A treecode algorithm based on tricubic interpolation

Henry A. Boateng[a], Svetlana Tlupova[b]

[a] *Department of Mathematics, San Francisco State University, 1600 Holloway Avenue, San Francisco, 94132, CA, USA*
[b] *Department of Mathematics, Farmingdale State College, 2350 Broadhollow Road, Farmingdale, 11735, NY, USA*

Abstract

Treecode algorithms efficiently approximate N-body interactions in $O(N)$ or $O(N\log N)$. In order to treat general 3D kernels, recent developments employ polynomial interpolation to approximate the kernels. The polynomials are a tensor product of 1-dimensional polynomials. Here, we develop an $O(N\log N)$ tricubic interpolation based treecode method for 3D kernels. The tricubic interpolation is inherently three-dimensional and as such does not employ a tensor product. The form allows for easy evaluation of the derivatives of the kernel, required in dynamical simulations, which is not the case for the tensor product approach. We develop both a particle-cluster and cluster-particle variants and present results for the Coulomb, screened Coulomb and the real space Ewald kernels. We also present results of an MD simulation of a Lennard-Jones liquid using the tricubic treecode.

Keywords: Fast summation, N-body interactions, Treecode, Tricubic interpolation
2000 MSC: 70-08, 70-10, 76-04, 85-04, 65D99

1. Introduction

This work concerns the evaluation of sums of the form

$$\phi(\mathbf{x}_m) = \sum_{n=1}^{N} \mathcal{K}(\mathbf{x}_m, \mathbf{y}_n) f_n, \qquad m = 1, ..., M, \tag{1}$$

where $\{\mathbf{x}_m\}, m = 1, ..., M$ is a set of target particles, $\{\mathbf{y}_n\}, n = 1, ..., N$ is a set of source particles with weights $\{f_n\}$, and $\phi(\mathbf{x})$ is a potential (or velocity). The kernel $\mathcal{K}(\mathbf{x}, \mathbf{y})$ represents the pairwise interaction between a target particle $\mathbf{x}$ and a source particle $\mathbf{y}$. Sums of this type arise in numerous applications in physics, chemistry, fluid dynamics, etc., where the kernel may be a scalar or a tensor and the weights are scalars or vectors. In applications where the target and source particles are the same, the $n = m$ term is excluded from the sum.

Direct computation of the sum in equation (1) requires $O(MN)$, or $O(N^2)$ operations for $M = N$, which is a significant computational bottleneck when N is large. A standard approach to reducing the computational cost is to partition the sum into a near-field interaction and a far-field interaction. The near-field interactions are computed exactly and the far-field interactions are approximated. One method for approximating the far-field is the particle-mesh method [23, 13, 3]. Particle-mesh methods interpolate the particles onto a uniform grid and employ an FFT to compute the sum and thus reduce the $O(MN)$ computational cost to $O(M \log N)$. An alternative approach to approximating the far-field is the tree-based methods [5, 18]. Tree-based methods restructure the target and/or source particles into a hierarchical tree of clusters of particles. The computational cost is reduced by replacing far-field particle-particle interactions by particle-cluster or cluster-cluster interactions.

The present work is concerned with the latter category of methods. Several variants of treecode algorithms have been developed to evaluate the sum in equation (1) in $O(N)$ [19, 18] or $O(N\log N)$ [5]. The early versions of treecode algorithms [5, 19, 18, 10, 17, 11, 12, 30, 38, 28, 7, 8, 39] were developed for specific kernels and employed analytic expansions specific to each kernel. More recent approaches are able to treat general kernel functions, for example, the kernel-independent FMM which uses equivalent particle distribution determined by solving linear systems [43, 44], and the black-box FMM which uses polynomial interpolation at Chebyshev points combined with SVD compression [14]. Recently, two interpolation based treecode algorithms have been developed, one based on barycentric Lagrange interpolation at Chebyshev points, which is kernel independent [40], and the other based on barycentric Hermite interpolation [24]. Both the barycentric Lagrange and barycentric Hermite interpolation treecodes employ a tensor product of three single variable polynomials to interpolate the 3D kernels.

In this paper, a treecode algorithm is presented based on tricubic interpolation of the kernel for pairwise interactions. Tricubic interpolation broadly refers to the method of local approximation of a function defined on a regular grid in three dimensions. The general approach [26] is to represent the function within a unit cube by a polynomial in the three spatial variables, with the unknown coefficients determined by requiring the function to have a given value or a given derivative at certain points, usually the corners of the unit cube. The method is equivalent to a sequential application of three one-dimensional cubic interpolants [26], but its intrinsically three-dimensional formulation has better computational efficiency especially when the interpolation is used at multiple points inside each cube element. It is also advantageous when the derivatives of the interpolated function are needed, since they can be found easily by analytical, rather than numerical, differentiation of the tricubic polynomial.

In the treecode algorithm, the particles are recursively divided into a hierarchical tree of clusters, and the pairwise interactions are replaced with particle-cluster interactions. An approximation for a far-field particle-cluster interaction is derived based on the tricubic interpolation of the kernel using the values of the kernel function and its derivatives at the eight corners of the cluster. The tricubic interpolation approach is chosen for its lower computational complexity and ease of evaluating exact derivatives of the interpolated kernel when compared to triple one-dimensional cubic interpolation. In addition, the interpolant [26] implemented here has global $\mathcal{C}^1$ continuity in approximating the kernel. It however requires up to third order derivatives of the kernel. We present both a particle-cluster and a cluster-particle variants of the treecode algorithm. In a follow up paper, we investigate the effect of global smoothness on the accuracy of treecodes.

The paper is organized as follows. In Section 2, we review the general approach of tricubic interpolation. In Section 3, we derive an approximation for a particle-cluster interaction based on tricubic interpolation. We also use the simplicity of finding analytical derivatives of the interpolated kernel to derive an approximation for the derivatives of ϕ. We analyze the error in a far-field approximation with the tricubic interpolant, and present the full particle-cluster treecode algorithm. Section 4 develops the cluster-particle variant suitable for simulations with disjoint sources and targets where the targets outnumber the sources [6]. Section 5 presents the treecode performance in terms of accuracy and CPU time for several kernels, as well as an MD simulation. Conclusions and future work are discussed in Section 6.

2. Tricubic interpolation

In tricubic interpolation, a function f is represented locally as a piecewise cubic polynomial of the form

$$f(x,y,z) = \sum_{i,j,k=0}^{3} a_{ijk} x^i y^j z^k, \tag{2}$$

within a mesh element that is a unit cube $0 \le x, y, z \le 1$, and the 64 coefficients a_{ijk} are determined from given data. An algorithm to evaluate these coefficients was presented in [26], by using the values of

$$S := \left\{ f,\ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z},\ \frac{\partial^2 f}{\partial x \partial y}, \frac{\partial f^2}{\partial x \partial z}, \frac{\partial f^2}{\partial y \partial z},\ \frac{\partial^3 f}{\partial x \partial y \partial z} \right\} \tag{3}$$

at the eight corners $p_1, ..., p_8$ of the unit cube, see Figure 1. First, the unknown coefficients a_{ijk} are ordered into a vector $\boldsymbol{\alpha}$ by defining

$$\alpha_{1+i+4j+16k} = a_{ijk}, \qquad i,j,k = 0,1,2,3. \tag{4}$$

Similarly, the function and its derivatives are stacked into a vector $\mathbf{b}$ as follows,

$$b_i = \begin{cases} f(p_i), & 1 \le i \le 8, \\ \dfrac{\partial f}{\partial x}(p_{i-8}), & 9 \le i \le 16, \\ \dfrac{\partial f}{\partial y}(p_{i-16}), & 17 \le i \le 24, \\ \dfrac{\partial f}{\partial z}(p_{i-24}), & 25 \le i \le 32, \\ \dfrac{\partial^2 f}{\partial x \partial y}(p_{i-32}), & 33 \le i \le 40, \\ \dfrac{\partial^2 f}{\partial x \partial z}(p_{i-40}), & 41 \le i \le 48, \\ \dfrac{\partial^2 f}{\partial y \partial z}(p_{i-48}), & 49 \le i \le 56, \\ \dfrac{\partial^3 f}{\partial x \partial y \partial z}(p_{i-56}), & 57 \le i \le 64. \end{cases} \tag{5}$$

When the analytical expressions for the derivatives of the function f are unavailable for (5), various techniques such as finite differences can be used. Evaluating the polynomial in (2) and its derivatives at the eight corners of the cube then leads to a sparse linear system for the unknown coefficients

$$B\boldsymbol{\alpha} = \mathbf{b}, \tag{6}$$

where B is a 64×64 invertible matrix with integer elements which can be solved explicitly as

$$\boldsymbol{\alpha} = B^{-1}\mathbf{b}. \tag{7}$$

The matrix B^{-1} is sparse and is computed exactly without numerical error [26]. It has exactly 1000 non-zero elements and a condition number $\kappa_2(B^{-1}) = 1.345 \times 10^4$. In Section 3 we show that in the treecode algorithm, only multiplication by the transpose of the inverse $(B^{-1})^T$ is needed, and as this matrix is sparse, the multiplication can be done in-line.

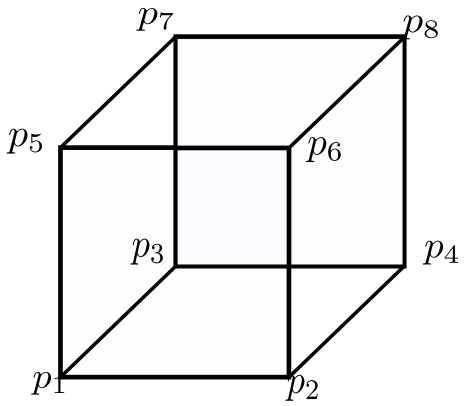

Figure 1: A schematic of the unit cube for the tricubic interpolant.

The representation (2) has several advantages as an interpolant. It was shown to be the minimum order necessary to maintain global $\mathcal{C}^1$ continuity in the approximated function [26]. Furthermore, the derivatives of the function can be computed analytically, in contrast with three one-dimensional cubic interpolants, where the derivatives are not easily accessible and finite differences or other methods are needed to recover the derivatives.

We can define a vector $\boldsymbol{\mu}$ where

$$\mu_{1+i+4j+16k} = x^i y^j z^k, \qquad i,j,k = 0,1,2,3, \tag{8}$$

and use the definition of $\boldsymbol{\alpha}$ in equation (4) to write equation (2) as an inner-product

$$f(x,y,z) = \boldsymbol{\alpha}^T \boldsymbol{\mu}. \tag{9}$$

Here and in the rest of the paper, a superscript T will denote a transpose.

2.1. Rectangular meshes of arbitrary size

The representation in (2) can be modified for a rectangular mesh element of arbitrary size and location by shifting and scaling each variable accordingly,

$$f(x,y,z) = \sum_{i,j,k=0}^{3} a_{ijk} \left(\frac{x-x_0}{\Delta x}\right)^i \left(\frac{y-y_0}{\Delta y}\right)^j \left(\frac{z-z_0}{\Delta z}\right)^k, \tag{10}$$

where $\Delta x, \Delta y, \Delta z$ are the lengths of the element in the three dimensions, and (x_0, y_0, z_0) is the lower left corner of the element. Note that in this case, the derivatives in (5) must be appropriately scaled. For example, differentiating (10) in the x variable, we get,

$$\frac{\partial f}{\partial x}(x,y,z) = \sum_{i=1,j,k=0}^{3} \frac{i}{\Delta x} a_{ijk} \left(\frac{x-x_0}{\Delta x}\right)^{i-1} \left(\frac{y-y_0}{\Delta y}\right)^j \left(\frac{z-z_0}{\Delta z}\right)^k. \tag{11}$$

Evaluating the function in (10) and the three derivatives at the lower left corner $\mathbf{x}_0$, we get

$$f|_{\mathbf{x}_0} = a_{000}, \qquad \frac{\partial f}{\partial x}\Big|_{\mathbf{x}_0} = \frac{a_{100}}{\Delta x}, \qquad \frac{\partial f}{\partial y}\Big|_{\mathbf{x}_0} = \frac{a_{010}}{\Delta y}, \qquad \frac{\partial f}{\partial z}\Big|_{\mathbf{x}_0} = \frac{a_{001}}{\Delta z}. \tag{12}$$

Consequently, the evaluation of coefficients still follows (7) but the right hand side in (5) is evaluated as

$$S^* : \Big\{ f,\ \Delta x \frac{\partial f}{\partial x}, \Delta y \frac{\partial f}{\partial y}, \Delta z \frac{\partial f}{\partial z},\ \Delta x \Delta y \frac{\partial^2 f}{\partial x \partial y}, \Delta x \Delta z \frac{\partial f^2}{\partial x \partial z}, \Delta y \Delta z \frac{\partial f^2}{\partial y \partial z},\ \Delta x \Delta y \Delta z \frac{\partial^3 f}{\partial x \partial y \partial z} \Big\}, \tag{13}$$

while the B matrix remains the same.

3. The particle-cluster treecode

We now present the main components of a treecode algorithm based on the tricubic interpolation. First, all source particles are divided into a hierarchy of clusters, and a target particle interacts with clusters of source particles, rather than individual sources, as described below.

3.1. A particle-cluster interaction

Consider a target particle $\mathbf{x}_m = (x_m, y_m, z_m)$ interacting with source particles $\mathbf{y}_n = (x_n, y_n, z_n)$ in a source cluster C, as shown in Figure 2. The cluster has a radius r, and the particle-cluster distance is $R = |\mathbf{x}_m - \mathbf{y}_c|$, where $\mathbf{y}_c$ is the cluster center.

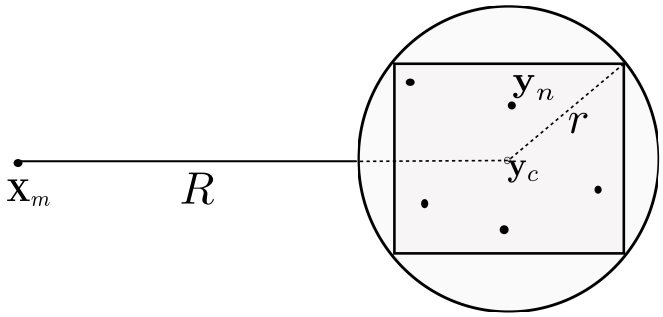

Figure 2: Particle-cluster interaction. The target particle is at position $\mathbf{x}_m$ and the source particles are at positions $\mathbf{y}_n$ in cluster C. Cluster C has center $\mathbf{y}_c$ and radius r. The particle-cluster distance is $R = |\mathbf{x}_m - \mathbf{y}_c|$.

The component of the sum (1) for this interaction is written as

$$\phi(\mathbf{x}_m, C) = \sum_{\mathbf{y}_n \in C} \mathcal{K}(\mathbf{x}_m, \mathbf{y}_n) f_n. \tag{14}$$

If the particle and the cluster are far enough apart (that is $\dfrac{r}{R} \leq \theta$ [5], where $0 \leq \theta < 1$), the sum in (14) is approximated in the following way. First, the cluster is shifted and scaled to the unit cube $[0,1]^3$,

$$x = \frac{x_n - x_{\min}}{\Delta x}, \quad y = \frac{y_n - y_{\min}}{\Delta y}, \quad z = \frac{z_n - z_{\min}}{\Delta z}, \tag{15}$$

where $\mathbf{y}_{\min} = (x_{\min}, y_{\min}, z_{\min})$ are the minimum x, y, z coordinates of the cluster C, and $\Delta\mathbf{y} = (\Delta x, \Delta y, \Delta z)$ is the size of the cluster box. The target point is shifted as well $\mathbf{x}_m \to \mathbf{x}_m - \mathbf{y}_{\min}$. Then the kernel function $\mathcal{K}(\mathbf{x}_m, \mathbf{y})$ is interpolated in the second (source) variable using the tricubic formula (2),

$$\phi(\mathbf{x}_m, C) = \sum_{\mathbf{y}_n \in C} \mathcal{K}(\mathbf{x}_m, \mathbf{y}_n) f_n \approx \sum_{\mathbf{y}_n \in C} \sum_{i,j,k=0}^{3} a_{ijk} x^i y^j z^k f_n. \tag{16}$$

Since the tricubic coefficients a_{ijk} do not depend on the individual source particles in the cluster, we switch the order of summation in (16), and use the definition in (8), to obtain the far-field approximation,

$$\begin{aligned} \phi(\mathbf{x}_m, C) &\approx \sum_{i,j,k=0}^{3} a_{ijk} \sum_{\mathbf{y}_n \in C} x^i y^j z^k f_n, \\ &= \boldsymbol{\alpha}_m^T \boldsymbol{\mu}^C, \end{aligned} \tag{17}$$

where

$$\mu^C_{1+i+4j+16k} = \sum_{\mathbf{y}_n \in C} \mu_{1+i+4j+16k} f_n \tag{18}$$

are the monomials of the cluster C. The significance of approximation (17) is first, the coefficients $\boldsymbol{\alpha}_m$ depend only on the target particle $\mathbf{x}_m$ and the cluster corners, and second, the cluster monomials $\boldsymbol{\mu}^C$ are independent of the target particle. We can achieve further time saving by using (7) to rewrite (17) as

$$\phi(\mathbf{x}_m, C) \approx \boldsymbol{\alpha}_m^T \boldsymbol{\mu}^C = (B^{-1}\mathbf{b}_m)^T \boldsymbol{\mu}^C = \mathbf{b}_m^T (B^{-1})^T \boldsymbol{\mu}^C = \mathbf{b}_m^T \mathbf{M}^C, \tag{19}$$

where

$$\mathbf{M}^C = (B^{-1})^T \boldsymbol{\mu}^C, \tag{20}$$

are the modified monomials of the cluster C which are also independent of the target particle $\mathbf{x}_m$. These modified monomials are therefore precomputed and stored for each cluster using (20), since the 64×64 matrix $(B^{-1})^T$ is known explicitly. Furthermore, the matrix multiplication in (20) can be done in-line since the matrix is sparse. These monomials $\mathbf{M}^C$ can then be reused for different targets. Equation (19) defines the far-field tricubic approximation for the potential at the target position $\mathbf{x}_m$ due to all the source particles $\mathbf{y}_n$ in cluster C. In summary, the particle-cluster approximation (19) is performed in two steps: first, the 64 elements of $\mathbf{b}_m$ are computed using (5), scaling the derivatives as in (13), and second, the dot product of $\mathbf{b}_m$ and $\mathbf{M}^C$ is computed in (19).

The cost of evaluating the particle-cluster interaction using (19) can be estimated as follows. The first step of computing the vector $\mathbf{b}_m$ is roughly 64 function evaluations (the kernel and its derivatives at 8 corner points), and the exact time can vary depending on the complexity of the kernel function and whether finite differences are used for the derivatives. Once this step is completed, assembling the velocity through the dot product in (19) is another 64 multiplications. The cost of direct summation in (14) is $O(N_c)$, where N_c is the number of particles in the cluster. The approximation process is more efficient than direct summation since the number of particles in each cluster $N_c \gg 64$. Summing over all clusters brings the total estimate to $O(64\log(N))$, or simply $O(\log(N))$, for each target particle, and the overall algorithm for N targets to $O(N\log(N))$, consistent with other treecodes.

3.2. Approximating the electric field

The use of tricubic interpolation to obtain the far-field approximation (19) for the potential in (14) makes it a simple task to compute derivatives (or the electric field) as follows. The electric field at target position $\mathbf{x}_m$ due to the source cluster C is given by

$$\mathbf{E}_m = -\nabla_{\mathbf{x}_m} \phi(\mathbf{x}_m, C) = \nabla_{\mathbf{y}_n} \phi(\mathbf{x}_m, C), \tag{21}$$

since the kernel $\mathcal{K}$ is a function of $|\mathbf{x} - \mathbf{y}|$. From the tricubic approximation of the potential given in (19), the field is approximated as

$$\mathbf{E}_m = \nabla_{\mathbf{y}_n} \phi(\mathbf{x}_m, C) \approx \nabla_{\mathbf{y}_n} \mathbf{b}_m^T \mathbf{M}^C = \mathbf{b}_m^T \nabla_{\mathbf{y}_n} \mathbf{M}^C, \tag{22}$$

since $\mathbf{b}_m$ is independent of $\mathbf{y}_n$. The derivative of the modified moments with respect to x_n, the first coordinate of the source variable $\mathbf{y}_n = (x_n, y_n, z_n)$, is

$$\frac{\partial \mathbf{M}^C}{\partial x_n} = \frac{\partial}{\partial x_n}\left((B^{-1})^T \boldsymbol{\mu}^C\right) = (B^{-1})^T \frac{\partial}{\partial x_n}\left(\boldsymbol{\mu}^C\right). \tag{23}$$

The derivative of $\boldsymbol{\mu}^C$ is computed element-wise as

$$\begin{aligned}
\frac{\partial}{\partial x_n}\left(\mu^C_{1+i+4j+16k}\right) &= \frac{\partial}{\partial x_n}\left(\sum_{\mathbf{y}_n\in C} x^i y^j z^k f_n\right) = \sum_{\mathbf{y}_n\in C} y^j z^k f_n \frac{\partial}{\partial x_n}\left(\frac{x_n - x_{\min}}{\Delta x}\right)^i, \\
&= \frac{i}{\Delta x}\sum_{\mathbf{y}_n\in C} x^{i-1} y^j z^k f_n, \\
&= \frac{i}{\Delta x}\mu^C_{1+(i-1)+4j+16k} \\
&= \mu^{C,i}. \qquad (24)
\end{aligned}$$

Then from (23),

$$\frac{\partial \mathbf{M}^C}{\partial x_n} = (B^{-1})^T \frac{\partial}{\partial x_n}\left(\boldsymbol{\mu}^C\right) = (B^{-1})^T \boldsymbol{\mu}^{C,i} = \mathbf{M}^{C,i}. \qquad (25)$$

Similarly,

$$\frac{\partial}{\partial y_n}\left(\mu^C_{1+i+4j+16k}\right) = \frac{j}{\Delta y}\mu^C_{1+i+4(j-1)+16k} = \mu^{C,j}, \qquad (26)$$

and

$$\frac{\partial}{\partial z_n}\left(\mu^C_{1+i+4j+16k}\right) = \frac{k}{\Delta z}\mu^C_{1+i+4j+16(k-1)} = \mu^{C,k}. \qquad (27)$$

Hence,

$$\frac{\partial \mathbf{M}^C}{\partial y_n} = (B^{-1})^T \boldsymbol{\mu}^{C,j} = \mathbf{M}^{C,j}, \qquad (28)$$

and

$$\frac{\partial \mathbf{M}^C}{\partial z_n} = (B^{-1})^T \boldsymbol{\mu}^{C,k} = \mathbf{M}^{C,k}. \qquad (29)$$

Then from (22),

$$\begin{aligned}
\mathbf{E}_m &\approx \mathbf{b}_m^T \nabla_{\mathbf{y}_n} \mathbf{M}^C, \\
&= \mathbf{b}_m^T \begin{bmatrix} \dfrac{\partial \mathbf{M}^C}{\partial x_n} \\ \dfrac{\partial \mathbf{M}^C}{\partial y_n} \\ \dfrac{\partial \mathbf{M}^C}{\partial z_n} \end{bmatrix} = \mathbf{b}_m^T \begin{bmatrix} \mathbf{M}^{C,i} \\ \mathbf{M}^{C,j} \\ \mathbf{M}^{C,k} \end{bmatrix}. \qquad (30)
\end{aligned}$$

The monomials $\mathbf{M}^{C,i}$, $\mathbf{M}^{C,j}$ and $\mathbf{M}^{C,k}$ are precomputed for each cluster in the same routine that precomputes $\mathbf{M}^C$ and reused for different targets.

3.3. Error analysis

Here we estimate the error in using tricubic interpolation (16) to evaluate the sum (14) for a fixed target particle $\mathbf{x}_m$ and a source cluster C. Without loss of generality, we assume the cluster C contains the source particles $\mathbf{x}_n$, $n = 1, ..., N$. Let (x, y, z) be the shifted and scaled coordinates

of the source $\mathbf{y}_n$, defined in (15). Since $|x| \le 1, |y| \le 1, |z| \le 1$, we define the largest error as the difference between approximations using the fourth-degree interpolant and the tricubic. We write the error as follows:

$$\mathcal{E}(\mathbf{x}_m, C) = \sum_{n=1}^{N} f_n \left[\sum_{i,j,k=0}^{4} a_{ijk} x^i y^j z^k - \sum_{i,j,k=0}^{3} a_{ijk} x^i y^j z^k \right]. \tag{31}$$

After cancellations, (31) becomes

$$\mathcal{E}(\mathbf{x}_m, C) = \sum_{n=1}^{N} f_n \left[\sum_{i=0}^{4}\sum_{k=0}^{4} a_{i4k} x^i y^4 z^k + \sum_{j=0}^{3}\sum_{k=0}^{4} a_{4jk} x^4 y^j z^k + \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij4} x^i y^j z^4 \right]. \tag{32}$$

Let $F = \max\limits_{1\le n\le N} |f_n|$ and $A = \max\limits_{0\le i,j,k\le 4} |a_{ijk}|$. Then

$$|\mathcal{E}(\mathbf{x}_m, C)| \le AF \sum_{n=1}^{N} \left[\sum_{i=0}^{4}\sum_{k=0}^{4} |x^i y^4 z^k| + \sum_{j=0}^{3}\sum_{k=0}^{4} |x^4 y^j z^k| + \sum_{i=0}^{3}\sum_{j=0}^{3} |x^i y^j z^4| \right]. \tag{33}$$

Since $|x| \le 1, |y| \le 1, |z| \le 1$, and letting $\beta = AF$, we get

$$\begin{aligned}
|\mathcal{E}(\mathbf{x}_m, C)| &\le \beta \sum_{n=1}^{N} \left[\sum_{i=0}^{4}\sum_{k=0}^{4} |y^4| + \sum_{j=0}^{3}\sum_{k=0}^{4} |x^4| + \sum_{i=0}^{3}\sum_{j=0}^{3} |z^4| \right] \\
&= \beta \sum_{n=1}^{N} \left[25|y^4| + 20|x^4| + 16|z^4| \right] \\
&= \beta \sum_{n=1}^{N} \left[25 \left(\frac{y_n - y_{\min}}{\Delta y} \right)^4 + 20 \left(\frac{x_n - x_{\min}}{\Delta x} \right)^4 + 16 \left(\frac{z_n - z_{\min}}{\Delta z} \right)^4 \right] \\
&\le \beta \sum_{n=1}^{N} \left[25 \left(\frac{l}{\Delta w} \right)^4 + 20 \left(\frac{l}{\Delta w} \right)^4 + 16 \left(\frac{l}{\Delta w} \right)^4 \right] \\
&= 61\beta N \frac{l^4}{(\Delta w)^4},
\end{aligned} \tag{34}$$

where $l = \max\limits_{n} \{|x_n - x_{\min}|, |y_n - y_{\min}|, |z_n - z_{\min}|\} \le \max\{\Delta x, \Delta y, \Delta z\}$ and $\Delta w = \min\{\Delta x, \Delta y, \Delta z\}$. Without loss of generality, let $\Delta x = \max\{\Delta x, \Delta y, \Delta z\}$. Then, for a rectangular parallelepiped cluster, the radius $r = \dfrac{\Delta x}{2}\sqrt{1 + \left(\dfrac{\Delta y}{\Delta x}\right)^2 + \left(\dfrac{\Delta z}{\Delta x}\right)^2}$ and (34) can be written as

$$\begin{aligned}
|\mathcal{E}(\mathbf{x}_m, C)| \le \frac{61\beta N}{(\Delta w)^4} (\Delta x)^4 &= \frac{488\beta N}{(\Delta w)^4} \frac{r^4}{\left(\sqrt{1 + \left(\frac{\Delta y}{\Delta x}\right)^2 + \left(\frac{\Delta z}{\Delta x}\right)^2} \right)^4}, \\
&\le \frac{488\beta N R^4}{(\Delta w)^4} \left(\frac{r}{R} \right)^4 = \gamma\rho \left(\frac{r}{R} \right)^4,
\end{aligned} \tag{35}$$

where $\gamma = \dfrac{488\beta R^4}{\Delta w}$ and $\rho = \dfrac{N}{(\Delta w)^3}$ is proportional to the particle density of the cluster. For a cubic cluster, ρ is exactly the particle density. For each target particle, the treecode algorithm

cycles through the clusters in the tree recursively, and evaluates a particle-cluster interaction using the approximation in (19) only when the particle and the cluster are well-separated, that is, the acceptance criterion, typically called the MAC, is satisfied:

$$\frac{r}{R} \leq \theta, \tag{36}$$

where r is the cluster radius, R is the particle-cluster distance, and θ is a user-specified parameter. If the MAC is not satisfied, the children of the cluster are checked, and if the cluster is a leaf (no children), then the particle-cluster interaction is computed directly by (14). Thus, the error for the tricubic approximation is

$$|\mathcal{E}(\mathbf{x}_m, C)| \leq \gamma\rho\ \theta^4. \tag{37}$$

To provide numerical evidence for the θ dependence of the error given in (37), we compute the error in the Coulomb potential $\phi(\mathbf{x}, \mathbf{y}) = \dfrac{1}{|\mathbf{x}-\mathbf{y}|}$ for a particle-cluster interaction with fixed particle density ρ. The source cluster is a unit cube centered at $(0, 0, 0)$ containing 1000 uniformly distributed points. The target particle is located at $\mathbf{x}_m = (2.0 + dx, 0, 0)$ with $dx = 0 : 0.1 : 8$. The MAC parameter $\theta = \dfrac{r}{R} = \dfrac{\sqrt{3}}{2(2+dx)}$. A plot of the error vs θ is shown in Figure 3. The plot provides graphical evidence of the θ^4 dependence of the error.

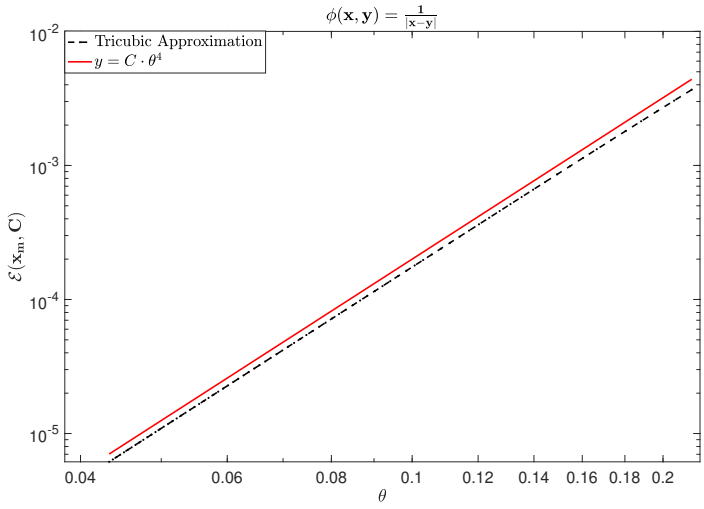

Figure 3: Log-log plot of the error $\mathcal{E}$ vs θ for constant ρ.

Figure 4 is an attempt to provide evidence for the dependence of the error on the particle density. The error is again from a particle-cluster interaction with the Coulomb potential. In this study, the MAC parameter, θ, is kept constant. The cluster is the unit cube centered at $(0, 0, 0)$ and the target particle is fixed at $(2, 0, 0)$. The source points $N \in \{64, 125, 512, 1000, 4096\}$. The points in this case are uniform grid points in the cube. The plot shows an increase in the error with the density. We note that the coefficients a_{ijk} have an effect on the error as well. They are kernel dependent and are related to the rate of decay of the kernel. Unlike the MAC parameter and the particle density, our understanding of the effect of the coefficients is mainly heuristic. In Section 5, we provide numerical results showing the dependence of the error of the full treecode on the MAC parameter and the particle density.

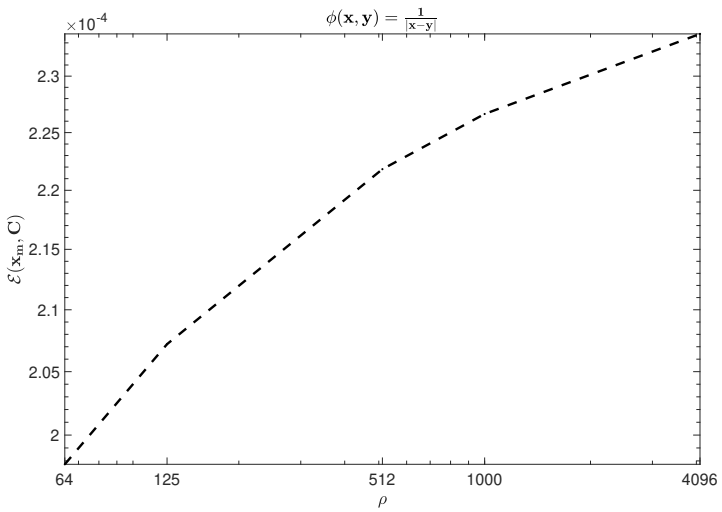

Figure 4: Log-log plot of the error $\mathcal{E}$ vs ρ for constant θ.

3.4. The particle-cluster algorithm

With the particle-cluster approximation established, our treecode algorithm is similar to other treecodes [8, 40]. For completeness, we give an overview of the algorithm with the pseudocode presented in Algorithm 1. First, the particle data (coordinates $\mathbf{x}$ and weights f where weights are either one or three-dimensional) are read from a file. Then, the particles are divided recursively into clusters to generate a tree structure. The root cluster is the smallest rectangular box that encloses all particles. The root is bisected in each coordinate direction to create 8 child clusters. The process is repeated for each child cluster, recursively until a cluster has fewer than N_0 particles, where N_0 is a user-specified leaf-size parameter. For each cluster, the modified monomials $\mathbf{M}^C$, $\mathbf{M}^{C,i}$, $\mathbf{M}^{C,j}$, $\mathbf{M}^{C,k}$, each of length 64, are computed using equations (20), (25), (28) and (29). This concludes the precomputation needed at the start of the algorithm. The algorithm then loops through the target particles. For each target particle, the interaction with all the source particles is done recursively through the clusters. For a given particle-cluster interaction, if the MAC (36) is satisfied, we compute the approximations in (19) and (30), where $\mathbf{b}_m$ is first evaluated and $\mathbf{M}^C$, $\mathbf{M}^{C,i}$, $\mathbf{M}^{C,j}$ and $\mathbf{M}^{C,k}$ are simply looked up from the precomputation. If the MAC is not met, and the cluster is a leaf, the interaction is evaluated directly using (14). Otherwise, all children of the cluster are checked.

4. The cluster-particle treecode

Here we describe an alternative treecode method for computing the sum in (1), based on partitioning the set of target particles $\{\mathbf{x}_m\}$ into an octree and applying a near-field approximation [6]. Figure 5 shows a cluster-particle interaction between targets $\mathbf{x}_m$ in a target cluster C and a source particle $\mathbf{y}_n$. The target cluster and the source particle are well-separated if $r/R \leq \theta$, in which case the algorithm approximates the potential and electric field at the targets by a near-field tricubic interpolation.

Suppose for the cluster C, $\{\mathbf{y}_s\}$ is the "interaction list", that is, the set of source particles well-separated from C. We shift these particles as before $\mathbf{y}_s \to \mathbf{y}_s - \mathbf{x}_{\min}$, where $\mathbf{x}_{\min} = (x_{\min}, y_{\min}, z_{\min})$ are the minimum x, y, z coordinates of the cluster C. The cluster is shifted and scaled to the unit

Algorithm 1 tricubic treecode: particle-cluster

input: particle coordinates $\mathbf{x}_m$, $m = 1, \ldots, N$, f_n, $n = 1, \ldots, N$, parameters θ, N_0
Set $\{\mathbf{y}_n\}_{n=1}^N = \{\mathbf{x}_m\}_{m=1}^M = N$
output: potential ϕ_m, electric field $\mathbf{E}_m$, $m = 1, \ldots, N$
program **main**
  build tree of source particles $\mathbf{y}_n$
  precompute and store $\mathbf{M}^C$, $\mathbf{M}^{C,i}$, $\mathbf{M}^{C,j}$, $\mathbf{M}^{C,k}$ using (20), (25), (28) and (29), for each cluster
  for $m = 1, \ldots, N$, **compute_potential**($\mathbf{x}_m$, root), end for
end program
subroutine **compute_potential**($\mathbf{x}$, C)
  if MAC (36) is satisfied
    compute $\mathbf{b}_m$ using (5)
    compute particle-cluster interaction by approximations (19) and (30)
  else
    if C is a leaf, compute particle-cluster interaction by direct sum (14)
  else
    for each child C' of C, **compute_potential**($\mathbf{x}$, C'), end for
end subroutine

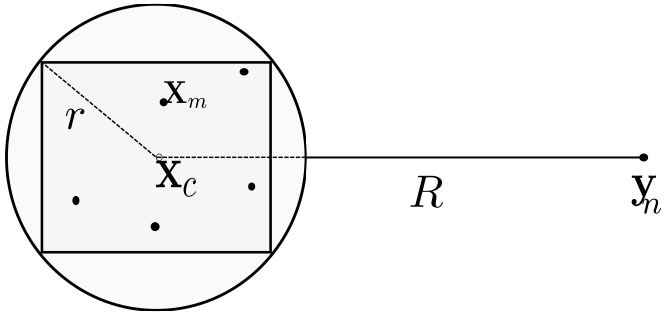

Figure 5: Cluster-particle interaction. The source particle is at position $\mathbf{y}_n$ and the target particles are at positions $\mathbf{x}_m$ in cluster C. Cluster C has center $\mathbf{x}_c$ and radius r. The particle-cluster distance is $R = |\mathbf{y}_n - \mathbf{x}_c|$.

cube similar to (15). Then the kernel is interpolated in the target variable, and the cluster-particle interaction can be evaluated as

$$\begin{aligned}
\phi(\mathbf{x}_m, \mathbf{y}_s) &= \sum_{\{\mathbf{y}_s\}} \mathcal{K}(\mathbf{x}_m, \mathbf{y}_s) f_s, \\
&\approx \sum_{\{\mathbf{y}_s\}} \sum_{i,j,k=0}^{3} a_{ijk} \left(\frac{x_m - x_{\min}}{\Delta x} \right)^i \left(\frac{y_m - y_{\min}}{\Delta y} \right)^j \left(\frac{z_m - z_{\min}}{\Delta z} \right)^k f_s, \\
&= \sum_{\{\mathbf{y}_s\}} \sum_{i,j,k=0}^{3} a_{ijk}\, x^i y^j z^k f_s. \qquad (38)
\end{aligned}$$

The tricubic coefficients a_{ijk} depend on the corners of the target cluster and the source particles in the interaction list, but do not depend on the individual particles in the target cluster C. As such,

we can re-arrange the summation and use equations (4) and (8) to rewrite the approximation as,

$$\phi(\mathbf{x}_m, \mathbf{y}_s) \approx \sum_{i,j,k=0}^{3} x^i y^j z^k \sum_{\{\mathbf{y}_s\}} a_{ijk}\, f_s = \sum_{i,j,k=0}^{3} x^i y^j z^k \sum_{\{\mathbf{y}_s\}} \alpha_{c,1+i+4j+16k}$$
$$= \boldsymbol{\alpha}_c^T \boldsymbol{\mu}_m, \tag{39}$$

where from equation (7),

$$\alpha_{c,1+i+4j+16k} = \sum_{\{\mathbf{y}_s\}} a_{ijk}\, f_s = \sum_{\{\mathbf{y}_s\}} (B^{-1}\mathbf{b}_s)_{ijk}\, f_s = \Big(B^{-1} \sum_{\{\mathbf{y}_s\}} \mathbf{b}_s\, f_s\Big)_{ijk}, \tag{40}$$

and

$$\boldsymbol{\alpha}_c = B^{-1} \sum_{\{\mathbf{y}_s\}} \mathbf{b}_s f_s, \tag{41}$$

are the modified tricubic coefficients of the cluster C. Equation (39) defines the near-field tricubic approximation for the potential at the target position $\mathbf{x}_m \in C$ due to the source particles in the interaction list of C, $\{\mathbf{y}_s\}$.

We see that the particle-cluster approximation in (17) and the cluster-particle approximation in (39) are both of the polynomial form given in (9). For the particle-cluster approximation, the coefficient vector $\boldsymbol{\alpha}_m$ and the monomial vector $\boldsymbol{\mu}^C$ depend on the target and source particles respectively. This is reversed in cluster-particle where the coefficient vector $\boldsymbol{\alpha}_c$ depends on the sources and the monomial vector $\boldsymbol{\mu}_m$ depends on the targets.

4.1. Approximating the electric field

Computing the near-field approximation for the derivatives of the potential is straightforward. From (38), we note that

$$\frac{\partial}{\partial x_m}\phi(\mathbf{x}_m, C) \approx \sum_{i,j,k=0}^{3} a_{ijk}\, y^j z^k \frac{\partial x^i}{\partial x_m} = \frac{1}{\Delta x} \sum_{i,j,k=0}^{3} a_{ijk}\, i\, x^{i-1} y^j z^k, \tag{42}$$
$$= \frac{1}{\Delta x} \boldsymbol{\alpha}_c^T \boldsymbol{\mu}_{m,i}, \tag{43}$$

where

$$\boldsymbol{\mu}_{m,i} = \sum_{i,j,k=0}^{3} i\, x^{i-1} y^j z^k. \tag{44}$$

Similarly

$$\frac{\partial}{\partial y_m}\phi(\mathbf{x}_m, C) \approx \frac{1}{\Delta y} \boldsymbol{\alpha}_c^T \boldsymbol{\mu}_{m,j}, \tag{45}$$

$$\frac{\partial}{\partial z_m}\phi(\mathbf{x}_m, C) \approx \frac{1}{\Delta z} \boldsymbol{\alpha}_c^T \boldsymbol{\mu}_{m,k}, \tag{46}$$

with

$$\boldsymbol{\mu}_{m,j} = \sum_{i,j,k=0}^{3} j\, x^i y^{j-1} z^k, \tag{47}$$

and

$$\boldsymbol{\mu}_{m,k} = \sum_{i,j,k=0}^{3} k\, x^i y^j z^{k-1}. \tag{48}$$

The electric field at target position $\mathbf{x}_m$ in cluster C due to the source particles in the interaction list $\{\mathbf{y}_s\}$ is given by

$$\mathbf{E}_m = -\nabla_{\mathbf{x}_m}\phi(\mathbf{x}_m, \mathbf{y}_s) \approx -\boldsymbol{\alpha}_c^T \begin{bmatrix} \dfrac{\boldsymbol{\mu}_{m,i}}{\Delta x} \\ \dfrac{\boldsymbol{\mu}_{m,j}}{\Delta y} \\ \dfrac{\boldsymbol{\mu}_{m,k}}{\Delta z} \end{bmatrix}. \tag{49}$$

The cluster-particle treecode algorithm is described in Algorithm 2. First, the target particles are hierarchically reordered into a tree following the same procedure described in the particle-cluster algorithm. The rest of the algorithm is done in two stages. Stage 1 loops through the source particles and performs the interaction of each source particle $\mathbf{y}_n$ with the clusters in the tree. If a source particle $\mathbf{y}_n$ and a cluster are well-separated, that means that $\mathbf{y}_n$ is in the interaction list of that particular cluster and the algorithm updates the near-field modified tricubic coefficients in (41) for the cluster. Otherwise the source particle interacts with the children of the cluster unless the cluster is a leaf in which case the cluster-particle interaction is computed by direct sum. Stage 2 completes the evaluation of the near-field approximation by descending the tree and evaluating (39) and (49) for all target particles that interacted with source particles by approximation in stage 1.

The tree of targets has $O(\log N)$ levels. In both stage 1 and stage 2, the code descends through the tree. In stage 1, the code loops through the $\log N$ levels for each of the N source particles to evaluate (41), thus the operation count is $O(N \log N)$. In stage 2 the code evaluates (39) and (49) for each of the N target sites at each of the $\log N$ levels resulting in a cost of $O(N \log N)$. Thus the cluster-particle treecode also has an overall cost of $O(N \log N)$.

Algorithm 2 tricubic treecode: cluster-particle

input: particle coordinates $\mathbf{x}_m$, $m = 1, \ldots, N$, f_n, $n = 1, \ldots, N$, parameters θ, N_0
Set $\{\mathbf{y}_n\}_{n=1}^N = \{\mathbf{x}_m\}_{m=1}^N$
output: potential ϕ_m, electric field $\mathbf{E}_m$, $m = 1, \ldots, N$
program **main**
    build tree of target particles $\mathbf{x}_m$
    for $n = 1, \ldots, N$, **compute_cp_stage1**(root, $\mathbf{y}_n$), end for
    compute_cp_stage2(root)
end program
subroutine **compute_cp_stage1**(C, $\mathbf{y}$)
    if MAC (36) is satisfied
        update modified tricubic coefficients $\boldsymbol{\alpha}_c$ using (41)
    else if C is a leaf
        compute cluster-particle interaction by direct summation
    else
        for each child C' of C, **compute_cp_stage1**(C', $\mathbf{y}$), end for
end subroutine
subroutine **compute_cp_stage2**(C)
    if C interacted with a source particle by tricubic approximation in stage 1
        for each target $\mathbf{x}_m$ in C, compute the full approximation in (39) and (49), end for
    for each child C' of C, **compute_cp_stage2**(C'), end for
end subroutine

5. Numerical results

5.1. Implementation details

The algorithms are written in double precision C++ using the Clang compiler frontend with the -O2 optimization. The source code is available online in a Github repository [9]. The tests presented here were performed on a Dell PowerEdge R940xa Linux box with 2.1GHz Intel Xeon Gold processors.

5.2. Efficiency of the treecode algorithms

This section presents results for the particle-cluster and cluster-particle treecode approximations of the potential and electric field for systems of size $N \in \{10^4, 8\times10^4, 64\times10^4\}$. where the particles are randomly distributed in a cube of dimension $[-5,5]\times[-5,5]\times[0,10]$ and the weights $f_n \in (-1,1)$. The maximum number of particles in a leaf of the tree is set to $N_0 = 1000$ and the MAC parameter $\theta = 0.3 : 0.1 : 0.8$.

Let $r = |\mathbf{x}-\mathbf{y}|$ and $\kappa = 1$. We test the two treecode algorithms on three kernels common in physical applications; the Coulomb kernel

$$\mathcal{K}(\mathbf{x},\mathbf{y}) = \frac{1}{r}, \tag{50}$$

the screened Coulomb kernel

$$\mathcal{K}(\mathbf{x},\mathbf{y}) = \frac{\mathrm{e}^{-\kappa r}}{r}, \tag{51}$$

and the kernel of the real-space component of the Ewald sum

$$\mathcal{K}(\mathbf{x},\mathbf{y}) = \frac{\mathrm{erfc}(\kappa r)}{r}. \tag{52}$$

For each kernel, we approximate ϕ in (1) and the electric field $\nabla\phi$ using equations (19) and (30) for the particle-cluster treecode and equations (39) and (49) for the cluster-particle treecode. We compute the relative error, in ℓ^2-norm, in the approximation of the potential, $\mathrm{Error}(\phi)$, given by

$$\mathrm{Error}(\phi) = \Big(\sum_{m=1}^{N} \Big|\phi^d(\mathbf{x}_m) - \phi^t(\mathbf{x}_m)\Big|^2 \Big/ \sum_{m=1}^{N} \Big|\phi^d(\mathbf{x}_m)\Big|^2 \Big)^{1/2}, \tag{53}$$

as well as in the approximation of the electric field, $\mathrm{Error}(\nabla\phi)$, defined as

$$\mathrm{Error}(\nabla\phi) = \Big(\sum_{m=1}^{N} \Big|\nabla\phi^d(\mathbf{x}_m) - \nabla\phi^t(\mathbf{x}_m)\Big|^2 \Big/ \sum_{m=1}^{N} \Big|\nabla\phi^d(\mathbf{x}_m)\Big|^2 \Big)^{1/2}, \tag{54}$$

where $\phi^d, \nabla\phi^d$ are the exact potential and electric field computed by direct summation, $\phi^t, \nabla\phi^t$ are the treecode approximations.

Figures 6 and 7 show the results of the treecode approximations for a system with varying density and constant density respectively.

In Figure 6, the particles are randomly distributed in a cube of dimension $[-5,5]\times[-5,5]\times[0,10]$. With $N \in \{10^4, 8\times10^4, 64\times10^4\}$, the particle density $\rho \in \{10, 80, 640\}$. The top row is a plot of the error in the potential against N. The middle row is a plot of the error in the electric field against N and the bottom row is a plot of the CPU time against the N. The plots are for both particle-cluster and cluster-particle and for all three kernels. Both algorithms have the same qualitative behavior for all three kernels. For a fixed MAC parameter θ, equations (35) and (37) predict that the error

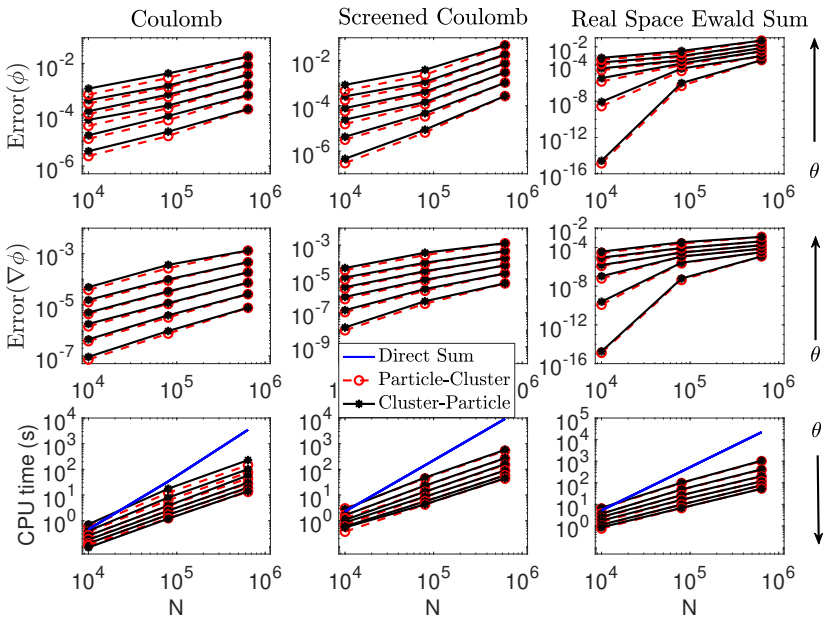

Figure 6: Accuracy and CPU time in N for the three kernels. N: number of particles, θ: MAC parameter. Fixed length of cube, $L = 10$.

increases with ρ or with N for a fixed length L. We see this increase in error as N increases for all the kernels for both algorithms. As expected, for fixed N, the error decreases with decreasing θ. The plot of CPU time against N shows the $O(N \log N)$ behavior of both algorithms compared with the $O(N^2)$ behavior of direct sum.

For Figure 7, the particles are randomly distributed in the cube $[-L, L]^3$. With $N \in \{10^4, 8 \times 10^4, 64 \times 10^4\}$, the length $L \in \{10, 20, 40\}$ in order to maintain a constant density of $\rho = 10$. As expected from equations (35) and (37), the error in the potential (top row) and the electric field (middle row) are near constant with N with constant density as expected. Again, both algorithms exhibit very similar behavior for all our test parameters.

5.3. Molecular dynamics simulation of liquid Argon (Ar)

To further investigate the accuracy of the algorithms, we performed a molecular dynamics (MD) simulation of liquid Argon. The interactions of Argon atoms are governed by the Lennard-Jones potential energy

$$\psi(r) = 4\epsilon\Big[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\Big]. \tag{55}$$

We implemented the particle-cluster treecode in the MD simulation software package DL_POLY Classic [36] to approximate the Lennard-Jones potential energy and the forces $-\nabla\psi(r)$. The Lennard-Jones potential well-depth parameter $\epsilon = 0.9661$ kJ/mol and the distance at which there is zero potential energy $\sigma = 3.405$Å. We simulated a system of $N = 100$ Argon atoms in a cubic box of length 17.4Å, with periodic boundary conditions, at 85^oK and with timestep 1fs. The system was equilibrated for 5000 MD steps using an Evans thermostat [36] after which the thermostat was turned off and statistics were taken over 20000 additional MD steps.

We run four different simulations. In one simulation, the atomic interactions were computed using direct summation with no cutoffs, or a treecode with $\theta \approx 0$. In the other three simulations, the atomic interactions were computed with the particle-cluster treecode with $\theta \in \{0.3, 0.5, 0.7\}$

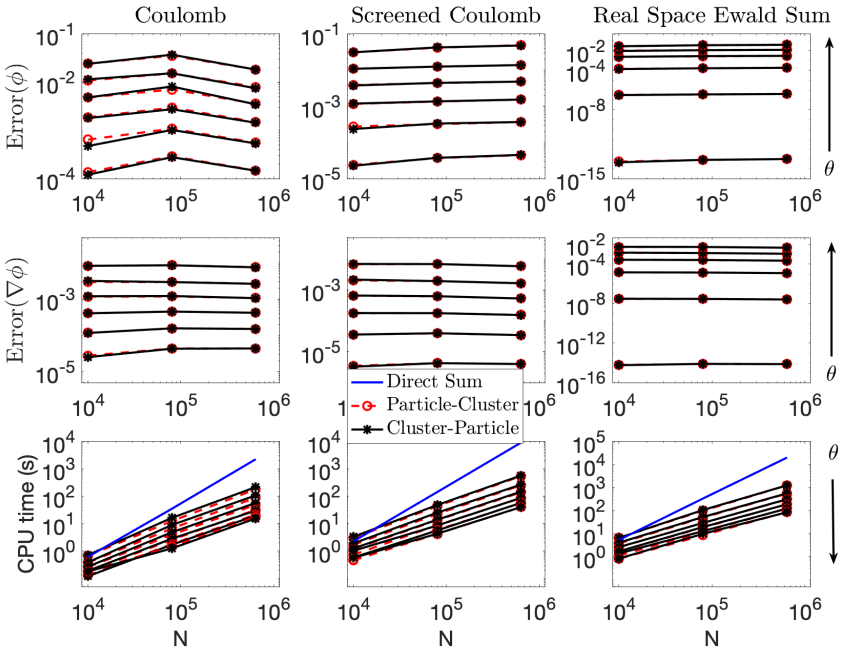

Figure 7: Accuracy and CPU time vs N for the three kernels. N: number of particles, θ: MAC parameter. Fixed particle density, $\rho = 10$.

and the maximum number of particles in a leaf $N_0 = 4$. In all the simulations, we computed the radial distributions functions $g(r)$ [1] every 10 steps averaged over all the particles and over 2000 steps. We also computed the velocity $C_{VV}(t)$ and force-force $C_{FF}(t)$ autocorrelation functions [1] with velocities and forces which were stored at each 5th step. The correlation functions were also averaged over all the atoms.

In Figure 8 we compare the radial distribution functions $g(r)$ of the direct summation to the treecode for the three MAC values $\theta \in \{0.3, 0.5, 0.7\}$. The radial distribution function is a structural property that provides a measure of the arrangement of atoms in the liquid. The radial distribution function is not very sensitive to accuracy differences, thus all the three treecode simulation results match very well with the direct sum results.

Figures 9 and 10 are the velocity-velocity and force-force autocorrelation functions respectively. These are dynamical quantities and are more sensitive to accuracy differences. Comparisons of the correlation functions provide a measure of the similarity of the dynamics of the MD simulations using the treecodes to that of the direct sum. Although all three treecodes are qualitatively similar to the direct sum results, the treecode with $\theta = 0.3$ provides the best overall quantitative match as expected. Treecode algorithms are typically used to approximate long-range kernels. The Lennard-Jones potential, however, is a short-range kernel. The simulation results show that the treecode can provide efficient approximations for a short-range kernel as well. This suggests that in MD simulations with both short-range and long-range potentials, such as in electrostatic systems with both Lennard-Jones and Coulomb potentials, a treecode can be used to provide simultaneously approximation of both potentials in order to achieve better computational speed, instead of the standard approach of computing the short-range and long-range interactions separately.

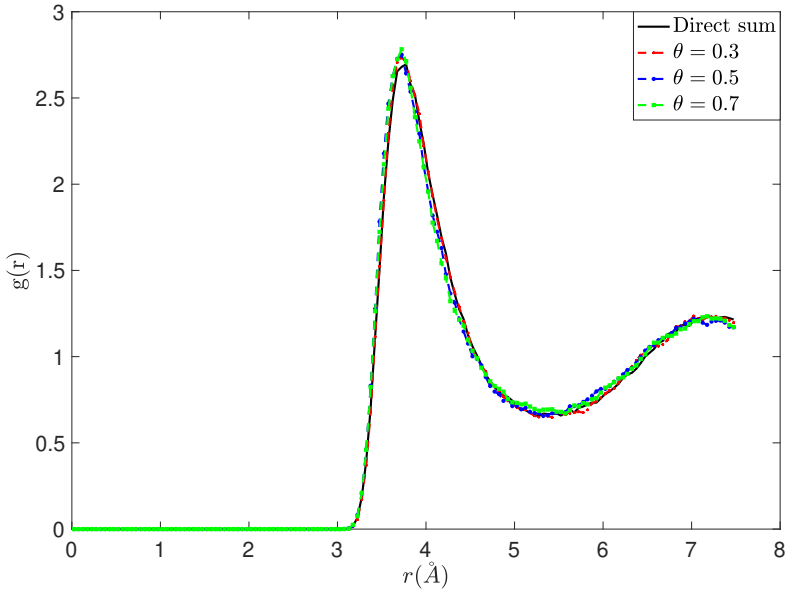

Figure 8: Radial Distribution Function $g(r)$

6. Conclusions

This paper developed two treecode methods, particle-cluster and cluster-particle, based on a tricubic interpolation method in three dimensions. The kernel representing pairwise particle interactions is interpolated by a cubic polynomial in three dimensions in a way that is computationally efficient and allows straightforward approximations of the derivatives of the interpolated kernel.

An error analysis was provided that shows that the decay rate of the error in the approximation of an interaction between a particle and a cluster is quartic in the MAC, θ. A numerical evidence for quartic decay of the error was also provided.

We performed numerical tests on the Coulomb, screened Coulomb and real space Ewald sum kernels. The numerical tests demonstrated the typical $O(N \log N)$ scaling of the treecode for both versions of the treecode. Additionally, the numerical tests showed, as expected, that particle-cluster and cluster-particle have similar numerical efficiency when the targets and sources in the treecode algorithm are the same.

We also provided an application of the particle-cluster treecode in a molecular dynamics simulation of liquid Ar. The simulation results provided evidence that the treecode is able to reproduce both structural and dynamical properties of a chemical system, even for the short-range Lennard-Jones kernel.

The algorithms presented here used analytical derivatives of the kernels. One extension of the work is to develop a kernel independent extension of the algorithms which uses numerical derivatives for the kernels. The tricubic interpolation employed in the treecodes guarantees global $\mathcal{C}^1$ continuity. In a follow up paper, we study the effect of smoothness on the accuracy of treecode methods. Higher smoothness can only be achieved through the use of higher order interpolating polynomials, such as a triquintic. Future extensions of this work will develop and implement higher order interpolations to achieve higher global smoothness.

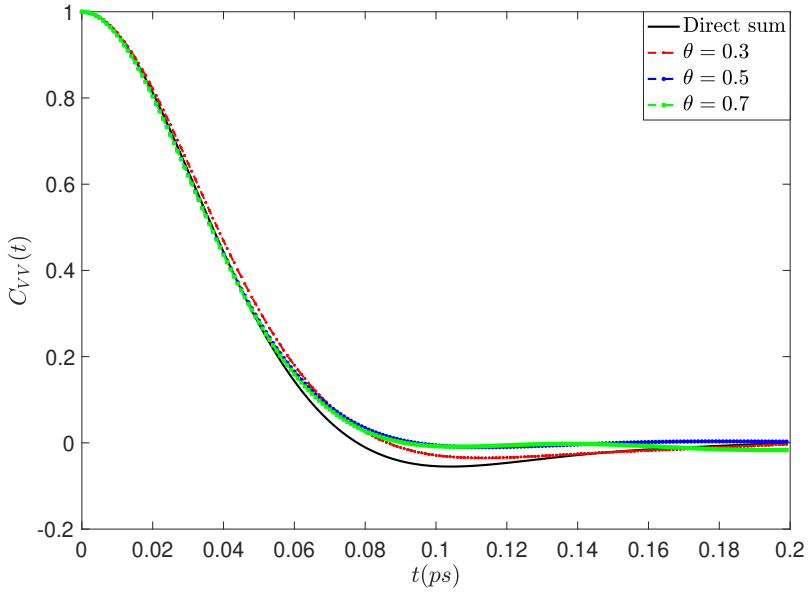

Figure 9: Velocity-velocity autocorrelation function $C_{VV}(t)$

Acknowledgments

The work of HAB was partially supported by the National Science Foundation grant CHE-2016048 and start-up funds from San Francisco State University. The work of ST was partially supported by the National Science Foundation grant DMS-2012371. Partial support is also acknowledged from the Visiting Faculty Program of the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS). We used computing resources provided by San Francisco State University as well as resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory (LBNL), operated under Contract No. DE-AC02-05CH11231.

References

[1] M. P. Allen and D. J. Tildesly *Computer simulation of liquids*, 1st ed.; Oxford University Press, (1987)

[2] E. Agullo, B. Bramas, O. Coulaud, E. Darve, M. Messner and T. Takahashi, *Task-based FMM for multicore architectures*, SIAM J. Sci. Comput., 36 (2014), pp. C66–C93.

[3] L. af Klinteberg, D. S. Shamshirgar and A.-K. Tornberg, *Fast Ewald summation for free-space Stokes potentials*, Res. Math. Sci., 4 (2017), Article 1.

[4] C. R. Anderson, *An implementation of the Fast Multipole Method without multipoles*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 923–947.

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

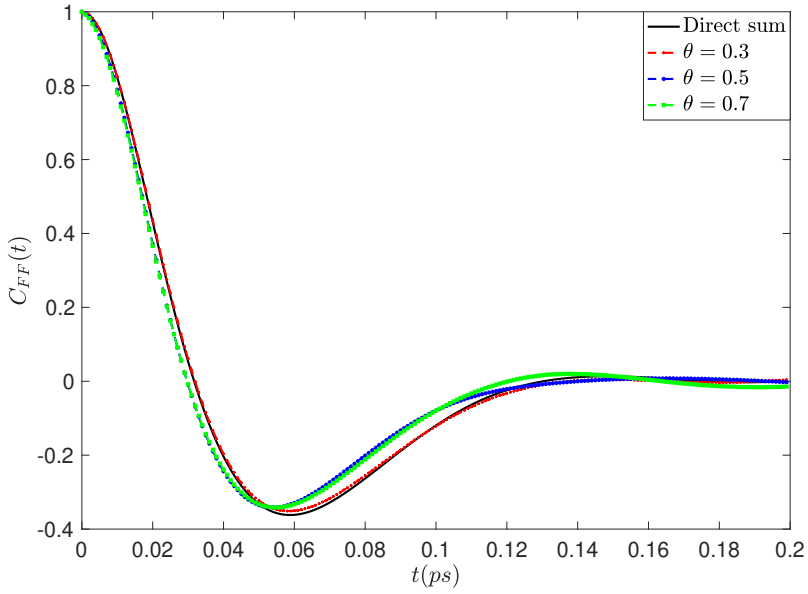

Figure 10: Force-force autocorrelation function $C_{FF}(t)$

[6] H. A. Boateng and R. Krasny, *Comparison of Treecodes for Computing Electrostatic Potentials in Charged Particle Systems with Disjoint Targets and Sources*, J. Comput. Chem., 34 (2015), pp. 2159–2167.

[7] H. A. Boateng and I. T. Todorov, *Arbitrary order permanent Cartesian multipolar electrostatic interactions*, J. Chem. Phys., 142, 034117 (2015), pp. 1–13.

[8] H. A. Boateng, *Mesh-free hierarchical clustering methods for fast evaluation of electrostatic interactions of point multipoles*, J. Chem. Phys., 147, 164104 (2017), pp. 1–16.

[9] H. Boateng and S. Tlupova, *C-1 tricubic treecode*, https://github.com/haboateng/C-1-tricubic-treecode, Last assessed: 03-26-2022

[10] H. Cheng, L. Greengard and V. Rokhlin, *A fast adaptive multipole algorithm in three dimensions*, J. Comput. Phys., 155 (1999), pp. 468–498.

[11] C. I. Draghicescu and M. Draghicescu, *A fast algorithm for vortex blob interactions*, J. Comput. Phys., 116 (1995), pp. 69–78.

[12] Z.-H. Duan and R. Krasny, *An adaptive treecode for computing nonbonded potential energy in classical molecular systems*, J. Comput. Chem., 22 (2001), pp. 184–195.

[13] U. Essmann, L. Perera, M. Berkowitz, T. Darden, H. Lee and L. Pedersen, *A smooth particle mesh Ewald method*, J. Chem. Phys., 103 (1995), pp. 8577–8593.

[14] W. Fong and E. Darve, *The black-box fast multipole method*, J. Comput. Phys., 228 (2009), pp. 8712–8725.

[15] W.-H. Geng and R. Krasny, *A treecode-accelerated boundary integral Poisson-Boltzmann solver for solvated biomolecules*, J. Comput. Phys., 247 (2013), pp. 62–78.

[16] Z. Gimbutas and V. Rokhlin, *A generalized Fast Multipole Method for nonoscillatory kernels*, SIAM J. Sci. Comput., 24 (2002), pp. 796–817.

[17] L. F. Greengard and J. Huang, *A new version of the Fast Multipole Method for screened Coulomb interactions in three dimensions*, J. Comput. Phys., 180 (2002), pp. 642–658.

[18] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[19] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA 1988.

[20] L. Greengard, *The Numerical Solution of The N-Body Problem*, Comput. Phys., 4 (1990), pp. 142–152.

[21] L. Greengard, *Fast Algorithms For Classical Physics*, Science, 265 (1994), pp. 909–914.

[22] D. J. Hardy, M. A. Wolff, J. Xia, K. Schulten and R. D. Skeel, *Multilevel summation with B-spline interpolation for pairwise interactions in molecular dynamics simulations*, J. Chem. Phys., 144 (2016), Article 114112.

[23] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, Taylor & Francis, Bristol, 1988.

[24] R. Krasny and L. Wang, *A treecode based on barycentric Hermite interpolation for electrostatic particle interactions*, Comput. Math. Biophys., 7 (2019), pp. 73–84.

[25] I. Lashuk, A. Chandramowlishwaran, H. Langston, T.-A. Nguyen, R. Sampath, A. Shringarpure, R. Vuduc, L. Ying, D. Zorin and G. Biros, *A massively parallel adaptive fast-multipole method on heterogeneous architectures*, Commun. ACM, 55 (2012), pp. 101–109.

[26] F. Lekien and J. Marsden, *Tricubic interpolation in three dimensions*, Int. J. Numer. Meth. Engng, 63 (2005), pp. 455–471.

[27] P.-D. Létourneau, C. Cecka and E. Darve, *Cauchy Fast Multipole Method for general analytic kernels*, SIAM J. Sci. Comput., 36 (2014), pp. A396–A426.

[28] P. Li, H. Johnston and R. Krasny, *A Cartesian treecode for screened Coulomb interactions*, J. Comput. Phys., 228 (2009), pp. 3858–3868.

[29] Z. Liang, Z. Gimbutas, L. Greengard, J. Huang and S. Jiang, *A fast multipole method for the Rotne-Prager-Yamakawa tensor and its applications*, J. Comput. Phys., 234 (2013), pp. 133–139.

[30] K. Lindsay and R. Krasny, *A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow*, J. Comput. Phys., 172 (2001), pp. 879–907.

[31] J. Makino, *Yet another fast multipole method without multipoles - Pseudoparticle multipole method*, J. Comput. Phys., 151 (1999), pp. 910–920.

[32] D. Malhotra and G. Biros, *Algorithm 967: A distributed-memory fast multipole method for volume potentials*, ACM Trans. Math. Softw., 43 (2016), Article 17.

[33] W. B. March, B. Xiao and G. Biros, *ASKIT: Approximate skeletonization kernel-independent treecode in high dimensions*, SIAM J. Sci. Comput., 37 (2015), pp. A1089–A1110.

[34] P. G. Martinsson and V. Rokhlin, *An accelerated kernel-independent fast multipole method in one dimension*, SIAM J. Sci. Comput., 29 (2007), pp. 1160–1178.

[35] R. D. Skeel, I. Tezcan and D. J. Hardy, *Multiple Grid Methods for Classical Molecular Dynamics*, J. Comput. Chem., 23 (2002), pp. 673–684.

[36] W. Smith, T. R. Forester and I. T. Todorov, *The DL_POLY Classic User Manual*, STFC Daresbury Laboratory: Daresbury, Warrington WA4 4AD, 2012.

[37] T. Takahashi, C. Cecka and E. Darve, *Optimization of the parallel black-box fast multipole method on CUDA*, in Proceedings of Conference on Innovative Parallel Computing (InPar), 2012, San Jose, CA, USA. DOI: 10.1109/InPar.2012.6339607.

[38] J. Tausch, *The Fast Multipole Method for arbitrary Green's functions*, Contemp. Math., 329 (2003), pp. 307–314.

[39] L. Wang, S. Tlupova and R. Krasny, *A treecode algorithm for 3D Stokeslets and stresslets*, Adv. Appl. Math. Mech., 11 (2019), pp. 737–756.

[40] L. Wang, R. Krasny and S. Tlupova, *A kernel-independent treecode algorithm based on barycentric Lagrange interpolation*, Comm. Comput. Phys., 28(4) (2020), pp. 1415–1436.

[41] M. S. Warren and J. K. Salmon, *A parallel hashed oct-tree N-body algorithm*, in Proceedings of the 1993 ACM/IEEE Conference on Supercomputing, 1993, pp. 12–21.

[42] Z. Xu, X. Cheng and H. Yang, *Treecode-based generalized Born method*, J. Chem. Phys., 134 (2011), Article 064107.

[43] L. Ying, G. Biros and D. Zorin, (2004) *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, J. Comput. Phys., 196 (2004), pp. 591–626.

[44] L. Ying, *A kernel independent fast multipole algorithm for radial basis functions*, J. Comput. Phys., 213 (2006), pp. 451–457.

[45] B. Zhang, J. Huang, N. P. Pitsianis and X. Sun, *A Fourier-series-based kernel-independent fast multipole method*, J. Comput. Phys., 230 (2011), pp. 5807–5821.