbf{n}||=0}^{p} (-1)^{||\mathbf{n}||} \mathcal{M}_i^{\mathbf{n}} m_{\mathbf{k},C}^{\mathbf{s}+\mathbf{n}} \nabla_{\mathbf{x}_i} (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \tag{18}$$

$$= - \sum_{||\mathbf{k}||=0}^{\ell} \left(\sum_{||\mathbf{n}||=0}^{p} (-1)^{||\mathbf{n}||} \mathcal{M}_i^{\mathbf{n}} m_{\mathbf{k},C}^{\mathbf{s}+\mathbf{n}} \right) \begin{bmatrix} k_1 (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}-e_1} \\ k_2 (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}-e_2} \\ k_3 (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}-e_3} \end{bmatrix}, \tag{19}$$

where $e_1 = \langle 1, 0, 0 \rangle$, $e_2 = \langle 0, 1, 0 \rangle$, and $e_3 = \langle 0, 0, 1 \rangle$.

4. The cluster-particle algorithm

The algorithm requires the user to specify the order of Taylor approximation, ℓ, the MAC, θ, and the maximum number of particles in a leaf, N_0. The tree of target particles is then constructed. Assume that the tree has L levels, then the root with all the particles can be labeled as level 1 and the deepest level containing leaves is level L. A cluster at level l, $1 \le l \le L$, is labeled as C_l where the subscript refers to the level of the tree at which the cluster is located. Clearly, all the particles $\mathbf{x}_i$ belong to the root cluster, C_1, which is the only cluster on level 1. In addition, each particle belongs to a nested sequence of clusters, $\mathbf{x}_i \in C_L \subseteq \cdots \subseteq C_1$. The effect of a source particle $\mathbf{y}_j$ on the targets in a cluster is computed by Eq. (14c) if the cluster and the source particle are well separated and by Eq. (13) otherwise, if the cluster is a leaf. The power series coefficient in Eq. (15) contains the effect of all well-separated sources on the cluster. The pseudo-code for cluster-particle[38,46] is given in Algorithm 2. The cluster-particle algorithm has two stages after the tree is constructed.

In stage 1, the code loops through the N source particles and interacts each source particle $\mathbf{y}_j$ with the tree through a call to the subroutine **compute-cp1** in **line 5**. The interaction with the root does not satisfy the MAC; as such, the algorithm descends to the next level. If a target cluster and the source particle are well-separated, then the power series coefficients $m_{\mathbf{k},C}^{\mathbf{s}}$ in Eq. (15) are updated. Otherwise the algorithm descends to the children of the cluster and calls itself recursively. When

Algorithm 2. Cluster-particle treecode.

```
1  program cp-treecode
2    input: N particles x_i, multipoles M_i, parameters p, θ, N_0
3    output: potential, electric field, energies, forces, torques
4    Set {y_j}_{j=1}^N = {x_i}_{j=1}^N; construct tree of target particles x_i
5    for j = 1, N; compute-cp1(root, y_j); end
6    compute-cp2(root)
7  end program
8  subroutine compute-cp1(C, y)
9    if MAC is satisfied
10     update power series coefficients m_{k,C}^s by Eq. (15)
11   else if C is a leaf
12     use Eq. (13) to compute interactions directly
13   else for each child of C
14     compute-cp1(child,y)
15 end subroutine
16 subroutine compute-cp2(C)
17   if C interacted with a source particle by Taylor approximation in stage 1
18     loop through target sites x_i in C
19     approximate interactions using power series in Eq. (14c)
20   for each child of C
21     compute-cp2(child)
22 end subroutine
```

a leaf C_L is reached and the MAC is not satisfied, then the interaction is computed by direct summation for all target sites $\mathbf{x}_i \in C_L$ using Eq. (13).

In stage 2, the algorithm calls the subroutine **compute-cp2** in **line 6**. The subroutine descends the tree, checks to see if a cluster interacted with a well-separated source in stage 1, and if true, evaluates Eq. (14c) for all particles in the cluster. The algorithm is complete after stage 2.

5. Recurrence relation

The moments in Eq. (9) for the particle-cluster algorithm require multi-dimensional derivatives of the kernel. Similarly, for the cluster-particle algorithm, the coefficients in Eq. (15) require multi-dimensional derivatives of the kernel. In previous work,[23] we derived a recurrence relation for multi-dimensional derivatives of the power-law kernel, $\phi(\mathbf{x},\mathbf{y}) = \frac{1}{|\mathbf{x}-\mathbf{y}|^\nu}$. In this work, we consider the Coulomb pair potential for which $\nu = 1$. Define $\mathbf{x} = \langle x_1, x_2, x_3\rangle$, $\mathbf{y} = \langle y_1, y_2, y_3\rangle$, and the $\mathbf{k}$th multi-dimensional derivative of $\phi(\mathbf{x},\mathbf{y}) = \frac{1}{|\mathbf{x}-\mathbf{y}|}$ as $\partial_{\mathbf{y}}^{\mathbf{k}}\phi(\mathbf{x},\mathbf{y}) = \partial_{\mathbf{y}}^{\mathbf{k}}\phi$. Then the recurrence relation for the $\mathbf{k}$th multi-dimensional derivative is given as

$$\partial_{\mathbf{y}}^{\mathbf{k}}\phi = \frac{1}{|\mathbf{x}-\mathbf{y}|}\left\{\left(2-\frac{1}{||\mathbf{k}||}\right)\sum_{i=1}^{3} k_i(x_i-y_i)\partial_{\mathbf{y}}^{\mathbf{k}-\mathbf{e}_i}\phi + \left(\frac{1}{||\mathbf{k}||}-1\right)\sum_{i=1}^{3} k_i(k_i-1)\partial_{\mathbf{y}}^{\mathbf{k}-2\mathbf{e}_i}\phi\right\}, \tag{20}$$

where $e_1 = \langle 1,0,0\rangle$, $e_2 = \langle 0,1,0\rangle$, and $e_3 = \langle 0,0,1\rangle$. The function value, $\mathbf{k} = 0$, is computed explicitly and used as the starting value for the recurrence to compute the $||\mathbf{k}|| \geq 1$ terms. In the recurrence, the derivatives for $||\mathbf{k}|| < 0$ are set to zero. For the particle-cluster algorithm, $\phi(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x}_i,\mathbf{y}_c)$, and for cluster-particle, $\phi(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x}_c,\mathbf{y}_j)$.

6. Implementation details

The algorithms were written in Fortran 90 and the codes are available online[48,49] under the GNU General Public License. The tests were performed on a Dell PowerEdge R730 Linux box with two 2.3 GHz Intel Xeon processors each with 36 threads and 32 GB memory. The code was compiled with a GNU Fortran compiler and used the -O3 optimization.

IV. NUMERICAL RESULTS

In this section, we present results from applying the two algorithms to evaluate the free space multipolar Coulomb potential force at N sites due to the interaction between the particles for $N \in \{10^4, 10^5, 10^6\}$. The particles are randomly distributed in the unit cube $[-0.5, 0.5]^3$, and the multipoles were generated randomly from the interval $(-1, 1)$. In Secs. IV A–IV C, the results are for tests where only the potential was computed. The tests in Sec. IV D evaluated both the potential and the force.

A. Efficiency of the treecode methods—Accuracy and CPU time

To study the efficiency of our two methods, we used point multipoles up to quadrupoles ($p = 2$) and hexadecapoles ($p = 4$), in Eq. (5) or (13). The algorithms used the following parameter set for the tree: $\theta \in \{0.5, 0.75, 0.9\}$ as the acceptance criterion for approximating an interaction with a Taylor approximation, $\ell \in \{0, 2, 4, 6, 8, 10, 12\}$ for the order of the approximation, and $N_0 = 500$ as the maximum number of particles in a leaf during construction of the tree. The potential at a target site due to a well-separated interaction is approximated by Eq. (7c) in particle-cluster and by Eq. (14c) in cluster-particle. The root mean square error, E, of the approximation is evaluated as

$$\mathrm{E} = \left(\frac{\sum_{i=1}^{N}\left|V(\mathbf{x}_i) - \hat{V}(\mathbf{x}_i)\right|^2}{\sum_{i=1}^{N}|V(\mathbf{x}_i)|^2}\right)^{1/2}, \tag{21}$$

where $V(\mathbf{x}_i)$ is the exact potential at $\mathbf{x}_i$ defined in Eq. (5) or Eq. (13) and $\hat{V}(\mathbf{x}_i)$ is the approximation.

Figures 4 and 5 show the accuracy and timing results for the serial implementation of both algorithms for multipolar orders $p = 4$ and $p = 2$, respectively. The top rows of both Figs. 4 and 5 show the root mean square error, E, for both algorithms for different orders of Taylor approximation, $\ell \in \{0, 2, 4, 6, 8, 10, 12\}$, and different system sizes, $N \in \{10^4, 10^5, 10^6\}$. In the plots in the top row, the order of approximation increases downwards. Both algorithms achieve extremely good accuracy for all orders, MACs, and system sizes. As expected, the error decreases with increasing order of the approximation. One quirk of our results is that for all orders, the error decreases with increasing system size. This is because we used the same parameter, $N_0 = 500$, for all system sizes. A larger system size, with a fixed N_0, means a deeper tree with more levels and leaves. This leads to more direct summation and hence an increase in accuracy. We also see that for all orders, the tests with the smallest MAC, $\theta = 0.5$, has the least error and the tests with the largest MAC, $\theta = 0.9$, shows the largest error. A small MAC means fewer interactions between particles and cluster satisfy the multipole acceptability criterion and thus more interactions between particles and clusters are computed exactly. This explains the higher accuracy for $\theta = 0.5$. The trade-off is more CPU time.

The bottom rows of Figs. 4 and 5 are a plot of the CPU times, in seconds, it took for the particle-cluster and cluster-particle methods to generate the corresponding result in the top row. Also included is the CPU time for an efficient direct sum computation. In these plots, the Taylor approximation increases from bottom to top. For both algorithms, the CPU time increases with increasing order of approximation. In addition, for each MAC, we see excellent speedup in CPU time for the treecode algorithms over the direct method as the system size gets larger. A larger MAC results in faster compute time but lower accuracy. Table I provides further evidence of the efficiency of the tree algorithms over direct summation. Table I(a) is the direct sum CPU time for computing the potential for $N \in \{10^4, 10^5, 10^6\}$ point-multipoles with multipole order $p = 4$. Tables I(b) and I(c) are the corresponding CPU times for particle-cluster and cluster-particle for Taylor approximation order $\ell \in \{0, 4, 8\}$ and MAC parameter,

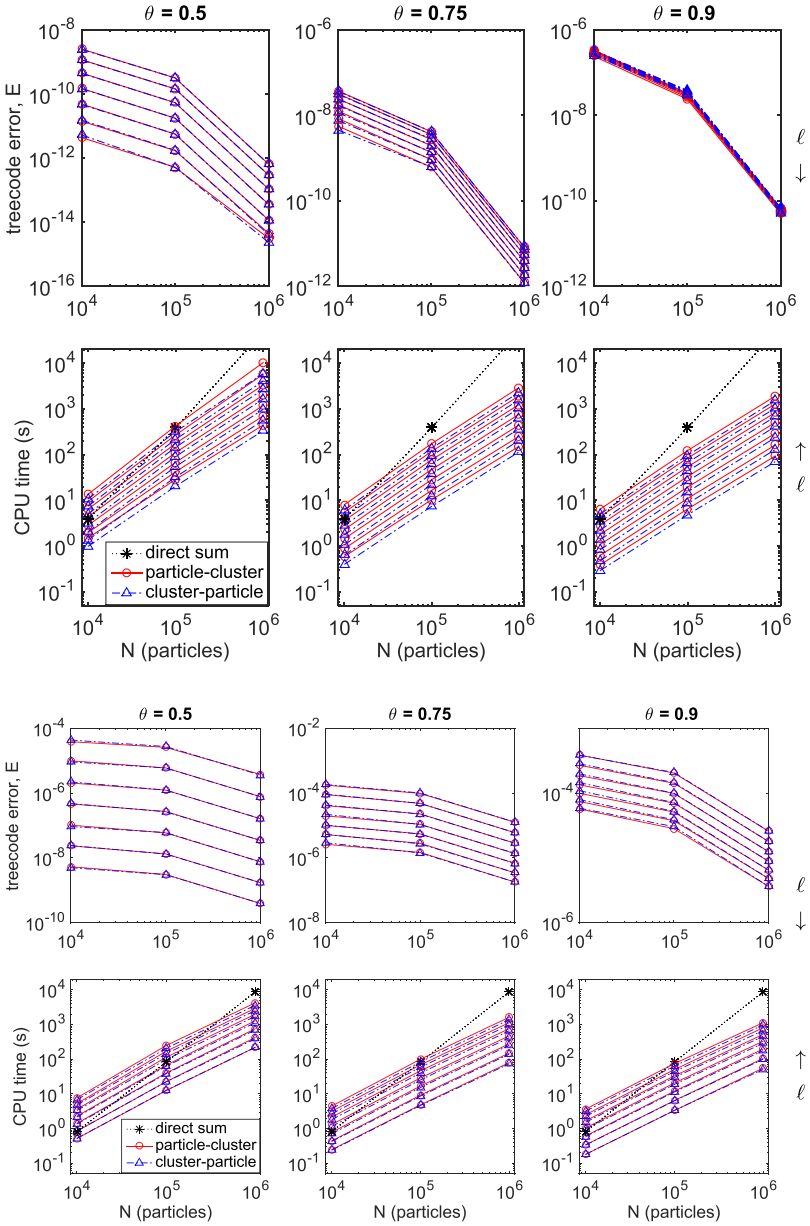

FIG. 4. Comparison of the treecode methods to direct summation. Top row: Error (E), bottom row: CPU time (s). Order of multipole, $p = 4$. $N \in \{10^4, 10^5, 10^6\}$, $\theta \in \{0.5, 0.75, 0.9\}$, $N_0 = 500$, and $\ell = 0 : 2 : 12$.

FIG. 5. Comparison of the treecode methods to direct summation. Top row: Error (E), bottom row: CPU time (s). Order of multipole, $p = 2$. $N \in \{10^4, 10^5, 10^6\}$, $\theta \in \{0.5, 0.75, 0.9\}$, $N_0 = 500$, and $\ell = 0 : 2 : 12$.

$\theta = 0.75$. The treecode algorithms are faster than direct sum with errors between 10^{-12} and 10^{-6}. For example, for $N = 10^5$ when the order of approximation $\ell = 4$, the error for both particle-cluster and cluster-particle is less than 10^{-8} and the CPU time of 28.127 s and 22.065 s, respectively, is more than *ten times faster* than the CPU time of direct sum which is 382.021 s. For $N = 10^6$, the zero order approximation for both algorithms when $p = 4$ has an error which is less than 10^{-10} and the CPU times are more than *three hundred times faster* than the CPU time of direct summation.

The top rows of Figs. 4 and 5 show that the errors produced by both algorithms are lower for multipolar order $p = 4$ compared to the error when $p = 2$ for the same order of approximation. This is expected. The treecodes approximate the far-field interactions between particles and clusters. The Coulomb potential decays faster for $p = 4$ compared to $p = 2$. As a result, the effect of far-field interactions is smaller for $p = 4$ than for $p = 2$ and correspondingly the error produced from the Taylor approximation is smaller for $p = 4$ compared to that for $p = 2$. Higher Taylor approximation orders, $\ell > 12$, are required to achieve higher accuracy for lower multipolar orders. Table II shows results for $p = 2$ with $N = 10^6$,

TABLE I. CPU time (s) for N particles, $N \in \{10^4, 10^5, 10^6\}$, with multipole order $p = 4$, and Taylor approximation order $\ell \in \{0, 4, 8\}$. (a) Direct sum, [(b) and (c)] treecodes with MAC parameter $\theta = 0.75$.

	ℓ	(a) Direct sum	(b) Particle-cluster	(c) Cluster-particle
	0		0.590	0.399
$N = 10^4$	4	3.822	1.444	1.063
	8		3.600	2.811
	0		10.282	7.351
$N = 10^5$	4	382.021	28.127	22.065
	8		76.251	61.403
	0		148.896	112.880
$N = 10^6$	4	46 926.447	436.724	361.465
	8		1221.899	1014.569

TABLE II. Error (E) and CPU time (s) for $N = 10^6$ particles, $N_0 = 500$, $\theta = 0.75$, with multipole order $p = 2$, and Taylor approximation order $\ell \in \{12, 16, 20, 24\}$. (a) Direct sum, [(b) and (c)] treecodes with MAC parameter $\theta = 0.75$.

		(b) Particle-cluster		(c) Cluster-particle	
ℓ	(a) Direct sum	Error (E)	CPU time (s)	Error (E)	CPU time (s)
12	8777.380	1.82×10^{-7}	1633.852	1.83×10^{-7}	1380.015
16		5.10×10^{-8}	3065.121	5.06×10^{-8}	2515.155
20		1.48×10^{-8}	5584.925	1.49×10^{-8}	4206.313
22		4.32×10^{-9}	10 043.635	4.50×10^{-9}	6656.17

TABLE III. Error (E) and CPU time (s) for $N = 10^7$ particles, $N_0 = 500$, $\theta = 0.75$, with multipole order $p = 2$, and Taylor approximation order $\ell \in \{12, 16, 20, 24\}$. (a) Parallel direct sum, [(b) and (c)] parallel treecodes with MAC parameter $\theta = 0.75$. All three methods were run over 32 processors.

		(b) Particle-cluster		(c) Cluster-particle	
ℓ	(a) Direct sum	Error (E)	CPU time (s)	Error (E)	CPU time (s)
0	83 345.388	1.11×10^{-5}	58.667	1.71×10^{-5}	57.231
4		2.31×10^{-6}	203.686	2.31×10^{-6}	183.638
8		5.52×10^{-7}	690.703	5.52×10^{-7}	473.586
12		1.46×10^{-7}	1689.673	1.46×10^{-7}	1006.529
16		4.06×10^{-8}	3270.021	4.05×10^{-8}	1899.527
20		1.17×10^{-8}	5685.749	1.17×10^{-8}	3169.747

$N_0 = 500$, $\theta = 0.75$, and $\ell \in \{12, 16, 20, 24\}$. We see an increase in accuracy over the $\theta = 0.75$ results in Fig. 5 with the increase in the order of approximation. Although there is a corresponding increase in CPU time, both algorithms produce errors of about 10^{-8} in half the time required for direct summation and are thus efficient replacements for direct summation.

The efficiency of treecode algorithms increases with increasing system size. Table III shows results for parallel versions of direct sum and the two algorithms over 32 processors with $p = 2$ with $N = 10^7$, $N_0 = 500$, $\theta = 0.75$, and $\ell \in \{0, 4, 8, 12, 16, 20\}$. We see an increase in the efficiency of the treecode algorithms for $N = 10^7$ compared to $N = 10^6$ in Table II. Both algorithms produce errors similar to the errors for $N = 10^6$ but are more than an order of magnitude faster than the direct sum.

Section IV C provides more details and results on the parallelization of the two algorithms.

1. Parameter space of the treecode algorithms

To study the parameter space of the treecode algorithms, we plotted the CPU times for different approximation orders against the corresponding errors for different MAC parameters, $\theta \in \{0.5, 0.75, 0.9\}$. Figure 6 is one such plot with $N_0 = 500$ and $N \in \{10^4, 10^5, 10^6\}$. In these plots, the approximation order ℓ increases from bottom to top. The plots show that for a fixed system size, N, and a fixed order of approximation, ℓ, the accuracy of both algorithms increases with decreasing MAC parameter, θ. The usefulness of the plot is that it is a guide to select a set of parameters to efficiently achieve a desired accuracy level. For example, for $N = 10^6$, the plot shows that an error of 10^{-12} can be achieved with two different parameter sets as follows:

1. A parameter set of $N_0 = 500$, $\theta = 0.75$, and $\ell = 12$.
2. A parameter set of $N_0 = 500$, $\theta = 0.5$, and $\ell = 0$.

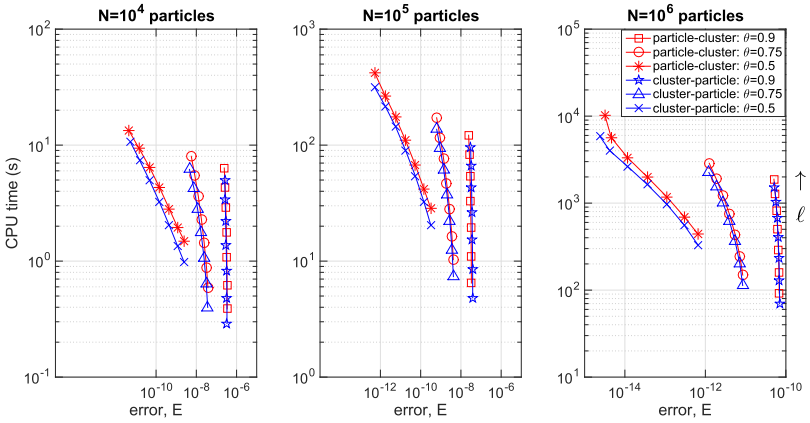

FIG. 6. A plot of CPU time against error for $p = 4$, $N \in \{10^4, 10^5, 10^6\}$, $\theta \in \{0.5, 0.75, 0.9\}$, $N_0 = 500$, and $\ell = 0 : 2 : 12$.

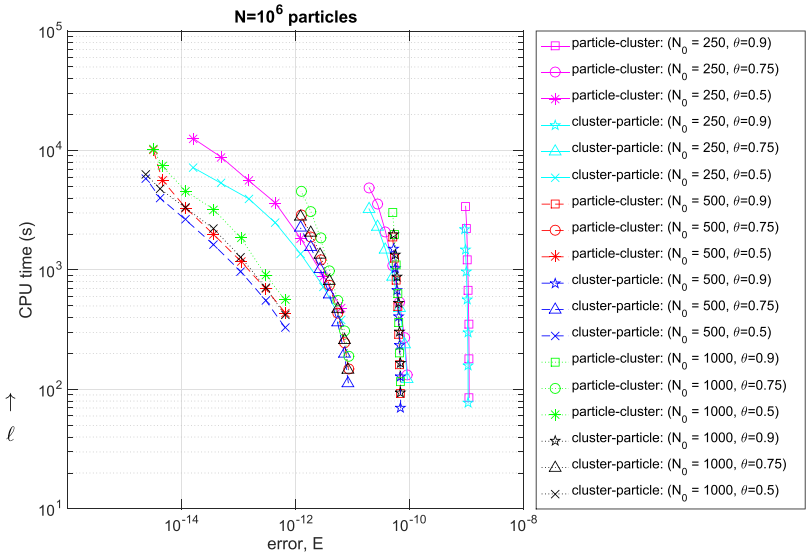

FIG. 7. A plot of CPU time against error for $p = 4$, $N = 10^6$, $\theta \in \{0.5, 0.75, 0.9\}$, $N_0 \in \{250, 500, 1000\}$, and $\ell = 0 : 2 : 12$.

The more efficient choice of parameters is set 2 $(\theta, \ell) = (0.5, 0)$ which gives an accuracy better than 10^{-12} with CPU time about an order of magnitude faster than parameter set 1.

Figures 7 and 8 show the result for $p = 4$ and $p = 2$, respectively, with fixed $N = 10^6$ and different $N_0 \in \{250, 500, 1000\}$. In these plots, the approximation order, ℓ, increases from bottom to top as well. The plots show that in general for both algorithms, $N_0 = 500$ produces the best accuracy for all MAC parameters, $\theta \in \{0.5, 0.75, 0.9\}$, when $N = 10^6$ and $N_0 = 250$ is the least accurate choice. The efficiency of $N_0 = 1000$ lies between that of $N_0 = 250$ and $N_0 = 1000$. The effect of N_0 is not as systematic as that of θ or ℓ. For each simulation system, a near optimal N_0 is chosen by performing a few tests with different N_0's and producing a plot similar to Figs. 7 and 8.

B. Comparison of CPU times for different multipole orders

Figures 4–8 and Table I show that although particle-cluster and cluster-particle have very similar behavior, cluster-particle is in general faster for multipolar order $p = 4$. Boateng and Krasny[38] have shown that for a system of N point charges, $p = 0$, particle-cluster is faster than cluster-particle. However, for point multipoles, $p \geq 1$, the formulations of the two algorithms in Eqs. (7c) and (14c) make cluster-particle the faster algorithm. For particle-cluster, the approximation of the potential at position $\mathbf{x}_i$ by Eq. (7c) has an explicit double sum over $||\mathbf{k}||$ and $||\mathbf{s}||$. This is because each Taylor coefficient, $b^{\mathbf{s}}_{\mathbf{k}}$ defined in Eq. (8), is unique to a target position, $\mathbf{x}_i$, and depends on $\mathbf{x}_i$. To evaluate Eq. (7c) for each target site requires repeated memory access of the multi-dimensional array that stores the Taylor coefficient $b^{\mathbf{s}}_{\mathbf{k}}(\mathbf{x}_i, yc)$ for each $\mathbf{k}$ value of the outer sum. For cluster-particle, Eq. (14c) is just a single sum. This is because the coefficients, $m^{\mathbf{s}}_{\mathbf{k},C}$ defined in Eq. (15), depend only on the center of the clusters and not on the target sites inside the cluster. As a result, the sum over $||\mathbf{s}||$ to form $m^{\mathbf{s}}_{\mathbf{k},C}$ is completed before evaluating Eq. (14c). This reduces the memory access requirements of cluster-particle and makes it faster than particle-cluster for $p \geq 1$. Figure 9 is a plot of the CPU time of particle-cluster against the CPU time of cluster-particle for different multipolar orders, $p \in \{0, 1, 2, 4\}$. The plots are for

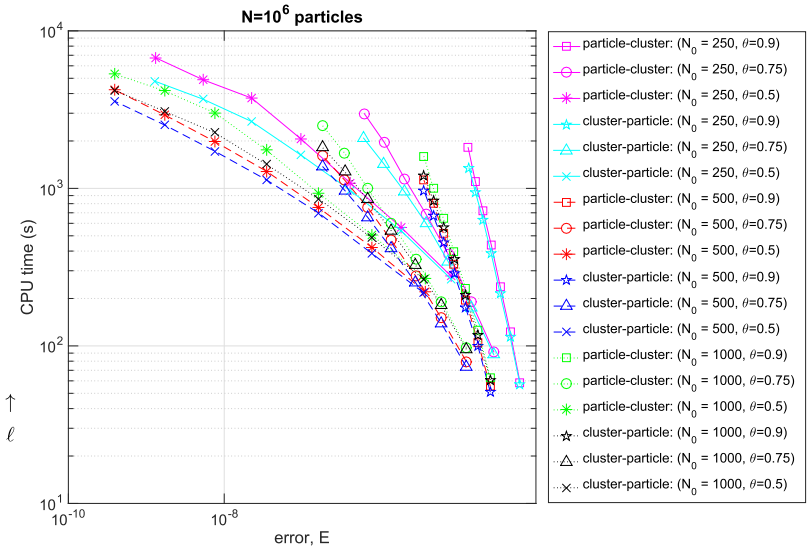

FIG. 8. A plot of CPU time against error for $p = 2$, $N = 10^6$, $\theta \in \{0.5, 0.75, 0.9\}$, $N_0 \in \{250, 500, 1000\}$, and $\ell = 0 : 2 : 12$.

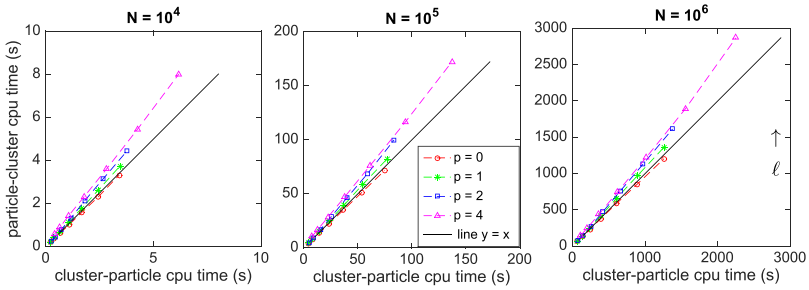

FIG. 9. A plot of particle-cluster CPU time against cluster-particle CPU time for multipolar orders $p \in \{0, 1, 2, 4\}$. Tree parameters: $\theta = 0.75$, $N \in \{10^4, 10^5, 10^6\}$, $N_0 = 500$, $\ell = 0 : 2 : 12$.

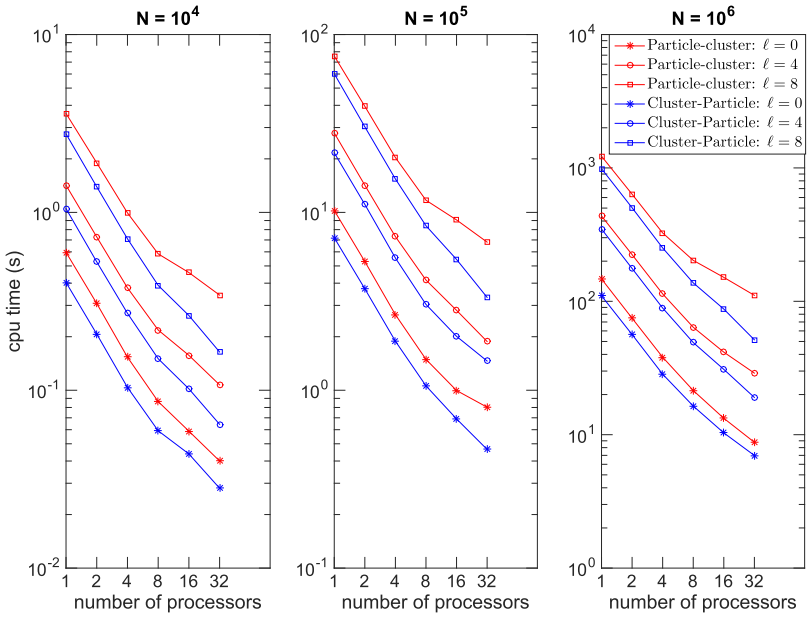

FIG. 10. Parallel scaling of the treecodes; $p = 4$, $\theta = 0.75$, $N \in \{10^4, 10^5, 10^6\}$, and $\ell \in \{0, 4, 8\}$.

systems sizes $N \in \{10^4, 10^5, 10^6\}$, MAC parameter $\theta = 0.75$, $N_0 = 500$, and $\ell = 0 : 2 : 12$. Also plotted is the line $y = x$.

As expected, particle-cluster is slightly faster than cluster-particle for all orders of approximation for the zero order multipole since the plots for $p = 0$ is below the line $y = x$. But for $p \geq 1$, the lines are all above the $y = x$ line which shows that cluster-particle is faster for multipole orders of one or larger.

C. Parallelisation

This section presents results on the parallelisation of the treecode algorithms on a modest number of processors. There are several approaches to parallelising the treecode.[50–52] A replicated data approach was employed for the parallelisation in this work with the message passing interface (MPI). Each processor of the cluster had a copy of the particles and built a copy of the tree. Load balancing was achieved by splitting the targets, in the case of particle-cluster, and the sources, in the case of cluster-particle, over the processors in **line 5** of Algorithms 1 and 2, respectively. A global sum of potentials was performed at the end for each algorithm. Figure 10 is a plot of CPU time versus number of processors $P = 2^j$, j: 0: 1: 5, for both algorithms for system sizes $N \in \{10^4, 10^5, 10^6\}$ with multipolar order $p = 4$. The MAC parameter $\theta = 0.75$, $N_0 = 500$, and approximation orders $\ell \in \{0, 4, 8\}$. The figure shows that both algorithms, for all approximation orders, achieve good parallel speedup for small system size as well as large system size. Table IV provides numerical evidence of the parallel speedup of both algorithms. The compute time on different processor counts is given for both algorithms for $N = 10^6$ and the order of approximation $\ell = 8$. We compute the parallel speedup

$$\sigma = \frac{t_1}{t_P}, \tag{22}$$

where t_1 is the CPU time on a single processor and t_P is the CPU time on P processors.

The table shows that cluster-particle has better scaling over the 32 processors for the parameter regime studied. We ascribe the difference in parallel speedup of the two algorithms to the difference in formulations of Eqs. (7c) and (14c) and effects as explained in Sec. IV B. In future work, we will study the parallel versions of the algorithms in detail.

TABLE IV. CPU time (s) and speedup (σ) for $N = 10^6$ particles, on $P = 2^j$, j: 0: 1: 5 processors, with multipole order $p = 4$, Taylor approximation order $\ell = 8$, and MAC parameter $\theta = 0.75$.

	Particle-cluster		Cluster-particle	
P (number of processors)	t_P	σ	t_P	σ
1	1224.077	1	986.881	1
2	635.232	1.93	502.856	1.96
4	325.573	3.76	251.329	3.93
8	202.831	6.03	137.571	7.17
16	151.196	8.10	87.459	11.28
32	110.870	11.04	51.431	19.19

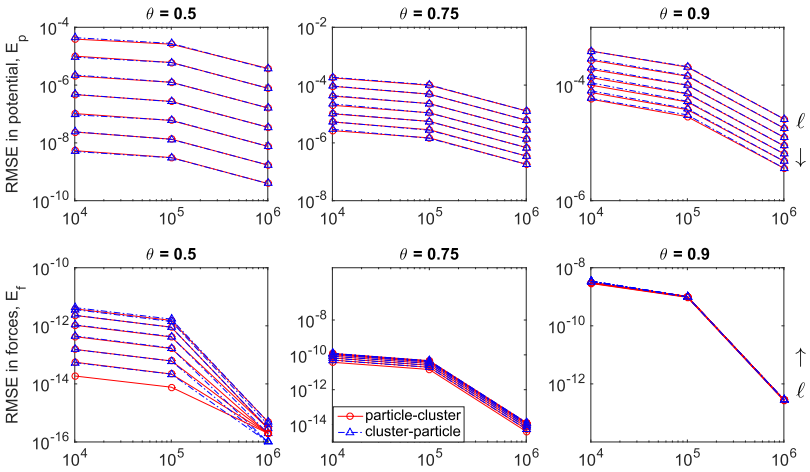

FIG. 11. Errors in potential and force. Top row: Errors in potential (E_p). Bottom row: Errors in force (E_f). Order of multipole, $p = 2$. $N \in \{10^4, 10^5, 10^6\}$, $\theta \in \{0.5, 0.75, 0.9\}$, $N_0 = 500$, and $\ell = 0:2:12$.

D. Mixed multipolar rank (orders)

In this section, we look at the performance of the algorithms in a regime where the point-multipoles have different orders $p \in \{0, 1, 2\}$. In the test cases presented here, the algorithms compute both the potential and the force using the relevant formulae in Eqs. (7c), (12), (14c), and (19). The root mean square error, E_p, of the approximation of the potential is evaluated as

$$E_p = \left(\frac{\sum_{i=1}^{N} \left| V(\mathbf{x}_i) - \hat{V}(\mathbf{x}_i) \right|^2}{\sum_{i=1}^{N} |V(\mathbf{x}_i)|^2} \right)^{1/2}, \tag{23}$$

where $V(\mathbf{x}_i)$ is the exact potential at $\mathbf{x}_i$ and $\hat{V}(\mathbf{x}_i)$ is the approximation. In addition, the root mean square error, E_f, in the force approximation is computed as

$$E_f = \left(\frac{\sum_{i=1}^{N} \left| \mathbf{f}_i - \hat{\mathbf{f}}_i \right|^2}{\sum_{i=1}^{N} |\mathbf{f}_i|^2} \right)^{1/2}, \tag{24}$$

where $\mathbf{f}_i$ is the exact force on particle i and $\hat{\mathbf{f}}_i$ is the approximation.

1. Full rank test case

In order to have a base for comparison, we first present results for a test case where all the point-multipoles have order 2, i.e., $p = 2$. Figure 11 shows the root mean square errors in the potential and force for $N \in \{10^4, 10^5, 10^6\}$ with order of approximation $\ell \in \{0, 2, 4, 6, 8, 10, 12\}$, $N_0 = 500$, and MAC parameters $\theta \in \{0.5, 0.75, 0.9\}$.

The errors in the potential shown in the top row of Fig. 11 are the same as the errors shown in the top row of Fig. 5. The errors in the force are much smaller than the order in the potential. This is as expected since for multipolar order $p = 2$, the kernel for the force decays as $\frac{1}{r^6}$ while the error in the potential decays as $\frac{1}{r^3}$. This makes the algorithms very suitable for molecular dynamics simulations where the forces are required. We expect the approximations of the potential energy and torques to have errors on the order of the force errors. Figure 12 shows a comparison of the CPU times for the two algorithms to that for direct summation. The speedup of the two algorithms over direct summation is more pronounced when the force is computed compared to the bottom row results in the test case shown in Fig. 5 which includes only a

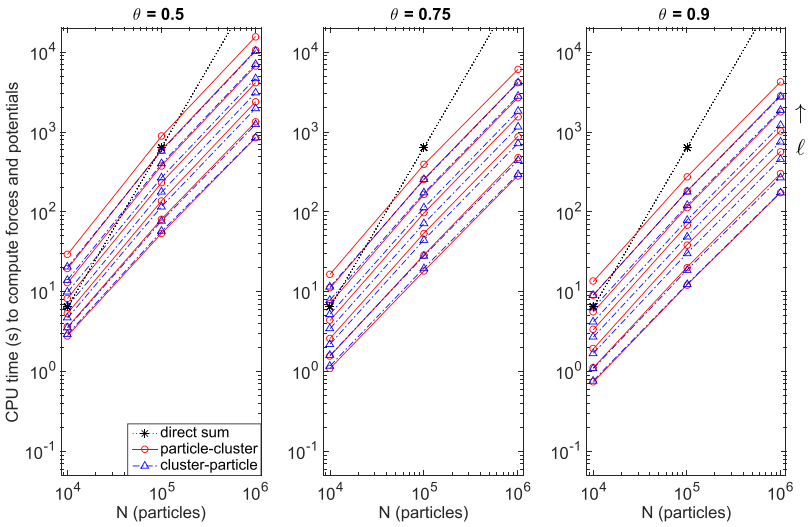

FIG. 12. Comparison of the CPU times (s) of the treecode methods to direct summation. Order of multipole, $p = 2$. $N \in \{10^4, 10^5, 10^6\}$, $\theta \in \{0.5, 0.75, 0.9\}$, $N_0 = 500$, and $\ell = 0:2:12$.

TABLE V. CPU time (s) for N particles, $N \in \{10^4, 10^5, 10^6\}$, with multipole order $p = 2$, and Taylor approximation order $\ell \in \{0, 4, 8\}$. (a) Direct sum, [(b) and (c)] treecodes with MAC parameter $\theta = 0.75$.

	ℓ	(a) Direct sum	(b) Particle-cluster	(c) Cluster-particle
	0		1.099	1.182
$N = 10^4$	4	6.459	2.610	2.213
	8		7.176	5.272
	0		18.185	19.764
$N = 10^5$	4	640.332	54.185	44.697
	8		165.247	112.706
	0		282.986	295.263
$N = 10^6$	4	66 001.855	878.905	718.255
	8		2673.246	1828.692

TABLE VI. Distribution of maximum multipolar order for N particles, $N \in \{10^4, 10^5, 10^6\}$.

	Charges ($p = 0$)	Dipoles ($p = 1$)	Quadrupoles ($p = 2$)
$N = 10^4$	3430	3284	3286
$N = 10^5$	33 289	33 380	33 331
$N = 10^6$	332 946	333 424	333 630

computation of the potential. Table V provides numerical results for the comparisons of CPU times.

For $N = 10^6$ particles and $\ell = 8$, both algorithms have an error of 10^{-10}. In this regime, particle-cluster is 24 times faster than direct sum, while cluster-particle is 36 times faster. Because of the fast decay of the force kernel, the zero order approximation for $N = 10^6$ has an error in the force of 10^{-10} for both algorithms as well, with a speedup of over 200 compared to direct summation.

2. Mixed rank

In this section, we present results for test cases where the N interacting point-multipoles have different maximum multipolar orders. Each point multipole has either just charges ($p = 0$), charges and dipoles ($p = 1$), or charges, dipoles, and quadrupoles ($p = 2$). Table VI shows the distribution of multipolar orders for system sizes $N \in \{10^4, 10^5, 10^6\}$. As an example, for the system with $N = 10^4$ particles, 3430 had only charges, 3284 had charges and dipoles, and 3286 had charges, dipoles, and quadrupoles.

TABLE VII. CPU time (s) for N particles, $N \in \{10^4, 10^5, 10^6\}$, with mixed multipole order $p \in \{0, 1, 2\}$, and Taylor approximation order $\ell \in \{0, 4, 8\}$. (a) Direct sum, [(b) and (c)] treecodes with MAC parameter $\theta = 0.75$.

	ℓ	(a) Direct sum	(b) Particle-cluster	(c) Cluster-particle
	0		0.558	0.592
$N = 10^4$	4	2.600	1.666	1.389
	8		4.910	3.413
	0		9.773	10.388
$N = 10^5$	4	270.293	34.142	28.016
	8		111.457	74.461
	0		159.296	157.076
$N = 10^6$	4	27 470.762	559.882	449.681
	8		1876.735	1229.321

We developed a mixed-rank version of direct summation as a base for comparison. The mixed rank version contains different direct sum routines specifically designed for the ranks of the interacting particles. As a result, the cost of interactions varies depending on the sum of the orders of the interacting point multipoles. This version was about 2.5 times faster than the full rank version [compare values in Tables V(a) and VII(a)]. It is conceptually straightforward to adapt the particle-cluster and cluster-particle algorithms for mixed rank interactions. The key is to adaptively change the upper limit of the summation over $||\mathbf{s}||$ and $||\mathbf{n}||$ in the formulae for the quantities in question, depending on the rank of the interacting particles.

Figure 13 shows the root mean square errors in the potential and force for $N \in \{10^4, 10^5, 10^6\}$ with order of approximation $\ell \in \{0, 2, 4, 6, 8, 10, 12\}$, $N_0 = 500$, and MAC parameters $\theta \in \{0.5, 0.75, 0.9\}$.

As expected, the errors in both the potential and force for the mixed rank are a little higher than in the full rank. This is because the mixed-rank test case includes low-rank interactions where the kernels for the potential and forces decay at a lower rate than $\frac{1}{r^3}$ and $\frac{1}{r^6}$, respectively, expected for full-rank interactions. Nevertheless, the force computations in

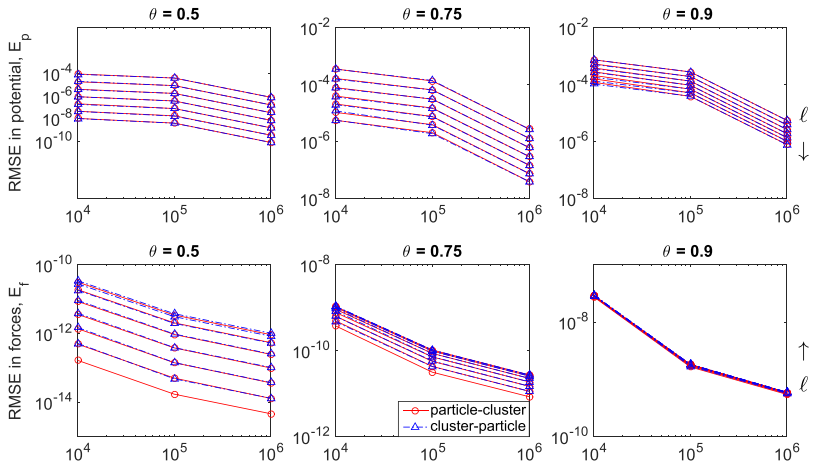

FIG. 13. Errors in potential and force for system with mixed ranks. Top row: Errors in potential (E_p). Bottom row: Errors in force (E_f). Order of multipoles, $p \in \{0, 1, 2\}$. $N \in \{10^4, 10^5, 10^6\}$, $\theta \in \{0.5, 0.75, 0.9\}$, $N_0 = 500$, and $\ell = 0 : 2 : 12$.

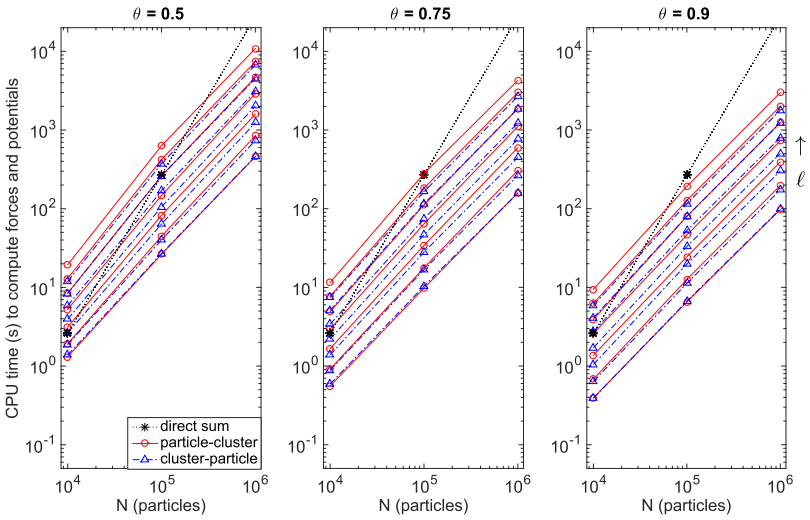

FIG. 14. Comparison of the CPU times (s) of the treecode methods to direct summation. Mixed multipole ranks, $p \in \{0, 1, 2\}$. $N \in \{10^4, 10^5, 10^6\}$, $\theta \in \{0.5, 0.75, 0.9\}$, $N_0 = 500$, and $\ell = 0 : 2 : 12$.

the mixed-rank case have high accuracy as well. Figure 14 compares the CPU times of the mixed-rank versions of the algorithms to direct sum. Both algorithms achieve excellent speedup over direct sum.

Table VII provides numerical results for the CPU times of direct sum and the treecode algorithms.

For $\theta = 0.75$ and order of approximation $\ell = 0$, both algorithms provide an error in the force of 10^{-8} with a speedup over direct sum of about 4.5 when $N = 10^4$, 26 when $N = 10^5$, and 172 when $N = 10^6$. Because the potential energy and torques have similar decay rate to the force, we expect the algorithms to be similarly efficient in approximating these quantities.

V. CONCLUSION

In this work, we have developed two mesh-free treecode algorithms we call particle-cluster and cluster-particle for free-space multipolar electrostatic interactions. We used the algorithms to compute the potential and force on different sets of interacting particles and compared the results to direct summation. We found both algorithms to be highly accurate in computing forces for full-rank and mixed-rank systems and offer substantial speedup over direct sum calculations.

We explicated the dependence of the performance of the algorithms on the MAC parameter θ and the order of approximation ℓ. We compared the two algorithms and explained why even though particle-cluster is faster than cluster-particle for point-charges, in their current construction, cluster-particle is faster and perhaps more efficient than particle-cluster for multipolar interactions.

Because a significant fraction of molecular simulations are run in parallel, we provided results on the parallel scaling of both algorithms over 32 processors. We found the algorithms to have good scaling over the modest number of processors.

We believe this work as an important step in the design of fast evaluation methods for Cartesian multipolar electrostatic interactions. The algorithms will be most useful for high order permanent Cartesian multipolar electrostatic interactions and can be implemented as part of the AMOEBA polarizable force-fields in MD software such as TINKER,[53] DL_POLY,[54] and AMBER.[55] In the future, the author hopes to extend this work to simulations in periodic boundary conditions and to other pair potentials.

ACKNOWLEDGMENTS

This research used resources, provided through the Director's Discretionary Program, of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract No. DE-AC02-06CH11357.

APPENDIX: ADDITIONAL DERIVATIONS AND A SKETCH OF CONVERGENCE OF THE POTENTIAL

In this section, we derive the formulae for approximating the electric field and torque within the particle-cluster and cluster-particle algorithms.

1. Formulae for particle-cluster approximations

a. Approximation of the electric field

The exact electric field at $\mathbf{x}_i$ is given as $\mathbf{E}_{i,C} = -\nabla_{\mathbf{x}_i} V_{i,C} = \nabla_{\mathbf{y}} V_{i,C}$ and it is approximated through the use of Eq. (7b) as

$$\mathbf{E}_{i,C} \approx -\nabla_{\mathbf{x}_i}\left(\sum_{||\mathbf{k}||=0}^{\ell}\sum_{||\mathbf{s}||=0}^{p}\frac{1}{\mathbf{k}!}\partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}}\phi(\mathbf{x}_i,\mathbf{y}_c)\sum_{\mathbf{y}_j\in C}\mathcal{M}_j^{\mathbf{s}}(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}}\right)$$

$$= \sum_{||\mathbf{k}||=0}^{\ell}\sum_{||\mathbf{s}||=0}^{p}\frac{1}{\mathbf{k}!}\nabla_{\mathbf{y}}\left(\partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}}\phi(\mathbf{x}_i,\mathbf{y}_c)\right)H_{\mathbf{k}}^{\mathbf{s}}(C)$$

$$= \sum_{||\mathbf{k}||=0}^{\ell}\sum_{||\mathbf{s}||=0}^{p}\frac{1}{\mathbf{k}!}\begin{bmatrix}\partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}+e_1}\phi(\mathbf{x}_i,\mathbf{y}_c)\\ \partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}+e_2}\phi(\mathbf{x}_i,\mathbf{y}_c)\\ \partial_{\mathbf{y}}^{\mathbf{k}+\mathbf{s}+e_3}\phi(\mathbf{x}_i,\mathbf{y}_c)\end{bmatrix}H_{\mathbf{k}}^{\mathbf{s}}(C)$$

$$= \sum_{||\mathbf{k}||=0}^{\ell}\sum_{||\mathbf{s}||=0}^{p}\begin{bmatrix}b_{\mathbf{k}}^{\mathbf{s}+e_1}\\ b_{\mathbf{k}}^{\mathbf{s}+e_2}\\ b_{\mathbf{k}}^{\mathbf{s}+e_3}\end{bmatrix}H_{\mathbf{k}}^{\mathbf{s}}(C),$$

where $e_1 = \langle 1,0,0\rangle$, $e_2 = \langle 0,1,0\rangle$, and $e_3 = \langle 0,0,1\rangle$.

b. Approximation of the torque

To evaluate the torque on i in the α direction due to cluster C, $\mathcal{M}_i$ is rotated infinitesimally counter-clockwise about the α-axis to get $\mathcal{M}_{i,\alpha}$. Then the torque, $\tau^C_{i,\alpha}$, using the definition in the work of SPD,[17] is approximated as

$$\tau^C_{i,\alpha} = \sum_{||\mathbf{n}||=0}^{p} \mathcal{M}^{\mathbf{n}}_{i,\alpha}\partial^{\mathbf{n}}_i V_{i,C} \tag{A1a}$$

$$\approx \sum_{||\mathbf{k}||=0}^{\ell}\sum_{||\mathbf{n}||=0}^{p}(-1)^{||\mathbf{n}||}\mathcal{M}^{\mathbf{n}}_{i,\alpha}\sum_{||\mathbf{s}||=0}^{p} b^{\mathbf{s}+\mathbf{n}}_{\mathbf{k}}(\mathbf{x}_i,\mathbf{y}_c)H^{\mathbf{s}}_{\mathbf{k}}(C). \tag{A1b}$$

Equation (A1b) is the ℓth order approximation of the torque on particle i in the α-direction due to a well separated cluster C.

2. Formulae for cluster-particle approximations

a. Approximation of the electric field

The exact electric field at $\mathbf{x}_i$ is given as $\mathbf{E}_{i,C} = -\nabla_{\mathbf{x}_i} V_{i,C}$ and it is approximated through the use of Eq. (14c) as

$$\mathbf{E}_{i,C} \approx -\sum_{||\mathbf{k}||=0}^{\ell}\sum_{||\mathbf{s}||=0}^{p} m^{\mathbf{s}}_{\mathbf{k},C}\nabla_{\mathbf{x}_i}(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}}$$

$$= -\sum_{||\mathbf{k}||=0}^{\ell}\sum_{||\mathbf{s}||=0}^{p} m^{\mathbf{s}}_{\mathbf{k},C}\begin{bmatrix} k_1(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}-e_1}\\ k_2(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}-e_2}\\ k_3(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}-e_3}\end{bmatrix}. \tag{A2}$$

b. Approximation of the torque

To evaluate the torque on particle i, in cluster C, in the α direction due to source particles j, $\mathcal{M}_i$ is rotated infinitesimally counter-clockwise about the α-axis to get $\mathcal{M}_{i,\alpha}$. Then the torque, $\tau^C_{i,\alpha}$, using the definition in Ref. 17 and Eq. (14b), is approximated as

$$\tau^C_{i,\alpha} = \sum_{||\mathbf{n}||=0}^{p}\mathcal{M}^{\mathbf{n}}_{i,\alpha}\partial^{\mathbf{n}}_i V_{i,C} \tag{A3a}$$

$$= \sum_{||\mathbf{k}||=0}^{\ell}\sum_{||\mathbf{n}||=0}^{p}(-1)^{||\mathbf{n}||}\mathcal{M}^{\mathbf{n}}_{i,\alpha}\, m^{\mathbf{s}+\mathbf{n}}_{\mathbf{k},C}(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}}. \tag{A3b}$$

Equation (A3b) is the ℓth order approximation of the torque on particle i, in cluster C, in the α-direction due to M source particles j.

3. Error analysis

We provide a condensed analysis of the error in the particle-cluster approximation of the potential in Eq. (7a) for the Coulomb kernel $\phi(\mathbf{x}_i,\mathbf{y}_c) = \frac{1}{|\mathbf{x}_i-\mathbf{y}_c|}$. The approach employed here can be extended to error analysis of the potential energy, force, electric field, and torque. Similar analysis can be done for the cluster-particle approximations as well. We consider the error due to an approximation to order $\ell-1$. The potential from Eqs. (7a) and (7b) can be rewritten as

$$V_{i,C} \approx \sum_{||\mathbf{s}||=0}^{p}\sum_{\mathbf{y}_j\in C}\mathcal{M}^{\mathbf{s}}_j\partial^{\mathbf{s}}_j\sum_{||\mathbf{k}||=0}^{\ell-1}\frac{1}{\mathbf{k}!}\partial^{\mathbf{k}}_{\mathbf{y}}\phi(\mathbf{x}_i,\mathbf{y}_c)(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}} \tag{A4}$$

$$= \sum_{||\mathbf{s}||=0}^{p}\sum_{\mathbf{y}_j\in C}\mathcal{M}^{\mathbf{s}}_j\partial^{\mathbf{s}}_j\sum_{||\mathbf{k}||=0}^{\ell-1}a_{\mathbf{k}}(\mathbf{x}_i,\mathbf{y}_c)(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}}, \tag{A5}$$

where from Eq. (8), we define

$$a_{\mathbf{k}}(\mathbf{x}_i,\mathbf{y}_c) = b^{\mathbf{0}}_{\mathbf{k}}(\mathbf{x}_i,\mathbf{y}_c) = \frac{1}{\mathbf{k}!}\partial^{\mathbf{k}}_{\mathbf{y}}\phi(\mathbf{x}_i,\mathbf{y}_c). \tag{A6}$$

The error, $E_{i,C}$, from approximating $V_{i,C}$ by the $(\ell-1)$th Taylor approximation is given by the absolute value of the sum of the neglected terms,

$$\sum_{||\mathbf{s}||=0}^{p}\sum_{\mathbf{y}_j\in C}\mathcal{M}^{\mathbf{s}}_j\partial^{\mathbf{s}}_j\sum_{||\mathbf{k}||\geq\ell}^{\infty}a_{\mathbf{k}}(\mathbf{x}_i,\mathbf{y}_c)(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}}$$

$$= \sum_{||\mathbf{s}||=0}^{p}\sum_{\mathbf{y}_j\in C}\mathcal{M}^{\mathbf{s}}_j\partial^{\mathbf{s}}_j\sum_{n\geq\ell}^{\infty}A_n(\mathbf{x}_i,\mathbf{y}_c,\mathbf{y}_j), \tag{A7}$$

with

$$A_n(\mathbf{x}_i,\mathbf{y}_c,\mathbf{y}_j) = \sum_{||\mathbf{k}||=n}a_{\mathbf{k}}(\mathbf{x}_i,\mathbf{y}_c)(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{k}}. \tag{A8}$$

A recurrence relation for $A_n(\mathbf{x}_i,\mathbf{y}_c,\mathbf{y}_j)$ is available[44] and is given by

$$nR^2A_n - (2n-1)\alpha A_{n-1} + (n-1)\beta^2A_{n-2} = 0, \tag{A9}$$

where $R = |\mathbf{x}_i-\mathbf{y}_c|$ as defined in Fig. 2, $\alpha = (\mathbf{x}_i-\mathbf{y}_c)\cdot(\mathbf{y}_j-\mathbf{y}_c)$, and $\beta = |\mathbf{y}_j-\mathbf{y}_c|$. Comparison of Eq. (A9) to the recurrence relation for the Legendre polynomials $P_n(x)$ in one-dimension,

$$nP_n(x) - (2n-1)P_{n-1}(x) + (n-1)P_{n-2}(x) = 0, \tag{A10}$$

leads to the closed form formula for An[44] given by

$$A_n = \frac{1}{R}\left(\frac{\beta}{R}\right)^n P_n\left(\frac{\alpha}{\beta R}\right). \tag{A11}$$

Using the property that $|P_n(x)| \leq 1$ for $|x| \leq 1$ and the fact that

$$\left|\frac{\alpha}{\beta R}\right| = \left|\frac{(\mathbf{x}_i-\mathbf{y}_c)\cdot(\mathbf{y}_j-\mathbf{y}_c)}{|\mathbf{x}_i-\mathbf{y}_c|\,|\mathbf{y}_j-\mathbf{y}_c|}\right| \leq 1, \tag{A12}$$

A_n can be bounded as

$$\left|A_n(\mathbf{x}_i,\mathbf{y}_c,\mathbf{y}_j)\right| \leq \frac{1}{R}\left(\frac{|\mathbf{y}_j-\mathbf{y}_c|}{R}\right)^n \leq \frac{\theta^n}{R}, \tag{A13}$$

where we have use the fact that the Taylor approximation is only employed when $\frac{|\mathbf{y}_j-\mathbf{y}_c|}{R} \leq \theta$, the multipole acceptability criterion, and $0 < \theta < 1$. By noting that

$$\partial_j^{s_i} A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j) = \sum_{||\mathbf{k}||=n} k_i a_{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_c)(\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}-s_i\mathbf{e}_i}$$
$$= (\mathbf{y}_j - \mathbf{y}_c)^{-s_i\mathbf{e}_i} \sum_{||\mathbf{k}||=n} k_i a_{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_c)(\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}}, \tag{A14}$$

the error can be bounded as

$$\begin{aligned} E_{i,C} &= \left| \sum_{||\mathbf{s}||=0}^{p} \sum_{\mathbf{y}_j \in C} \mathcal{M}_j^{\mathbf{s}} \partial_j^{\mathbf{s}} \sum_{n\geq\ell}^{\infty} A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j) \right| \\ &\leq \sum_{||\mathbf{s}||=0}^{p} \sum_{\mathbf{y}_j \in C} \left| \mathcal{M}_j^{\mathbf{s}} \right| \sum_{n\geq\ell}^{\infty} \left| \partial_j^{\mathbf{s}} A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j) \right| \\ &\leq \sum_{||\mathbf{s}||=0}^{p} \sum_{\mathbf{y}_j \in C} \left| \mathcal{M}_j^{\mathbf{s}} \right| \left| \mathbf{y}_j - \mathbf{y}_c \right|^{-s} \sum_{n\geq\ell}^{\infty} n \left| A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j) \right| \\ &\leq K \sum_{n\geq\ell}^{\infty} n \cdot \theta^n, \end{aligned} \tag{A15}$$

with

$$K = \max_{C,\mathbf{s}} \left\{ \sum_{||\mathbf{s}||=0}^{p} \sum_{\mathbf{y}_j \in C} \left| \mathcal{M}_j^{\mathbf{s}} \right| \left| \mathbf{y}_j - \mathbf{y}_c \right|^{-s} \frac{1}{R} \right\}.$$

Observe that

$$\begin{aligned} \sum_{n\geq\ell}^{\infty} n \cdot \theta^n &= \ell\theta^\ell + (\ell+1)\theta^{\ell+1} + (\ell+2)\theta^{\ell+2} + \dots \\ &= \ell\theta^\ell \sum_{n=0}^{\infty} \theta^n + \theta^{\ell+1} \sum_{n=1}^{\infty} n\theta^{n-1} \\ &= \ell\theta^\ell \sum_{n=0}^{\infty} \theta^n + \theta^{\ell+1} \frac{d}{d\theta} \left(\sum_{n=0}^{\infty} \theta^n \right) \\ &= \frac{\ell\theta^\ell}{1-\theta} + \theta^{\ell+1} \frac{d}{d\theta} \left(\frac{1}{1-\theta} \right) \\ &= \frac{\ell\theta^\ell}{1-\theta} + \frac{\theta^{\ell+1}}{(1-\theta)^2}, \end{aligned} \tag{A16}$$

and thus from Eq. (A15), the bound on the error is

$$0 \leq E_{i,C} \leq K \left\{ \frac{\ell\theta^\ell}{1-\theta} + \frac{\theta^{\ell+1}}{(1-\theta)^2} \right\}. \tag{A17}$$

As the order of the Taylor approximation, $\ell \to \infty$, $\lim_{\ell\to\infty} \ell\theta^\ell = 0$, and

$$\lim_{\ell\to\infty} \ell\theta^\ell = \lim_{\ell\to\infty} \frac{\ell}{\theta^{-\ell}} = \lim_{\ell\to\infty} \frac{-1}{\theta^{-2\ell} \ln\theta\, \theta^\ell} = -\lim_{\ell\to\infty} \frac{\theta^\ell}{\ln\theta} = 0. \tag{A18}$$

Thus, as expected, the error of the particle-cluster multipole approximation decays to zero with increasing order of the approximation.

[1] R. C. Remsing, M. D. Baer, G. K. Schenter, C. J. Mundy, and J. D. Weeks, J. Phys. Chem. Lett. **5**, 2767 (2014).
[2] P. Jungwirth and B. Winter, Annu. Rev. Phys. Chem. **59**, 343 (2008).
[3] M. Vazdar, E. Pluharova, P. E. Mason, R. Vacha, and P. Jungwirth, J. Phys. Chem. Lett. **3**, 2087 (2012).
[4] S. D. Fried, L. P. Wang, S. G. Boxer, P. Ren, and V. S. Pande, J. Phys. Chem. B **117**, 16236 (2013).
[5] P. Nerenberg, B. Jo, C. So, A. Tripathy, and T. Head-Gordon, J. Phys. Chem. B **116**, 4524 (2012).
[6] P. S. Nerenberg and T. Head-Gordon, J. Chem. Theory Comput. **7**, 1220 (2011).
[7] M. E. Johnson, C. Malardier-Jugroot, R. Murarka, and T. Head-Gordon, J. Phys. Chem. B **113**, 4082 (2009).
[8] A. Albaugh, H. A. Boateng, R. T. Bradshaw, O. N. Demerdash, J. Dziedzic, Y. Mao, D. T. Margul, J. Swails, A. Zeng, D. A. Case, P. Eastman, L.-P. Wang, J. W. Essex, M. Head-Gordon, V. S. Pande, J. W. Ponder, Y. Shao, C. K. Skylaris, I. T. Todorov, M. E. Tuckerman, and T. Head-Gordon, J. Phys. Chem. B **120**, 9811 (2016).
[9] O. Demerdash, E.-H. Yap, and T. Head-Gordon, Annu. Rev. Phys. Chem. **65**, 149 (2014).
[10] J. W. Ponder, C. Wu, P. Ren, V. S. Pande, J. D. Chodera, M. J. Schnieders, I. Haque, D. L. Mobley, D. S. Lambrecht, R. A. DiStasio, Jr., M. Head-Gordon, G. N. I. Clark, M. E. Johnson, and T. Head-Gordon, J. Phys. Chem. B **114**, 2549 (2010).
[11] G. A. Cisneros, M. Karttunen, P. Ren, and C. Sagui, Chem. Rev. **114**, 779 (2014).
[12] F. N. Keutsch and R. J. Saykally, Proc. Natl. Acad. Sci. U. S. A. **98**, 10533 (2001).
[13] P. Ren, C. Wu, and J. W. Ponder, J. Chem. Theory Comput. **7**, 3143 (2011).
[14] J. Kong and J.-M. Yan, Int. J. Quantum Chem. **46**, 239 (1993).
[15] S. Y. Liem and P. L. A. Popelier, J. Chem. Theory Comput. **4**, 353 (2008).
[16] M. S. Shaik, M. Devereux, and P. L. A. Popelier, Mol. Phys. **106**, 1495 (2008).
[17] C. Sagui, L. Pedersen, and T. Darden, J. Chem. Phys. **120**, 73 (2004).
[18] A. Toukmaji, C. Sagui, J. Board, and T. Darden, J. Chem. Phys. **113**, 10913 (2000).
[19] T. M. Nymand and P. Linse, J. Chem. Phys. **112**, 6152 (2000).
[20] W. Smith, CCP5 Newsletter **4**, 13 (1982).
[21] A. Aguado and P. A. Madden, J. Chem. Phys. **119**, 7471 (2003).
[22] A. C. Simmonett, F. C. Pickard IV, H. F. Schaefer III, and B. R. Brooks, J. Chem. Phys. **140**, 184101 (2014).
[23] H. A. Boateng and I. T. Todorov, J. Chem. Phys. **134**, 034117 (2015).
[24] D. Lin, J. Chem. Phys. **143**, 114115 (2015).
[25] C. J. Fennell and J. D. Gezelter, J. Chem. Phys. **124**, 234104 (2006).
[26] U. Essman, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen, J. Chem. Phys. **103**, 8577 (1995).
[27] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (McGraw-Hill, 1981).
[28] R. D. Skeel, I. Tezcan, and D. J. Hardy, J. Comput. Chem. **23**, 673 (2002).
[29] D. J. Hardy, Z. Wu, J. C. Phillips, J. E. Stone, R. D. Skeel, and K. Schulten, J. Chem. Theory Comput. **11**, 766 (2015).
[30] A. Brandt, Math. Comput. **31**, 333 (1977).
[31] A. Brandt and A. A. Lubrecht, J. Comput. Phys. **90**, 348 (1990).
[32] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed. (SIAM, Philadelphia, 2000).
[33] L. Greengard and V. Rokhlin, J. Comput. Phys. **73**, 325 (1987).
[34] H.-Q. Ding, N. Karasawa, and W. A. Goddard III, J. Chem. Phys. **97**, 4309 (1992).
[35] J. Barnes and P. Hut, Nature **324**, 446 (1986).
[36] Z.-H. Duan and R. Krasny, J. Chem. Phys. **113**, 3492 (2000).
[37] Z.-H. Duan and R. Krasny, J. Comput. Chem. **22**, 184 (2001).
[38] H. A. Boateng and R. Krasny, J. Comput. Chem. **34**, 2159 (2013).
[39] J. P. Coles and M. Masella, J. Chem. Phys. **142**, 024109 (2015).
[40] H. Cheng, L. Greengard, and V. Rokhlin, J. Comput. Phys. **155**, 468 (1999).
[41] K. E. Schmidt and M. A. Lee, J. Stat. Phys. **89**, 411 (1997).
[42] T. J. Giese, M. T. Panteva, H. Chen, and D. M. York, J. Chem. Theory Comput. **11**, 436 (2015).
[43] J. K. Salmon and M. S. Warren, J. Comput. Phys. **111**, 136 (1994).
[44] K. Lindsay and R. Krasny, J. Comput. Phys. **172**, 879 (2001).
[45] R. Krasny and L. Wang, SIAM J. Sci. Comput. **33**, 2341 (2011).
[46] Q. Deng and T. A. Driscoll, SIAM J. Sci. Comput. **34**, A1126 (2012).
[47] J. Chen, L. Wang, and M. Anitescu, SIAM J. Sci. Comput. **36**, A289 (2014).
[48] H. A. Boateng, "Cartesian treecode for multipolar electrostatic interactions," https://github.com/haboateng/Cartesian-treecodes-for-multipolar-interaction (2017).

[49]H. A. Boateng (2017). "First release of treecode algorithms for multipolar interactions," Zenodo. https://doi.org/10.5281/zenodo.808909.
[50]J. K. Salmon, M. S. Warren, and G. S. Winckelmans, Int. J. Suppl. Appl. **8**, 129 (1994).
[51]H. Feng, A. Barua, S. Li, and X. Li, Commun. Comput. Phys. **15**, 365 (2014).
[52]J. Dubinski, New Astron. **1**, 133 (1996).
[53]J. W. Ponder, "TINKER: Software tools for molecular design, 6.3," Washington University School of Medicine, Saint Louis, MO, 2014.
[54]I. T. Todorov, W. Smith, K. Trachenko, and M. T. Dove, J. Mater. Chem. **16**, 1911 (2006).
[55]D. A. Case, T. E. Cheatham III, T. Darden, H. Gohlke, R. Luo, K. M. Merz, Jr., A. Onufriev, C. Simmerling, B. Wang, and R. J. Woods, J. Comput. Chem. **26**, 1668 (2005).