The effect of global smoothness on the accuracy of treecodes

Henry A. Boateng[1,*], and Svetlana Tlupova[2]

[1] *Department of Mathematics, San Francisco State University, San Francisco, CA 94132, USA.*
[2] *Department of Mathematics, Farmingdale State College, SUNY, Farmingdale, NY 11735, USA.*

Abstract. Treecode algorithms are widely used in evaluation of N-body pairwise interactions in $O(N)$ or $O(N\log N)$ operations. While they can provide high accuracy approximations, a criticism leveled at the methods is that they lack global smoothness. In this work, we study the effect of smoothness on the accuracy of treecodes by comparing three tricubic interpolation based treecodes with differing smoothness properties: a global $\mathcal{C}^1$ continuous tricubic, and two new tricubic interpolants, one that is globally $\mathcal{C}^0$ continuous and one that is discontinuous. We present numerical results which show that higher smoothness leads to higher accuracy for properties dependent on the derivatives of the kernel, nevertheless the global $\mathcal{C}^0$ continuous and discontinuous treecodes are competitive with the $\mathcal{C}^1$ continuous treecode. One advantage of the discontinuous treecode over the $\mathcal{C}^1$ continuous is that, in addition to function evaluations, the discontinuous treecode only requires evaluations of the first derivatives of the kernel while the $\mathcal{C}^1$ continuous treecode requires evaluations up to third order derivatives. When the first derivatives are computed using finite differences, the discontinuous version can be viewed as kernel independent and of utility for a wider array of kernels with minimal effort.

AMS subject classifications: 70-08, 70-10, 76-04, 85-04, 65D99,
Key words: Fast summation, Treecode, Tricubic interpolation.

1 Introduction

The evaluation of sums of the form

$$\phi(\mathbf{x}_m)=\sum_{n=1}^{N}\mathcal{K}(\mathbf{x}_m,\mathbf{y}_n)f_n, \qquad m=1,...,M, \tag{1.1}$$

is of interest in many applications in physics, chemistry, fluid dynamics, etc. Here $\{\mathbf{y}_n\},n=1,...,N$ is a set of particles with weights $\{f_n\}$, and $\phi(\mathbf{x})$ is a potential (or force or velocity). The kernel $\mathcal{K}(\mathbf{x},\mathbf{y})$ represents the pairwise interaction between a target particle $\mathbf{x}$ and a source particle $\mathbf{y}$. The kernel may be a scalar or a tensor, and it is understood that if the kernel is singular for $\mathbf{x}=\mathbf{y}$, then the sum omits the $n=m$ term.

The particle-particle interactions in equation (1.1) can be summed directly but that requires $O(MN)$ operations. This is a computationally intensive calculation when M,N are large, and many fast summation techniques have been developed to reduce the cost. Most of these methods can be grouped into tree-based methods and particle-mesh methods. In particle-mesh methods, the particles are projected onto a uniform grid where the FFT or multigrid can be used (e.g. P3M [20], particle-mesh Ewald [14], spectral Ewald [1], multilevel summation [19]).

*Corresponding author. *Email addresses:* `boateng@sfsu.edu` (HA. Boateng), `tlupovs@farmingdale.edu` (S. Tlupova)

In the tree-based approach, the particles are hierarchically partitioned into a tree structure, and the particle-particle interactions in equation (1.1) are replaced with particle-cluster or cluster-cluster approximations, and reduce the cost to $O(N)$ [17, 18] or $O(N\log N)$ [2]. The original treecode [2] used a monopole far-field approximation, while the fast multipole method (FMM) [17, 18] improved on this by employing higher-order far-field and near-field approximations expressed in terms of classical analytic multipole expansions. Later versions of the FMM used plane wave expansions for the 3D Laplace kernel [10] and spherical Bessel function expansions for the Yukawa potential [16]. Methods based on Cartesian Taylor expansions were also developed for some common kernels [3, 4, 12, 13, 24, 25, 28, 29]. These methods use analytic expansions specific to each kernel. More recently, kernel-independent methods have been developed that only require kernel evaluations and as such are applicable to more general kernel functions. Among these are the kernel-independent FMM which uses equivalent particle distribution determined by solving linear systems [32, 33], a black-box FMM which uses polynomial interpolation at Chebyshev points combined with SVD compression [15], and a kernel-independent treecode which uses barycentric Langrange interpolation at Chebyshev points [30]. Another recent development is a treecode based on barycentric Hermite polynomial interpolation [21] which requires third order derivatives of the kernel at Chebyshev evaluation points. The black-box FMM [15], the barycentric Lagrange interpolation [30] and the barycentric Hermite interpolation treecodes [21] all use a tensor product of three one-dimensional polynomials to approximate the three-dimensional kernel function.

While treecodes can provide high accuracy approximations, the approximations lack global continuity [27]. For example, in molecular dynamics (MD) simulations, the discontinuity can lead to an energy drift when the treecode multipole expansion is low order [27]. Higher order expansions resolve the issue of energy drift but at higher computational cost. In this paper, we seek to understand any effect that the smoothness of the approximations may have on the accuracy of the overall treecode algorithm. We do this by comparing treecodes based on tricubic interpolations of different smoothness. Tricubic interpolation [23] is a method of local approximation of a function defined on a regular grid in three dimensions. The function is approximated within a unit cube by a polynomial in the three spatial variables, with the unknown coefficients determined by requiring the polynomial to match the function and its derivatives at points on and within the unit cube. While tricubic interpolation is equivalent to a sequential application of three one-dimensional cubic interpolants [23], this formulation is intrinsically three-dimensional, which has better computational efficiency when the interpolant is used at multiple points inside each cube element. Notably for this work, this formulation is highly advantageous when the derivatives of the interpolated function are needed, because the tricubic polynomial can be easily differentiated analytically.

We present a comparative study of $O(N\log N)$ treecode methods that use tricubic interpolation of different degrees of smoothness. In our previous work [6], we developed a treecode algorithm based on a tricubic interpolant [23] with global $\mathcal{C}^1$ continuity in approximating the kernel. This interpolant requires up to third order derivatives of the kernel. Here, we develop two new tricubic interpolants, one with global $\mathcal{C}^0$ continuity and one that is discontinuous across the faces of the cube. The latter has the advantage of only requiring the kernel and its first order derivatives. We investigate the role the degree of smoothness in kernel interpolation may have on the accuracy of the derivative of the function ϕ in (1.1) when compared to the accuracy in ϕ itself. We compare the Coulomb potential with its electric field, and the fluid velocity in the method of regularized Stokeslets [11] with its divergence. This study points to an increase of accuracy in the electric field and divergence when an interpolant of stronger smoothness properties is used. Simulations of particle suspensions in viscous flows [31] demonstrate that numerical methods based on divergence-free interpolation yield particle distributions that are more similar to the exact solution, even while having higher error per particle. We provide further evidence of the correlation of smoothness with accuracy by comparing results from MD simulations of liquid Argon where the energy and forces are computed using the the three tricubic treecodes.

The paper is organized as follows. In Section 2, we review the general approach of tricubic interpolation with global $\mathcal{C}^1$ continuity. We then develop two additional interpolation methods, one with global $\mathcal{C}^0$ continuity and another which is discontinuous. In Section 3, we demonstrate how an approximation for a particle-cluster interaction based on tricubic interpolation is obtained, both for ϕ and its gradient. This forms the basis of our treecode algorithm. Section 4 presents the treecode performance in terms of accuracy and CPU time for the Coulomb potential and its electric field, the computation of the regularized Stokeslet velocity and divergence, and a comparison of results from the molecular dynamics simulation of liquid Argon. Conclusions and future work are discussed in Section 5.

2 Tricubic interpolation

Tricubic interpolation represents a function $f(x,y,z)$ locally as a piecewise cubic polynomial of the form

$$g(x,y,z)=\sum_{i,j,k=0}^{3} a_{ijk}x^i y^j z^k = \boldsymbol{\alpha}^T\boldsymbol{\mu}, \tag{2.1}$$

within a mesh element that is a unit cube $0\le x,y,z\le 1$. Here, the unknown coefficients a_{ijk} are ordered into a vector $\boldsymbol{\alpha}$ by defining

$$\alpha_{1+i+4j+16k}=a_{ijk}, \qquad i,j,k=0,1,2,3, \tag{2.2}$$

and the monomial basis of the tricubic is ordered into the vector $\boldsymbol{\mu}$ through a similar definition

$$\mu_{1+i+4j+16k}=x^i y^j z^k, \qquad i,j,k=0,1,2,3. \tag{2.3}$$

Sixty-four linearly independent constraints are required to determine the 64 coefficients a_{ijk}. These constraints are obtained by enforcing a subset of the eight equality conditions

$$\frac{\partial^{\gamma+\nu+\omega}g(x,y,z)}{\partial x^\gamma \partial y^\nu \partial z^\omega}=\frac{\partial^{\gamma+\nu+\omega}f(x,y,z)}{\partial x^\gamma \partial y^\nu \partial z^\omega}, \qquad \gamma,\nu,\omega=0,1, \tag{2.4}$$

at the vertices or inside the unit cube. We will group the eight conditions into the following four sets

$$\begin{aligned} S_0 &: \{g(x,y,z)=f(x,y,z)\},\\ S_1 &: \left\{\frac{\partial g}{\partial x}=\frac{\partial f}{\partial x},\ \frac{\partial g}{\partial y}=\frac{\partial f}{\partial y},\ \frac{\partial g}{\partial z}=\frac{\partial f}{\partial z}\right\},\\ S_2 &: \left\{\frac{\partial^2 g}{\partial x\partial y}=\frac{\partial^2 f}{\partial x\partial y},\ \frac{\partial^2 g}{\partial x\partial z}=\frac{\partial^2 f}{\partial x\partial z},\ \frac{\partial^2 g}{\partial y\partial z}=\frac{\partial^2 f}{\partial y\partial z}\right\},\\ S_3 &: \left\{\frac{\partial^3 g}{\partial x\partial y\partial z}=\frac{\partial^3 f}{\partial x\partial y\partial z}\right\}. \end{aligned} \tag{2.5}$$

Enforcing the 64 linearly independent constraints on the polynomial g and the function f leads to a sparse linear system

$$B\boldsymbol{\alpha}=\mathbf{b}, \tag{2.6}$$

where B is a 64×64 invertible matrix generated by applying the constraints to the polynomial and $\mathbf{b}$ is obtained by applying the constraints to the function. The system can then be solved explicitly as

$$\boldsymbol{\alpha} = B^{-1}\mathbf{b} \tag{2.7}$$

to provide the coefficient vector $\boldsymbol{\alpha}$. The 64 constraints determine the global smoothness of the tricubic interpolation. In the next three subsections, we present a global $\mathcal{C}^1$ continuous tricubic, a global $\mathcal{C}^0$ continuous tricubic and finally a tricubic which lacks global continuity.

2.1 $\mathcal{C}^1$ continuous tricubic [23]

The $\mathcal{C}^1$ continuous tricubic was developed by Lekien and Marsden (LM) [23]. In their work, LM proved that

- The set $S_0 \cup S_1 \cup S_2$ enforced at the vertices of the cube is a necessary and sufficient condition for $\mathcal{C}^0$ continuity.
- The set $S_0 \cup S_1 \cup S_2 \cup S_3$ enforced at the vertices of the cube is a necessary and sufficient condition for $\mathcal{C}^1$ continuity.

Hence, the $\mathcal{C}^1$ continuous tricubic enforces the full set of eight equality constraints at the eight corners $P_1,\ldots,P_8$ of the unit cube, shown in Figure 1 (left). The function and its derivatives are stacked into a vector $\mathbf{b}$ as follows,

$$b_i = \begin{cases} f(P_i), & 1\le i\le 8,\\ \dfrac{\partial f}{\partial x}(P_{i-8}), & 9\le i\le 16,\\ \dfrac{\partial f}{\partial y}(P_{i-16}), & 17\le i\le 24,\\ \dfrac{\partial f}{\partial z}(P_{i-24}), & 25\le i\le 32,\\ \dfrac{\partial^2 f}{\partial x\partial y}(P_{i-32}), & 33\le i\le 40,\\ \dfrac{\partial^2 f}{\partial x\partial z}(P_{i-40}), & 41\le i\le 48,\\ \dfrac{\partial^2 f}{\partial y\partial z}(P_{i-48}), & 49\le i\le 56,\\ \dfrac{\partial^3 f}{\partial x\partial y\partial z}(P_{i-56}), & 57\le i\le 64. \end{cases} \tag{2.8}$$

Evaluating the polynomial in (2.1) and its derivatives at the eight corners of the cube then leads to a sparse linear system for the unknown coefficients which can be solved explicitly using equation (2.7).

The matrix B^{-1} is sparse and is computed exactly without numerical error [23]. It has exactly 1000 non-zero elements and a condition number $\kappa_2(B^{-1}) = 1.345\times 10^4$. In the treecode algorithm, only multiplication by the transpose of the inverse $(B^{-1})^T$ is needed [6], and as this matrix is sparse, the multiplication can be done in-line.

LM also showed that the representation (2.1) has the minimum order necessary to maintain global $\mathcal{C}^1$ continuity in the approximated function. Another advantage of the interpolant in the form of (2.1) is that the derivatives of the interpolated function can be computed analytically,

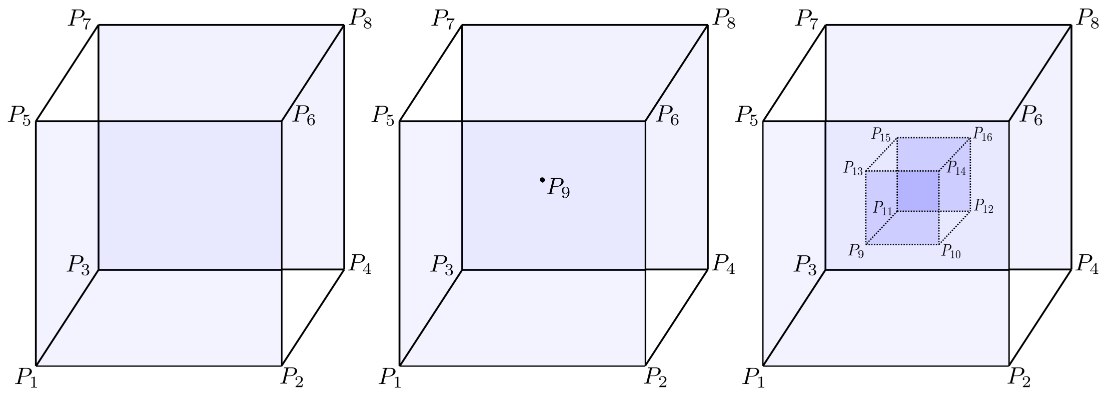

Figure 1: Schematics of the unit cube for three tricubic interpolants. Left: The unit cube for the $\mathcal{C}^1$ continuous tricubic [23]; center: unit cube with one additional point at the center for $\mathcal{C}^0$ tricubic; right: unit cube with 8 additional points inside for the globally discontinuous tricubic.

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.

If analytical expressions for the derivatives of the function f are unavailable at the corners of the cube for (2.8), various techniques such as finite differences can be used. On the other hand, while analytical expressions of derivatives of many kernels can be found easily, for some frequently used kernels, such as the regularized Stokeslets [11], the derivatives of higher order have rather complicated expressions. This has necessitated the development of the discontinuous tricubic we present later in Section 2.3, which requires only evaluations of the function and its first derivatives.

2.2 $\mathcal{C}^0$ continuous tricubic

In order to generate a tricubic with global $\mathcal{C}^0$ continuity, we remove the S_3 constraint from the vertices. We consider the unit cube with an additional point $P_9=(1/2,1/2,1/2)$ at the center of the cube, see Figure 1(center). At the corners of the cube, we evaluate $S_0\cup S_1\cup S_2$, while at the center P_9 we evaluate all $S_0\cup S_1\cup S_2\cup S_3$. We stack these values into the right hand side of a linear system of the form (2.6) as follows,

$$b_i=\begin{cases} f(p_i), & 1\le i\le 9,\\ \dfrac{\partial f}{\partial x}(P_{i-9}), & 10\le i\le 18,\\ \dfrac{\partial f}{\partial y}(P_{i-18}), & 19\le i\le 27,\\ \dfrac{\partial f}{\partial z}(P_{i-27}), & 28\le i\le 36,\\ \dfrac{\partial^2 f}{\partial x\partial y}(P_{i-36}), & 37\le i\le 45,\\ \dfrac{\partial^2 f}{\partial x\partial z}(P_{i-45}), & 46\le i\le 54,\\ \dfrac{\partial^2 f}{\partial y\partial z}(P_{i-54}), & 55\le i\le 63,\\ \dfrac{\partial^3 f}{\partial x\partial y\partial z}(P_9), & i=64. \end{cases} \tag{2.9}$$

The resulting 64×64 matrix is again sparse with 1602 non-zero integer elements and a condition number $\kappa_2(B) = 2.11\times 10^5$. The inverse of the matrix, B^{-1} is once again computed exactly, i.e. without numerical error, and is available in a Github repository [7].

We note here that this method will not retain global $\mathcal{C}^1$ continuity in the approximated function. While the interpolated function is $\mathcal{C}^\infty$ inside each element, the global $\mathcal{C}^1$ continuity is achieved if and only if the function f and its three first derivatives are continuous on each of the six faces of the cube [23]. As we are not enforcing continuity in the third derivative at the corners, we lose the global $\mathcal{C}^1$ continuity but the interpolant is $\mathcal{C}^0$ continuous.

2.3 Discontinuous tricubic

Finally, we develop the discontinuous tricubic interpolant by evaluating $S_0 \cup S_1$ at the corners of the cube as well as at the following additional 8 points within the unit cube:

$$\begin{aligned}
P_9 &= (1/4,1/4,1/4),\\
P_{10} &= (3/4,1/4,1/4),\\
P_{11} &= (1/4,3/4,1/4),\\
P_{12} &= (3/4,3/4,1/4),\\
P_{13} &= (1/4,1/4,3/4),\\
P_{14} &= (3/4,1/4,3/4),\\
P_{15} &= (1/4,3/4,3/4),\\
P_{16} &= (3/4,3/4,3/4).
\end{aligned}$$

These eight internal points are the corners of the smaller cube, as shown in Figure 1(right). The choice of these points is motivated in part by the resulting linear system having an inverse matrix with integer or fractional coefficients and thus being easily computed for practical use.

The resulting system (2.6) has the following right hand side:

$$b_i = \begin{cases} f(P_i), & 1 \le i \le 16,\\ \dfrac{\partial f}{\partial x}(P_{i-16}), & 17 \le i \le 32,\\ \dfrac{\partial f}{\partial y}(P_{i-32}), & 33 \le i \le 48,\\ \dfrac{\partial f}{\partial z}(P_{i-48}), & 49 \le i \le 64. \end{cases} \tag{2.10}$$

The 64×64 system matrix B is again sparse with 2260 non-zero integer or fractional elements and a condition number $\kappa_2(B) = 1.60\times 10^6$. The matrix $27B^{-1}$, has integer elements only, and it is provided online in a Github repository [7].

With this method, the approximated function will be discontinuous across the faces of the cube. This is because continuity in the second derivative at the corners, as needed for $\mathcal{C}^0$ continuity across the faces [23], is no longer enforced. At the same time, this reduces the need for higher order derivatives of kernels that have rather complicated analytical expressions, such as the regularized Stokeslets [11].

2.4 Rectangular parallelepiped meshes of arbitrary size

The representation in (2.1) can be modified for a rectangular parallelepiped mesh element of arbitrary size and location, with sides parallel to the x, y and z axis. The modification [6,23] consists of mapping the mesh to a unit cube $[0,1]^3$. The map shifts and scales each variable accordingly such that

$$g(x,y,z) = \sum_{i,j,k=0}^{3} a_{ijk} \left(\frac{x-x_0}{\Delta x}\right)^i \left(\frac{y-y_0}{\Delta y}\right)^j \left(\frac{z-z_0}{\Delta z}\right)^k, \tag{2.11}$$

where $\Delta x, \Delta y, \Delta z$ are the lengths of the element in the three dimensions, and $\mathbf{x}_0 = (x_0, y_0, z_0)$ is the lower left corner of the element. Evaluating the polynomial in (2.11) and the first derivatives at $\mathbf{x}_0$ for example, we get

$$g|_{\mathbf{x}_0} = a_{000}, \qquad \frac{\partial g}{\partial x}\bigg|_{\mathbf{x}_0} = \frac{a_{100}}{\Delta x}, \qquad \frac{\partial g}{\partial y}\bigg|_{\mathbf{x}_0} = \frac{a_{010}}{\Delta y}, \qquad \frac{\partial g}{\partial z}\bigg|_{\mathbf{x}_0} = \frac{a_{001}}{\Delta z}. \tag{2.12}$$

Consequently, the evaluation of coefficients still follows (2.7) with the same B matrix, but the right hand sides in (2.8), (2.9), and (2.10), are evaluated with the derivatives appropriately scaled, as

$$S_1^* : \left\{ \Delta x \frac{\partial f}{\partial x}, \Delta y \frac{\partial f}{\partial y}, \Delta z \frac{\partial f}{\partial z} \right\}, \tag{2.13}$$

$$S_2^* : \left\{ \Delta x \Delta y \frac{\partial^2 f}{\partial x \partial y}, \Delta x \Delta z \frac{\partial f^2}{\partial x \partial z}, \Delta y \Delta z \frac{\partial f^2}{\partial y \partial z} \right\}, \qquad S_3^* : \left\{ \Delta x \Delta y \Delta z \frac{\partial^3 f}{\partial x \partial y \partial z} \right\}. \tag{2.14}$$

2.5 A test of continuity in tricubic interpolants

We perform a simple numerical test of continuity of each interpolant across one of the faces of the unit cube. We compute the potential due to an interaction between a single target particle and a cluster C of source particles,

$$\phi(\mathbf{x}_m, C) = \sum_{\mathbf{y}_n \in C} \mathcal{K}(\mathbf{x}_m, \mathbf{y}_n) \mathbf{f}_n, \tag{2.15}$$

where

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) = \frac{1}{|\mathbf{x} - \mathbf{y}|} \tag{2.16}$$

is the Coulomb kernel. We assume a target particle $\mathbf{x}_m = (2,2,2)$ and a cluster C of $N_c = 10$ source particles, where we compare the following two clusters: Cluster 1: $\mathbf{y}_1 = (0.5, 0.5, 0)$, $\{\mathbf{y}_2, \ldots \mathbf{y}_{N_c}\} \in (0,1)^3$, and Cluster 2: Reflection of Cluster 1 through the z=0 plane. This choice tests the continuity across the face of the cube in the z-plane. The weights of the source particles were chosen as $\mathbf{f}_1 = 1$ and $\mathbf{f}_n = 0$ for $n = 2, \ldots, N_c$.

Continuity	Cluster	ϕ	$\frac{\partial \phi}{\partial x}$	$\frac{\partial \phi}{\partial y}$	$\frac{\partial \phi}{\partial z}$
Exact		0.342997170285018	-0.060528912403238	-0.060528912403238	-0.080705216537651
$\mathcal{C}^1$	1	0.343088174174915	-0.060519566734223	-0.060622219592429	-0.080865234360966
	2	0.343088174174914	-0.060519566734222	-0.060622219592428	-0.080865234360965
$\mathcal{C}^0$	1	0.343088174174915	-0.060519566734223	-0.060622219592429	-0.080011060027459
	2	0.343088174174925	-0.060519566734250	-0.060622219592474	-0.081396487098385
Discont.	1	0.343141138245223	-0.060582125720232	-0.060779298363555	-0.079488441610077
	2	0.343045183232005	-0.060510460381771	-0.060558868061402	-0.081155418436247

Table 1: A test of continuity across the face of the unit cube in the z-plane of three tricubic interpolants.

Table 1 shows the values of the potential and the three components of the electric field. Using the $\mathcal{C}^1$-continuous tricubic, we obtain the same values, to within double-precision roundoff error, in all four quantities for the two Clusters (rows 3-4, highlighted in blue). Using the $\mathcal{C}^0$-continuous tricubic, we obtain the same values for the potential (rows 5-6, column 3, red) but different values for the z-derivative (rows 5-6, last column, magenta). And finally, using the discontinuous tricubic, we obtain different values for the two Clusters not only in the field but in the potential itself (rows 7-8, column 3, magenta).

3 The tricubic treecode

In our previous work [6], we developed a treecode algorithm based on the tricubic interpolant of Lekien and Marsden [23]. The treecode based on the two new tricubic interpolations is similar. For completeness, we summarize the algorithm here. 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 $\mathbf{x}_m$ interacting with 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.

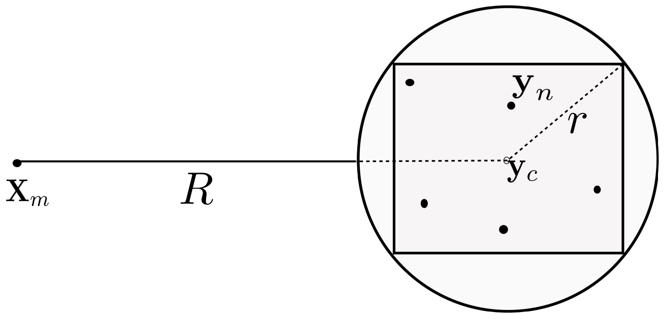

Figure 2: Particle-cluster interaction. The target particle is at position $\mathbf{x}_m$ and the source particles are at positions $\mathbf{y}_n$ in cluster C. Cluster C has center $\mathbf{y}_c$ and radius r. The particle-cluster distance is $R=|\mathbf{x}_m-\mathbf{y}_c|$.

The component of the sum (1.1) for this interaction is written as

$$\phi(\mathbf{x}_m,C)=\sum_{\mathbf{y}_n\in C}\mathcal{K}(\mathbf{x}_m,\mathbf{y}_n)f_n, \tag{3.1}$$

where $\mathbf{y}_n=(x_n,y_n,z_n)$. If the particle and the cluster are far enough apart (that is $\frac{r}{R}\le\theta$ [2], where $0\le\theta<1$), the sum in (3.1) is approximated in the following way. First, the cluster is shifted and scaled to the unit cube $[0,1]^3$,

$$x=\frac{x_n-x_{\min}}{\Delta x},\quad y=\frac{y_n-y_{\min}}{\Delta y},\quad z=\frac{z_n-z_{\min}}{\Delta z}, \tag{3.2}$$

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 $\mathcal{K}(\mathbf{x}_m,\mathbf{y}_n)$ is interpolated in the second (source) variable using the tricubic formula (2.1),

$$\phi(\mathbf{x}_m,C)=\sum_{\mathbf{y}_n\in C}\mathcal{K}(\mathbf{x}_m,\mathbf{y}_n)f_n\approx\sum_{\mathbf{y}_n\in C}\sum_{i,j,k=0}^{3}a_{ijk}x^iy^jz^kf_n. \tag{3.3}$$

Since the tricubic coefficients a_{ijk} do not depend on the individual source particles in the cluster, we switch the order of summation in (3.3), and use the definitions in equations (2.2) and (2.3), to obtain the far-field approximation,

$$\begin{aligned} \phi(\mathbf{x}_m,C) &\approx \sum_{i,j,k=0}^{3} a_{ijk} \sum_{\mathbf{y}_n\in C} x^i y^j z^k f_n, \\ &= \boldsymbol{\alpha}_m^T \boldsymbol{\mu}^C, \end{aligned} \tag{3.4}$$

where

$$\mu^C_{1+i+4j+16k} = \sum_{\mathbf{y}_n\in C} \mu_{1+i+4j+16k} f_n \tag{3.5}$$

are the monomials of the cluster C. The significance of approximation (3.4) 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 (2.7) to rewrite (3.4) as

$$\phi(\mathbf{x}_m,C) \approx \boldsymbol{\alpha}_m^T \boldsymbol{\mu}^C = (B^{-1}\mathbf{b}_m)^T \boldsymbol{\mu}^C = \mathbf{b}_m^T (B^{-1})^T \boldsymbol{\mu}^C = \mathbf{b}_m^T \mathbf{M}^C, \tag{3.6}$$

where

$$\mathbf{M}^C = (B^{-1})^T \boldsymbol{\mu}^C, \tag{3.7}$$

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 (3.7), since the 64×64 matrix $(B^{-1})^T$ is known explicitly. Furthermore, the matrix multiplication in (3.7) can be done in-line since the matrix is sparse. These monomials $\mathbf{M}^C$ can then be reused for different targets. Equation (3.6) 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 (3.6) is performed in two steps: first, the 64 elements of $\mathbf{b}_m$ are computed using (2.8), (2.9), or (2.10), scaling the derivatives as in (2.13)-(2.14), and second, the dot product of $\mathbf{b}_m$ and $\mathbf{M}^C$ is computed in (3.6). As described in [6], the cost of evaluating the particle-cluster interaction using (3.6) for N targets is estimated as $O(N\log N)$, consistent with other treecodes.

3.2 Computing derivatives of ϕ (3.1)

One of the advantages of the far-field approximation (3.6) that uses the tricubic interpolant is that the derivatives of ϕ can be easily computed analytically rather than numerically. We summarize the computation from [6].

The electric field at target position $\mathbf{x}_m$ due to the source cluster C is given by

$$\mathbf{E}_m = -\nabla_{\mathbf{x}_m} \phi(\mathbf{x}_m,C) = \nabla_{\mathbf{y}_n} \phi(\mathbf{x}_m,C), \tag{3.8}$$

since the kernel $\mathcal{K}$ is a function of $|\mathbf{x}-\mathbf{y}|$. From the tricubic approximation of the potential given in (3.6), the field is approximated as

$$\begin{aligned} \mathbf{E}_m &\approx \mathbf{b}_m^T \nabla_{\mathbf{y}_n} \mathbf{M}^C \\ &= \mathbf{b}_m^T \begin{bmatrix} \mathbf{M}^{C,i} \\ \mathbf{M}^{C,j} \\ \mathbf{M}^{C,k} \end{bmatrix}, \end{aligned} \tag{3.9}$$

since $\mathbf{b}_m$ is independent of $\mathbf{y}_n$. We compute the derivatives of the modified monomials as

$$\mathbf{M}^{C,i} := \frac{\partial \mathbf{M}^C}{\partial x_n} = (B^{-1})^T \frac{\partial}{\partial x_n}\left(\boldsymbol{\mu}^C\right) = (B^{-1})^T \boldsymbol{\mu}^{C,i}, \tag{3.10}$$

$$\mathbf{M}^{C,j} := \frac{\partial \mathbf{M}^C}{\partial y_n} = (B^{-1})^T \boldsymbol{\mu}^{C,j}, \tag{3.11}$$

$$\mathbf{M}^{C,k} := \frac{\partial \mathbf{M}^C}{\partial z_n} = (B^{-1})^T \boldsymbol{\mu}^{C,k}. \tag{3.12}$$

The derivatives of $\boldsymbol{\mu}^C$ are computed element-wise as

$$\mu^{C,i} := \frac{\partial}{\partial x_n}\left(\mu^C_{1+i+4j+16k}\right) = \frac{i}{\Delta x}\mu^C_{1+(i-1)+4j+16k}, \tag{3.13}$$

$$\mu^{C,j} := \frac{\partial}{\partial y_n}\left(\mu^C_{1+i+4j+16k}\right) = \frac{j}{\Delta y}\mu^C_{1+i+4(j-1)+16k}, \tag{3.14}$$

$$\mu^{C,k} := \frac{\partial}{\partial z_n}\left(\mu^C_{1+i+4j+16k}\right) = \frac{k}{\Delta z}\mu^C_{1+i+4j+16(k-1)}. \tag{3.15}$$

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 The particle-cluster algorithm

We now give an overview of the treecode algorithm that uses the far-field approximation (3.6) for ϕ and (3.9) for the derivatives of ϕ. The pseudocode is given in Algorithm 1, and it is similar to other treecodes [4,30]. First, the target particle coordinates $\{\mathbf{x}_m\}$, the source particle coordinates $\{\mathbf{y}_n\}$ and weights $\{f_n\}$ (weights are either one or three-dimensional) are provided as input. Then, the source particles are divided recursively into clusters to generate a tree structure as follows. The root cluster is taken as 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 $\mathbf{M}^C$,$\mathbf{M}^{C,i}$,$\mathbf{M}^{C,j}$,$\mathbf{M}^{C,k}$ are computed using equations (3.7), (3.10), (3.11), and (3.12). This concludes the precomputation needed at the start of the algorithm. The algorithm then loops through the target particles. For each target particle, the treecode algorithm cycles through the clusters in the tree recursively. The particle and the cluster are considered well-separated when the maximum acceptance criterion (MAC) is satisfied,

$$\frac{r}{R} \leq \theta, \tag{3.16}$$

where r is the cluster radius, R is the particle-cluster distance, and θ is a user-specified parameter. In this case, the particle-cluster interaction is computed using the far-field approximations (3.6), (3.9), where $\mathbf{b}_m$ is evaluated using (2.8), (2.9), or (2.10), and the modified monomials $\mathbf{M}^C$,$\mathbf{M}^{C,i}$,$\mathbf{M}^{C,j}$,$\mathbf{M}^{C,k}$ are simply looked up from the precomputation. 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 using (3.1).

Algorithm 1 tricubic treecode

```
1: input: target particle coordinates $\mathbf{x}_m, m = 1,\ldots,M$
2: input: source particle coordinates and weights $\mathbf{y}_n, f_n, n = 1,\ldots,N$
3: input: MAC parameter $\theta$, maximum leaf size $N_0$
4: output: potential $\phi_m$, electric field $\mathbf{E}_m$, $m = 1,\ldots,M$
5: program **main**
6:   build tree of source particle clusters
7:   for each cluster, precompute and store $\mathbf{M}^C, \mathbf{M}^{C,i}, \mathbf{M}^{C,j}, \mathbf{M}^{C,k}$ using (3.7), (3.10), (3.11), and (3.12)
8:   for $m = 1,\ldots,M$, **compute_potential**($\mathbf{x}_m$, root), end for
9: end program
10: subroutine **compute_potential**($\mathbf{x}$, $C$)
11:   if MAC (3.16) is satisfied
12:     compute $\mathbf{b}_m$ using (2.8), (2.9), or (2.10)
13:     compute particle-cluster interaction by approximations (3.6) and (3.9)
14:   else
15:     if $C$ is a leaf, compute particle-cluster interaction by direct sum (3.1)
16:   else
17:     for each child $C'$ of $C$, **compute_potential**($\mathbf{x}$, $C'$), end for
18: end subroutine
```

4 Numerical results

4.1 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 [7, 8]. The tests presented here were performed on a Dell PowerEdge R940xa Linux box with 2.1GHz Intel Xeon Gold processors.

4.2 Coulomb kernel

In this section, we compare the accuracy of the three treecodes in approximating the Coulomb potential and electric field for systems of size $N \in \{10^4,\, 8\times 10^4,\, 64\times 10^4\}$ 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 $\theta = 0.3:0.1:0.8$.

We approximate the potential ϕ in (1.1) and the electric field $\nabla\phi$ using equations (3.6) for the potential and (3.9) for the electric field. We compute the relative error, in ℓ^2-norm, in the approximation of the potential, Error(ϕ), given by

$$\mathrm{Error}(\phi) = \Big(\sum_{m=1}^{N} \Big|\phi^d(\mathbf{x}_m) - \phi^t(\mathbf{x}_m)\Big|^2 \Big/ \sum_{m=1}^{N} \Big|\phi^d(\mathbf{x}_m)\Big|^2 \Big)^{1/2}, \tag{4.1}$$

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

$$\mathrm{Error}(\nabla\phi) = \Big(\sum_{m=1}^{N} \Big|\nabla\phi^d(\mathbf{x}_m) - \nabla\phi^t(\mathbf{x}_m)\Big|^2 \Big/ \sum_{m=1}^{N} \Big|\nabla\phi^d(\mathbf{x}_m)\Big|^2 \Big)^{1/2}, \tag{4.2}$$

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

Figure 3 provides a graphical comparison of the accuracy of the three methods. The top row (a-c) is a plot of the relative error in potential, $\text{Error}(\phi)$, versus the MAC parameter θ. The middle row (e-f) is a plot of the relative error in the electric field, $\text{Error}(\nabla\phi)$ against θ. The bottom row (g-h) is a plot of $\text{Error}(\nabla\phi)$ versus $\text{Error}(\phi)$. There are two plots for the discontinuous tricubic treecode: One plot labeled $\texttt{Discontinuous}_{\text{A}}$ for which the treecode used analytical derivatives to compute the components of $\mathbf{b}$ in equation (2.10) and the other labeled $\texttt{Discontinuous}_{\text{FD}}$ where the treecode used a centered difference to approximate $\mathbf{b}$.

All three methods interpolate the potential at the vertices of the mesh, but the $\mathcal{C}^0$ continuous method has one additional interpolation point in the center of the mesh and the discontinuous method has eight additional interpolation points inside the mesh. As such, the polynomial for the discontinuous method will provide the best match for the potential inside the mesh, followed by the $\mathcal{C}^0$ continuous method and then the $\mathcal{C}^1$ continuous method. The top row of Figure 3 provides support for this conclusion. For all system sizes, the discontinuous method is the most accurate in computing the potential followed by the $\mathcal{C}^0$ continuous method and then the $\mathcal{C}^1$ continuous method.

The middle and bottom row of Figure 3 show the effect of the varying smoothness of the methods. We see in the middle row that even though the $\mathcal{C}^1$ continuous method was the least accurate in approximating the potential, this is not the case in approximating the electric field. In Figure 3(d) and Figure 3(e), the $\mathcal{C}^1$ continuous method is the most accurate in approximating the electric field. In Figure 3(f), the three methods have similar accuracy. The regularity of the electric field is one less than that of the potential, as such for the same accuracy in the potential, the higher smoothness of the $\mathcal{C}^1$ continuous method yields a better approximation than the other two methods. The bottom row of Figure 3 provides more evidence of the effect of smoothness on the accuracy of the electric field. The plots show that for a fixed accuracy in the potential, the $\mathcal{C}^1$ continuous method has the best accuracy in the electric field, followed by the $\mathcal{C}^0$ continuous method and finally the discontinuous method.

Figure 4 shows plots of the CPU time against θ for the Coulomb kernel. All three treecodes with analytical derivatives are similar in terms of efficiency. From the slopes of the curves, we see that the CPU time scales $\sim \theta^{-2.53}$. Even with the higher computational cost from smaller MAC parameters, the treecode still provides some efficiency over direct sum. The discontinuous treecode that uses finite difference derivatives is slower than the version that employs analytical derivatives, although both versions have similar accuracy. We will see in the next section, that the difference in efficiency narrows for the more complicated regularized Stokeslet. The more complex regularized Stokeslet kernel is a better use case than the relatively simple Coulomb kernel for the version of the discontinuous method that uses finite difference.

Figure 5 shows plots of the CPU time for the analytical treecodes and the direct sum against N. The figure shows the $O(N\log N)$ scaling of the three treecodes and the $O(N^2)$ scaling of direct sum. The discontinuous treecode uses analytical derivatives in this case.

4.3 Regularized Stokeslet kernel

Here, we examine the effect of smoothness on the accuracy of the treecodes for the more complicated regularized Stokeslet kernel. The method of regularized Stokeslets (MRS) [11] is a Lagrangian method for computing Stokes flow where the point forces are given by a cutoff function. The method has been used extensively in modeling various biological phenomena. The velocity $\mathbf{u}$

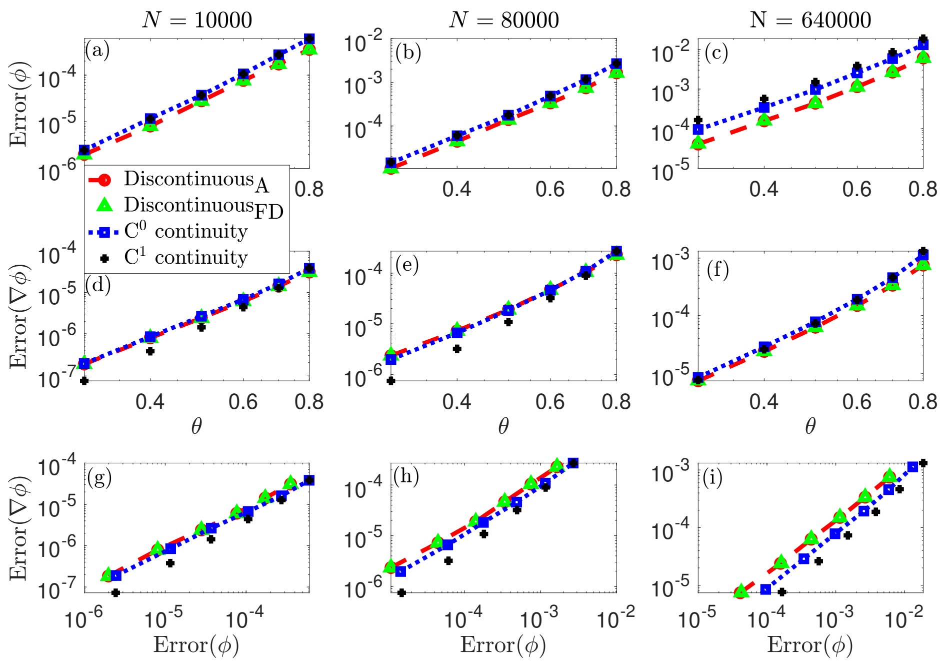

Figure 3: Accuracy in θ of the tricubic treecodes for the Coulomb kernel. N is the number of particles, θ is the MAC parameter.

at a point $\mathbf{x}$ is given by

$$\mathbf{u}(\mathbf{x}) = \sum_{n=1}^{N} \left(\mathbf{f}_n H_1(r_n) + [\mathbf{f}_n \cdot (\mathbf{x} - \mathbf{y}_n)] (\mathbf{x} - \mathbf{y}_n) H_2(r_n) \right), \tag{4.3}$$

where forces $\{\mathbf{f}_n\}$, $n = 1{:}N$, are located at positions $\{\mathbf{y}_n\}$, and $r_n = |\mathbf{x} - \mathbf{y}_n|$. The radial functions H_1 and H_2 are defined by

$$H_1(r) = \frac{2\epsilon^2 + r^2}{8\pi(r^2 + \epsilon^2)^{3/2}}, \quad H_2(r) = \frac{1}{8\pi(r^2 + \epsilon^2)^{3/2}}, \tag{4.4}$$

where $\epsilon > 0$ is the regularization parameter. In this work, $\epsilon = 0.02$. One notable feature of the MRS is that it is analytically divergence-free, i.e., $\nabla \cdot \mathbf{u} = 0$ for any $\epsilon \neq 0$, and we are interested in exploring treecode algorithms that preserve this feature to a larger degree.

The complexity of the MRS kernels makes them good candidates for treecodes where only kernel evaluations are needed, such as the kernel-independent treecode (KITC), but we know of only three recent works where fast summation methods were used to evaluate these kernels. Two recent applications [22], [26] used either the original kernel-independent fast multipole method (KIFMM) [32] or a later version in which the equivalent densities are defined on coronas (or volumetric shells) around each cluster [33]. The regularized Stokeslets were also evaluated in the kernel-independent treecode based on barycentric Lagrange interpolation [30]. Simulations of particle suspensions in viscous flows [31] demonstrate that numerical methods based on divergence-free interpolation yield particle distributions that are more similar to the exact solution, even while having higher error per particle. Our aim here is to demonstrate the effect of smoothness on the performance of the tricubic treecode not only in evaluating the velocity (4.3), but also in preserving

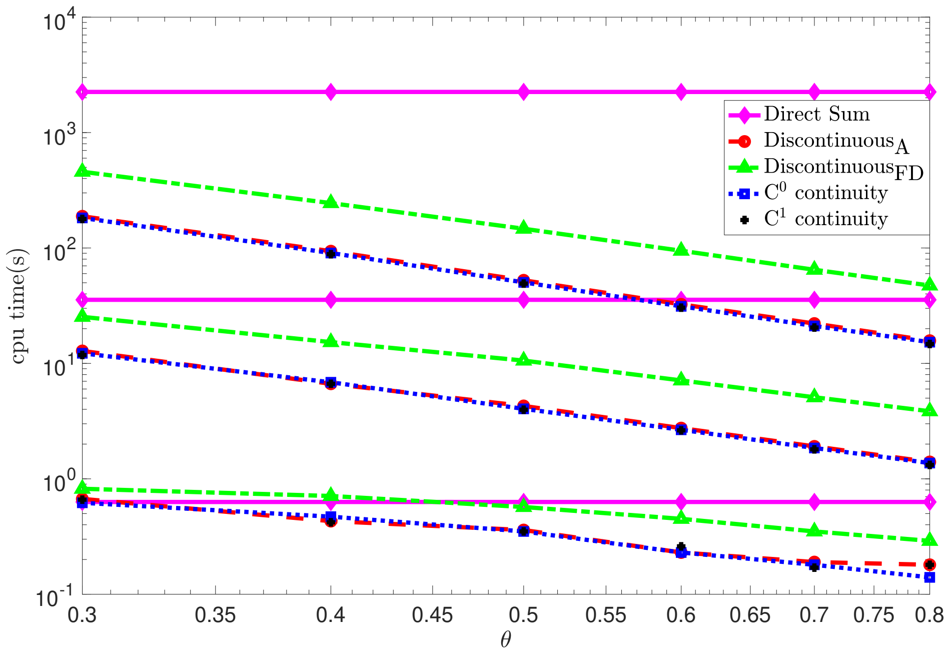

Figure 4: CPU time vs θ for the three tricubic treecode methods for the Coulomb kernel.

the divergence-free property of the MRS. In Appendix A, we derive the particle-cluster approximations for the velocity and the divergence similar to the approximations for scalar kernels given in equations (3.6) and (3.9). We also provide formulas for analytical derivatives of the MRS kernel.

We define the velocity error as

$$\text{Error}(\mathbf{u}) = \left(\sum_{m=1}^{N} |\mathbf{u}^d(\mathbf{x}_m) - \mathbf{u}^t(\mathbf{x}_m)|^2 \Big/ \sum_{m=1}^{N} |\mathbf{u}^d(\mathbf{x}_m)|^2 \right)^{1/2}, \tag{4.5}$$

where $\mathbf{u}^d$ is the exact velocity computed by direct summation, $\mathbf{u}^t$ is the treecode approximation, and $|\mathbf{u}|$ is the Euclidean norm. The error in divergence is defined as

$$\text{Error}(\nabla\cdot\mathbf{u}) = \frac{1}{N} \sum_{m=1}^{N} |(\nabla\cdot\mathbf{u})^t(\mathbf{x}_m)|, \tag{4.6}$$

where $(\nabla\cdot\mathbf{u})^t$ is the treecode approximation of divergence and $|\cdot|$ is the absolute value.

The systems and test parameters used for the regularized Stokeslet are similar to those used for the Coulomb kernel in Section 4.2. The only difference is that the weights for the MRS kernels $\mathbf{f}_n = (f_{n,1}, f_{n,2}, f_{n,3})$ are vectors with $f_{n,s} \in (-1,1)$, for $s = 1,2,3$.

Figure 6 shows the accuracy of the three treecodes for $N = 10K, 80K, 640K$ particles. The top row (a-c) is a plot of the error in the approximation of the velocity, Error$(\mathbf{u})$, against the MAC parameter θ. Similar to the Coulomb kernel, we see that the discontinuous treecode, with more interpolation points inside the mesh, produces the best accuracy since the velocity does not involve any derivatives of the kernel. The $\mathcal{C}^1$ continuous treecode is the least accurate of the three since it has the fewest interpolation points. The middle row (d-f) plots the error in the divergence of the velocity, Error$(\nabla\cdot\mathbf{u})$ versus θ and the last row (g-i) plots Error$(\nabla\cdot\mathbf{u})$ against Error$(\mathbf{u})$. The parameter Error$(\nabla\cdot\mathbf{u})$ is a measure of how well a method preserves the divergence-free property

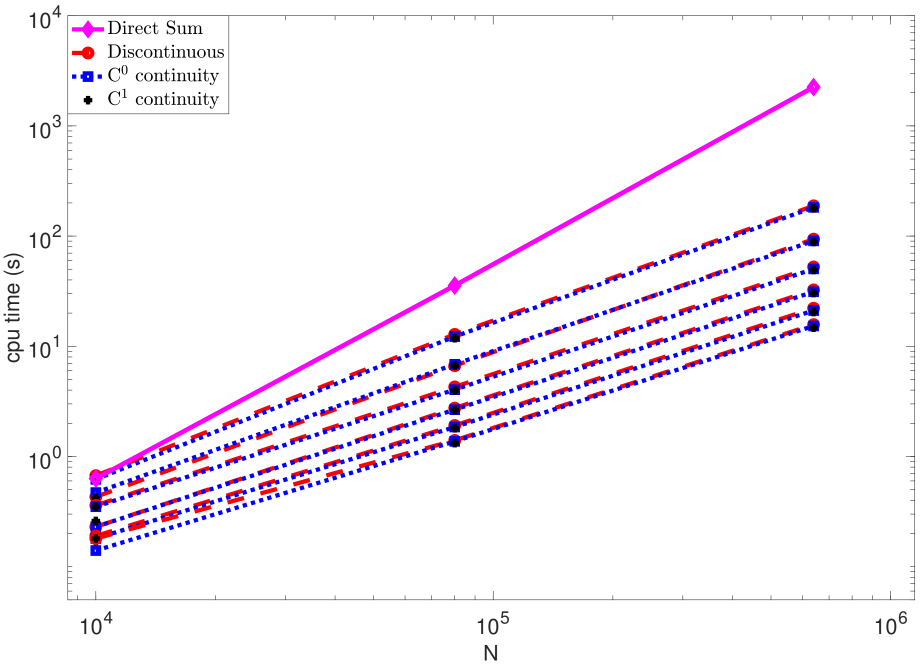

Figure 5: CPU vs N for the tricubic treecodes and direct sum. N is the number of particles, θ is the MAC parameter and the kernel is the Coulomb kernel.

of the MRS kernel. We see in the middle plots that because the divergence requires derivatives of the kernel, the $\mathcal{C}^1$ continuous treecode is the method that best preserves the divergence-free property for a fixed error in the velcotiy. The last row shows a clear separation of the three methods due to the differences in global smoothness. We see that for a fixed accuracy level in the velocity, a higher degree of smoothness leads to better preservation of the divergence-free property. This is similar to what we observed for the electric field for the Coulomb kernel in Section 4.2.

Figure 7 shows the performance of the treecodes in time for varying MAC parameter θ compared to direct sum. All three methods have identical computational performance for the analytical versions. We see that for the MRS kernel, the finite difference version of the discontinuous method is closer in computational time to the analytical version compared to the difference observed for the Coulomb kernel. The CPU time scales $\sim\theta^{-2.3}$ which is similar to the $\theta^{-2.5}$ scaling observed for the Coulomb potential. This suggests that the dependence of the CPU time on θ is largely independent of the kernel.

4.4 Molecular dynamics simulation

The results for the Coulomb and MRS kernels presented in the previous two sections were for a standalone computation. In this section we look at the effect of smoothness on dynamical properties of a liquid Argon system. The system is evolved in time as a canonical ensemble via molecular dynamics (MD) simulation on the Lennard-Jones potential energy surface

$$\psi(r)=4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12}-\left(\frac{\sigma}{r}\right)^{6}\right], \tag{4.7}$$

where $\epsilon=0.9661$ kJ/mol and $\sigma=3.405$Å. The system contained 100 Argon atoms in a cubic box of length 17.4Å with periodic boundary conditions and was equilibrated at 85^oK using an Evan

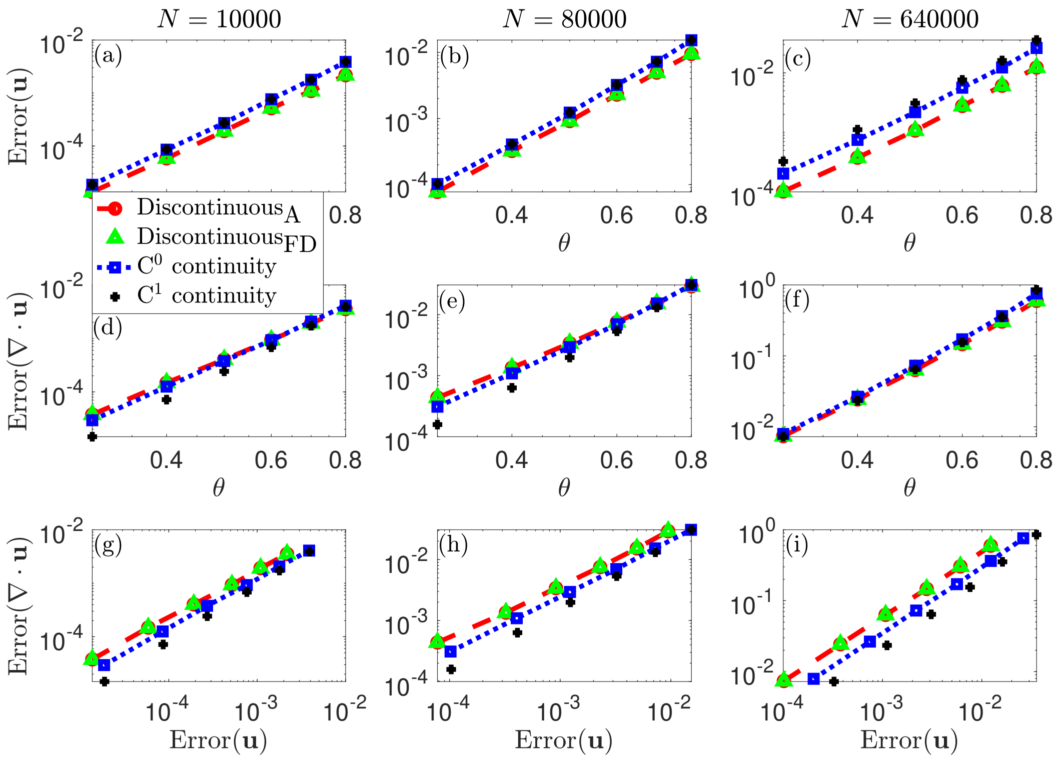

Figure 6: Accuracy in θ of the tricubic treecodes for the MRS kernel. N is the number of particles, θ is the MAC parameter.

thermostat [35]. The simulations were performed with the MD simulation package DL_POLY Classic [35]. We incorporated all three tricubic treecodes into DL_POLY Classic as options for computing the energy of the system and the forces on the atoms. In all the tests, the same initial configuration of the system was equilibrated over 5000 MD steps, then we took statistics over a further 20000 steps.

We studied the effect of smoothness of each method on the system's total energy fluctuation in time, its velocity-velocity auto-correlation function [34] and its temperature fluctuations over time. We performed simulations for three different timesteps, $\Delta t = 1$fs, $\Delta t = 5$fs and $\Delta t = 10$fs each with two different treecode MAC parameters $\theta = 0.5$ and $\theta = 0.7$ for a total of six MD simulations. The maximum number of particles in a leaf was set to $N_0 = 4$ for each of the treecodes. The MAC parameter controls the proportion of the treecode computations done with direct summation versus tricubic approximation. We chose relatively large MAC parameters of $\theta = 0.5$ and $\theta = 0.7$ to ensure the computations included a high proportion of tricubic approximations in order to be able to clearly observe the effects of smoothness. We also chose large timesteps to put the system in a regime of high energy and temperature fluctuations.

Figure 8 is a plot of the energy over the 20000 additional MD steps after equilibration. The energy does not depend on directly on derivatives of the kernel, as such we expect the discontinuous method, with the sixteen interpolation points, to be the most accurate and as a result have the least fluctuations in energy over time. The left column of Figure 8, panels (a), (c), (e), is a regime of relatively higher accuracy for the treecodes with $\theta = 0.5$, thus all three methods have similar energy fluctuations. The right-column, panels (b), (d), (f), with $\theta = 0.7$ is a less accurate regime. The treecode has a higher proportion of tricubic approximations in this regime compared to $\theta = 0.5$. As

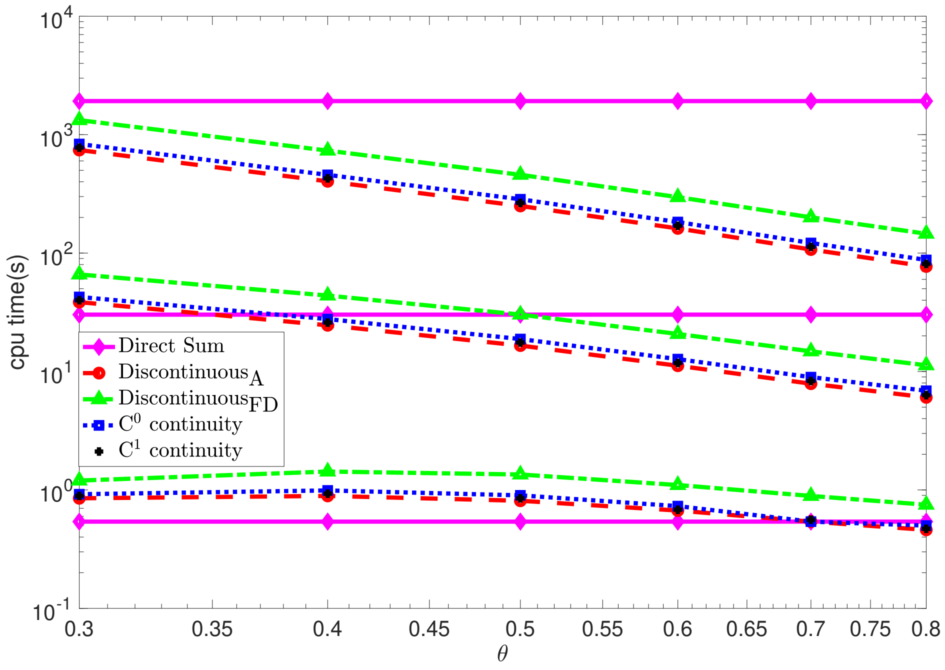

Figure 7: CPU time vs θ for the three tricubic treecode methods for the MRS kernel.

such the discontinuous and $\mathcal{C}^0$ methods, with more evaluation points, are more accurate than the $\mathcal{C}^1$ method. We see that the $\mathcal{C}^1$ continuous tricubic has the most fluctuations and the fluctuations increase as Δt increases. On the other hand, the energy for the $\mathcal{C}^0$ and discontinuous methods exhibit relatively little fluctuations compared to the $\mathcal{C}^1$ method, even with increasing Δt.

The velocity-velocity autocorrelation function $C_{VV}(t)$ is a function of the velocities of the atoms. The velocities are computed from an integration of the forces, $-\nabla\psi(r)$, hence $C_{VV}(t)$ depends on the derivatives of the Lennard-Jones kernel. Figure 9 is a plot of $C_{VV}(t)$ in time generated from simulations using the three treecode methods and direct sum with $\Delta t = 1$fs and $\theta = 0.5$ for the treecodes. The plots for the other five combinations of the timesteps and MAC parameter are similar.

Figure 10 compares the errors in the velocity-velocity autocorrelation function for the treecode methods. The error is computed by subtracting $C_{VV}(t)$ for the treecodes from that for direct sum. The plots show a clear dependence of $C_{VV}(t)$ on the smoothness of a treecode. The $\mathcal{C}^1$ treecode shows the least deviation from direct sum, followed by the $\mathcal{C}^0$ treecode and finally the discontinuous treecode. The middle row provides a very clear depiction of this trend.

Figure 11 is a plot of the fluctuations in instantaneous temperature over time for all six regimes. Given that atom i has velocity $\mathbf{v}_i$, the instantaneous temperature is obtained as

$$T(t) = \frac{\sum_{i=1}^{N} m_i v_i^2(t)}{\kappa_B(3N-3)}, \tag{4.8}$$

where m_i and $\mathbf{v}_i$ are the mass and velocity respectively of particle i, $v_i = |\mathbf{v}_i|$ and κ_B is Boltzmann's constant. Clearly $T(t)$ depends on the velocity and thus on derivatives of the kernel. The fluctuations are computed as $|85^o\text{K} - T(t)|$. For the regimes with $\Delta = 1$fs , panels (a) and (b), all three

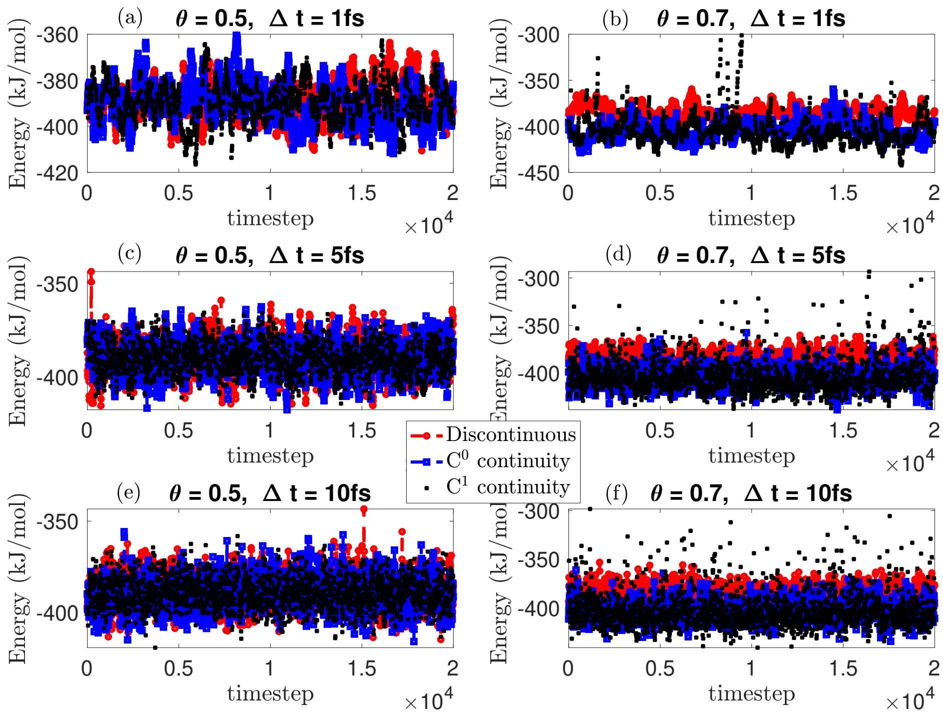

Figure 8: Fluctuations in energy for MAC parameters $\theta \in \{0.5, 0.7\}$ and time steps $\Delta t \in \{1\text{fs}, 5\text{fs}, 10\text{fs}\}$

methods have similar small temperature fluctuations. Panels (c)-(f) show that as the timestep increases, the discontinuous method produces the highest temperature fluctuations as expected. The magnitude of the fluctuations in the discontinuous method increase by 3 orders of magnitude with an order of magnitude increase in the timestep Δt.

For a system with N atoms, the treecode is an $O(N \log N)$ method and the typical radial cutoff approach for the Lennard-Jones potential is $O(N)$. Our choice of a Lennard-Jones liquid for the MD simulations was motivated by the need for a simple system to allow the effects of smoothness to be clearly observed. The tricubic treecode is most efficient in simulations with slowly decaying potentials. For example, it can be deployed to speed up computations of the Coulomb potential for electrostatic interactions in free-space or periodic boundary conditions [5] or the real space Ewald sum in periodic boundary conditions [9]. In MD simulations, typically the he short-range Lennard-Jones and long-range electrostatic interactions are computed separated. With the treecode, both computations can be combined in one routine for efficiency.

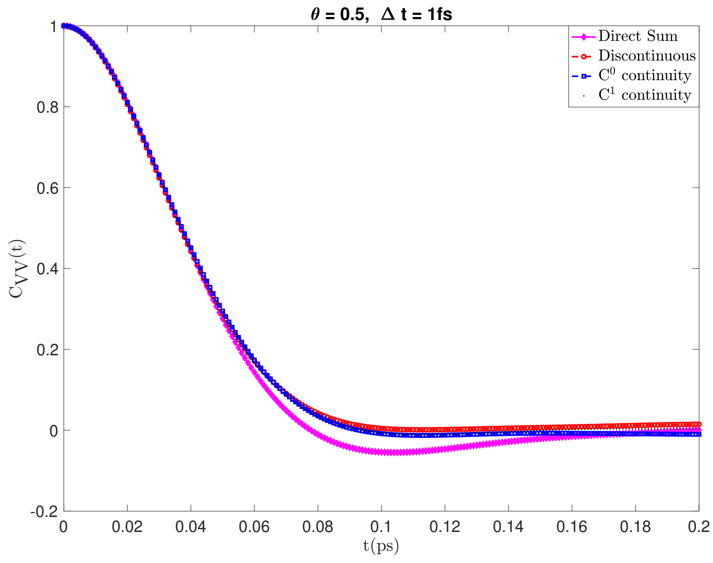

Figure 9: A sample velocity-velocity autocorrelation function $C_{VV}(t)$. Time step $\Delta t = 1$ fs and $\theta = 0.5$.

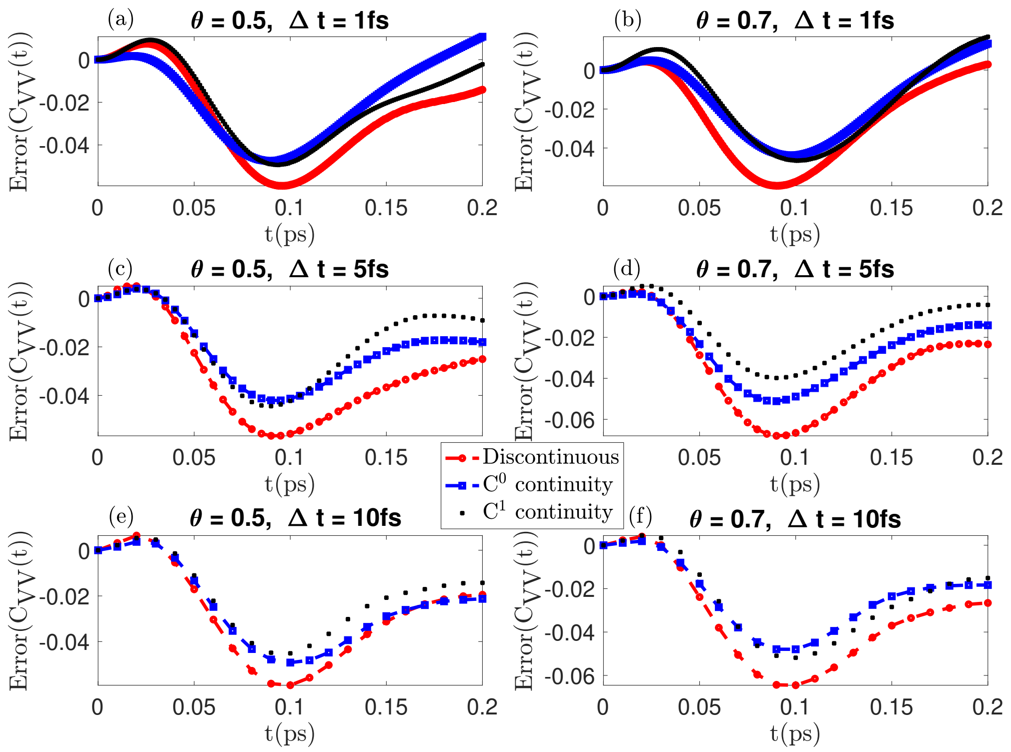

Figure 10: Errors in velocity-velocity autocorrelation function $C_{VV}(t)$ for MAC parameters $\theta \in \{0.5, 0.7\}$ and time steps $\Delta t \in \{1\text{fs}, 5\text{fs}, 10\text{fs}\}$.

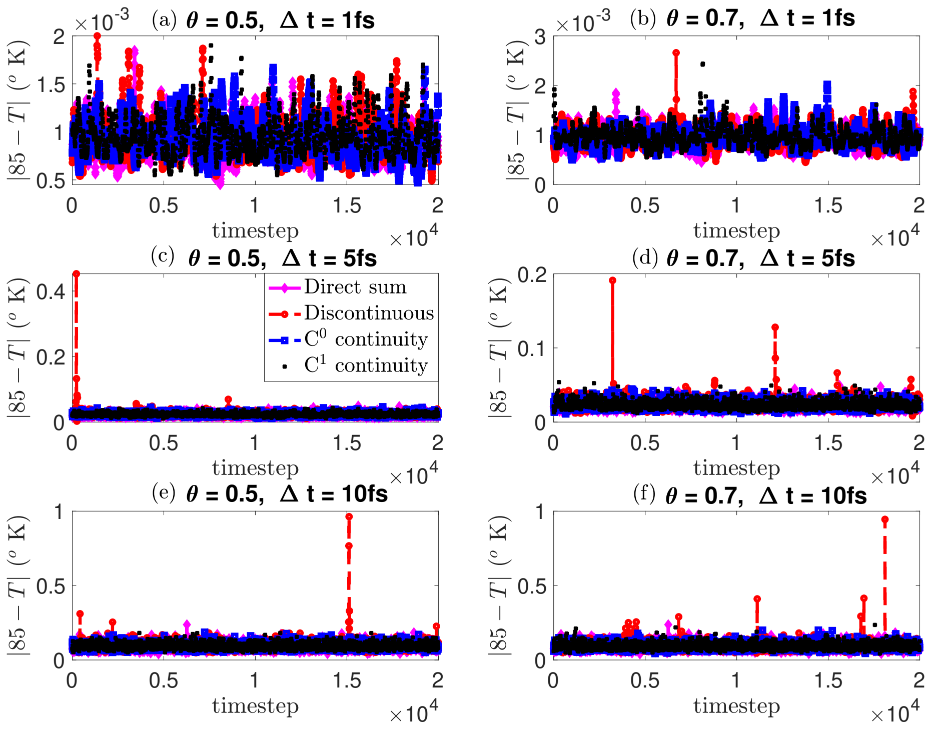

Figure 11: Fluctuations in temperature for MAC parameters $\theta \in \{0.5, 0.7\}$ and time steps $\Delta t \in \{1\text{fs}, 5\text{fs}, 10\text{fs}\}$.

5 Conclusions

This paper presented a comparative study of treecodes based on three tricubic interpolants of different smoothness: $\mathcal{C}^1$-continuous, $\mathcal{C}^0$-continuous, and discontinuous across the faces of tree cluster boxes. The comparisons presented in this paper indicate that when using polynomial approximation in a tree-based method, the global smoothness properties of the approximations may have an effect on the overall accuracy of the algorithm, especially when evaluating the derivatives of the N-sum. We have studied the Coulomb potential and its electric field, the velocity due to regularized Stokeslets and its divergence, and performed a simple molecular dynamics simulation of liquid Ar. Our results point to a slight increase in accuracy in the derivatives (electric field, divergence) relative to the accuracy in the quantity itself (potential, fluid velocity) when an interpolant of stronger smoothness is used. This suggests that the effects of $\mathcal{C}^1$ continuity in the interpolated kernel may be advantageous in simulations where preserving the accuracy of the derivative quantities in relation to the accuracy of the function is also sought, such as force and virial values in MD simulations and preserving the zero divergence in fluid dynamics. In computations where the derivatives quantities are not required, our results indicate that the discontinuous method, which has the most evaluation points of the three methods presented, provides the highest accuracy of the three methods.

Because the polynomial order of the tricubic treecodes is fixed at three, a small MAC parameter is usually required to achieve high accuracy for simulations with fixed particle density at more computational cost. Another way to achieve higher accuracy is by using higher order interpolants. The tricubic interpolation can only guarantee $\mathcal{C}^1$ continuity. In future work, we hope to improve on this work by using a higher order interpolating polynomial, such as a triquintic, to achieve higher accuracy and smoothness in the overall algorithm.

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.

A Derivations for regularized Stokeslet kernel

A.1 Particle-cluster approximation

Let $\mathbf{x}_m$ be a target point and $\{\mathbf{y}_n\}$ a set of source points in a cluster C. The cluster is shifted and scaled as in equation (3.2) and the target undergoes the same shift as explained in Section 3.1. The velocity $\mathbf{u}$ at the target point $\mathbf{x}_m$ due to the source points $\{\mathbf{y}_n\}$ in a cluster C is given by

$$\begin{aligned}\mathbf{u}(\mathbf{x}_m) &= \sum_{\mathbf{y}_n\in C} (\mathbf{f}_n H_1(r_n)+[\mathbf{f}_n\cdot(\mathbf{x}_m-\mathbf{y}_n)](\mathbf{x}_m-\mathbf{y}_n)H_2(r_n)),\\ &= \sum_{\mathbf{y}_j\in C} H_2(r_n)\begin{bmatrix} g(r_n)+x_n^2 & x_n y_n & x_n z_n \\ x_n y_n & g(r_n)+y_n^2 & y_n z_n \\ x_n z_n & y_n z_n & g(r_n)+z_n^2\end{bmatrix}\begin{bmatrix} f_{n,1}\\ f_{n,2}\\ f_{n,3}\end{bmatrix},\end{aligned} \tag{A.1}$$

where $H_2(r)$ is as defined in equation (4.4),

$$g(r)=2\epsilon^2+r^2, \tag{A.2}$$

$\mathbf{f}_n=(f_{n,1},f_{n,2},f_{n,3})$, $\mathbf{x}_m-\mathbf{y}_n=(x_n,y_n,z_n)$, $r_n=|\mathbf{x}_m-\mathbf{y}_n|$.

We can rewrite equation (A.1) as

$$\mathbf{u}(\mathbf{x}_m)=\sum_{\mathbf{y}_n\in C}\left\{\begin{bmatrix}\mathcal{K}_1(\mathbf{x}_m,\mathbf{y}_n)\\ \mathcal{K}_2(\mathbf{x}_m,\mathbf{y}_n)\\ \mathcal{K}_3(\mathbf{x}_m,\mathbf{y}_n)\end{bmatrix} f_{n,1}+\begin{bmatrix}\mathcal{K}_2(\mathbf{x}_m,\mathbf{y}_n)\\ \mathcal{K}_4(\mathbf{x}_m,\mathbf{y}_n)\\ \mathcal{K}_5(\mathbf{x}_m,\mathbf{y}_n)\end{bmatrix} f_{n,2}+\begin{bmatrix}\mathcal{K}_3(\mathbf{x}_m,\mathbf{y}_n)\\ \mathcal{K}_5(\mathbf{x}_m,\mathbf{y}_n)\\ \mathcal{K}_6(\mathbf{x}_m,\mathbf{y}_n)\end{bmatrix} f_{n,3}\right\}, \tag{A.3}$$

where the kernels $\mathcal{K}_1(\mathbf{x}_m,\mathbf{y}_n)=H_2(r_n)[g(r_n)+x_n^2]$, $\mathcal{K}_2(\mathbf{x}_m,\mathbf{y}_n)=H_2(r_n)x_n y_n$, $\mathcal{K}_3(\mathbf{x}_m,\mathbf{y}_n)=H_2(r_n)x_n z_n$, $\mathcal{K}_4(\mathbf{x}_m,\mathbf{y}_n)=H_2(r_n)[g(r_n)+y_n^2]$, $\mathcal{K}_5(\mathbf{x}_m,\mathbf{y}_n)=H_2(r_n)y_n z_n$ and $\mathcal{K}_6(\mathbf{x}_m,\mathbf{y}_n)=H_2(r_n)[g(r_n)+z_n^2]$. By following a similar procedure to that shown in Section 3.1, we find a particle-cluster approximation for each of the sums above as

$$\sum_{\mathbf{y}_n\in C}\mathcal{K}_\ell(\mathbf{x}_m,\mathbf{y}_n)f_{n,s}\approx \mathbf{b}_\ell^T\mathbf{M}_s^C, \tag{A.4}$$

where $\ell\in\{1,2,3,4,5,6\}$ and $s=1,2,3$. The modified monomials $\mathbf{M}_s^C$ depend only on the cluster while the vectors $\mathbf{b}_\ell$ depend on the kernel, thus

$$\mathbf{u}(\mathbf{x}_m)\approx\begin{bmatrix}\mathbf{b}_1^T\\ \\ \mathbf{b}_2^T\\ \\ \mathbf{b}_3^T\end{bmatrix}\mathbf{M}_1^C+\begin{bmatrix}\mathbf{b}_2^T\\ \\ \mathbf{b}_4^T\\ \\ \mathbf{b}_5^T\end{bmatrix}\mathbf{M}_2^C+\begin{bmatrix}\mathbf{b}_3^T\\ \\ \mathbf{b}_5^T\\ \\ \mathbf{b}_6^T\end{bmatrix}\mathbf{M}_3^C. \tag{A.5}$$

Using equations (3.10), (3.11), (3.12), the divergence is then computed as

$$\begin{aligned}\nabla\cdot\mathbf{u}(\mathbf{x}_m) &\approx \left(\mathbf{b}_1^T+\mathbf{b}_2^T+\mathbf{b}_3^T\right)\nabla\cdot\mathbf{M}_1^C+\left(\mathbf{b}_2^T+\mathbf{b}_4^T+\mathbf{b}_5^T\right)\nabla\cdot\mathbf{M}_1^C+\left(\mathbf{b}_3^T+\mathbf{b}_5^T+\mathbf{b}_6^T\right)\nabla\cdot\mathbf{M}_1^C,\\ &= \mathbf{b}_{123}^T\left(\mathbf{M}_1^{C,i}+\mathbf{M}_1^{C,j}+\mathbf{M}_1^{C,k}\right)+\mathbf{b}_{245}^T\left(\mathbf{M}_2^{C,i}+\mathbf{M}_2^{C,j}+\mathbf{M}_2^{C,k}\right)+\mathbf{b}_{356}^T\left(\mathbf{M}_3^{C,i}+\mathbf{M}_3^{C,j}+\mathbf{M}_3^{C,k}\right),\\ &= \mathbf{b}_{123}^T\mathbf{M}_1^{C,ijk}+\mathbf{b}_{245}^T\mathbf{M}_2^{C,ijk}+\mathbf{b}_{356}\mathbf{M}_3^{C,ijk},\end{aligned} \tag{A.6}$$

where $\mathbf{b}_{ijk}^T=\mathbf{b}_i^T+\mathbf{b}_j^T+\mathbf{b}_k^T$ and $\mathbf{M}_s^{C,ijk}=\mathbf{M}_s^{C,i}+\mathbf{M}_s^{C,j}+\mathbf{M}_s^{C,k}$.

A.2 Analytical derivatives of the regularized Stokeslet kernel

Analytical computation of the vectors $\mathbf{b}_\ell$ in equation (A.4) requires enforcing a subset of the equality constraints S_1, S_2 or S_3. Here we derive formulas for the constraints applied to the MRS kernel at the interpolation points.

We start by defining the symmetric matrix $\mathcal{S}=(\mathbf{x}-\mathbf{y}_n)(\mathbf{x}-\mathbf{y}_n)$. Then the formula for the velocity at a point $\mathbf{x}$ provided in equation (4.3) can be rewritten as

$$\mathbf{u}(\mathbf{x})=\sum_{n=1}^{N}\mathcal{K}(r_n)\mathbf{f}_n, \tag{A.7}$$

where $\mathcal{K}(r_n)=\Psi(r_n)H_2(r_n)$, $\Psi(r_n)=g(r_n)I+\mathcal{S}$ and I is the 3×3 identity matrix. Let $\mathbf{x}_m$ be a target point and $\mathbf{y}_c$ one of the interpolation points on or inside the cluster. Let $\mathbf{x}_m-\mathbf{y}_c=\langle x,y,z\rangle$ and $r=|\mathbf{x}_m-\mathbf{y}_c|$, then

$$\mathcal{S}=(\mathbf{x}_m-\mathbf{y}_c)(\mathbf{x}_m-\mathbf{y}_c)=\begin{bmatrix} x^2 & xy & xz \\ xy & y^2 & yz \\ xz & yz & z^2 \end{bmatrix}. \tag{A.8}$$

The kernel $\mathcal{K}$ and its derivatives provide the components of $\mathbf{b}_\ell$.

A.2.1 First partial derivatives

$$\frac{\partial\mathcal{K}}{\partial x_c}=\frac{\partial}{\partial x_c}\Psi(r)H_2(r)=\frac{\partial\Psi}{\partial x_c}H_2(r)+\Psi\frac{\partial H_2}{\partial x_c}. \tag{A.9}$$

We find

$$\frac{\partial H_2}{\partial x_c}=q(r)H_2(r)x, \tag{A.10}$$

with

$$q(r)=\frac{3}{r^2+\epsilon^2}, \tag{A.11}$$

and

$$\frac{\partial\Psi}{\partial x_c}=\frac{\partial}{\partial x_c}(g(r)I+\mathcal{S})=\frac{\partial g}{\partial x_c}I+\frac{\partial\mathcal{S}}{\partial x_c}, \tag{A.12}$$

with

$$\frac{\partial g}{\partial x}=-2x, \tag{A.13}$$

and

$$\frac{\partial\mathcal{S}}{\partial x}=-\begin{bmatrix} 2x & y & z \\ y & 0 & 0 \\ z & 0 & 0 \end{bmatrix}=\mathcal{S}_x. \tag{A.14}$$

Then,

$$\frac{\partial\Psi}{\partial x_c}=-2xI-\mathcal{S}_x=-\begin{bmatrix} 4x & y & z \\ y & 2x & 0 \\ z & 0 & 2x \end{bmatrix}=\mathcal{T}_x, \tag{A.15}$$

and thus,

$$\frac{\partial\mathcal{K}}{\partial x_c}=(\mathcal{T}_x+xq(r)\Psi(r))H_2(r). \tag{A.16}$$

Similarly,

$$\frac{\partial\mathcal{K}}{\partial y_c}=\left(\mathcal{T}_y+yq(r)\Psi(r)\right)H_2(r), \tag{A.17}$$

and

$$\frac{\partial\mathcal{K}}{\partial z_c}=(\mathcal{T}_z+zq(r)\Psi(r))H_2(r), \tag{A.18}$$

with

$$\mathcal{T}_y=-\begin{bmatrix} 2y & x & 0 \\ x & 4y & z \\ 0 & z & 2y \end{bmatrix}, \quad \text{and} \quad \mathcal{T}_z=-\begin{bmatrix} 2z & 0 & x \\ 0 & 2z & y \\ x & y & 4z \end{bmatrix}. \tag{A.19}$$

A.2.2 Second partial derivatives

$$\frac{\partial^2 \mathcal{K}}{\partial x_c \partial y_c} = H_2(r)\frac{\partial}{\partial x_c}\left(\mathcal{T}_y + yq(r)\Psi(r)\right) + \left(\mathcal{T}_y + yq(r)\Psi(r)\right)\frac{\partial}{\partial x_c}H_2(r), \tag{A.20}$$

and

$$\frac{\partial}{\partial x_c}\left(\mathcal{T}_y + yq(r)\Psi(r)\right) = \frac{\partial}{\partial x_c}\mathcal{T}_y + y\left(q(r)\frac{\partial \Psi}{\partial x_c} + \Psi(r)\frac{\partial q}{\partial x_c}\right). \tag{A.21}$$

Since

$$\frac{\partial q}{\partial x_c} = \frac{2}{3}x\left[q(r)\right]^2, \tag{A.22}$$

and

$$\frac{\partial}{\partial x_c}\mathcal{T}_y = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathcal{P}_{xy}, \tag{A.23}$$

we find

$$\frac{\partial^2 \mathcal{K}}{\partial x_c \partial y_c} = \left\{\mathcal{P}_{xy} + \left(x\mathcal{T}_y + y\mathcal{T}_x + \frac{5}{3}xyq(r)\Psi(r)\right)q(r)\right\}H_2(r). \tag{A.24}$$

Similarly,

$$\frac{\partial^2 \mathcal{K}}{\partial x_c \partial z_c} = \left\{\mathcal{P}_{xz} + \left(x\mathcal{T}_z + z\mathcal{T}_x + \frac{5}{3}xzq(r)\Psi(r)\right)q(r)\right\}H_2(r), \tag{A.25}$$

and

$$\frac{\partial^2 \mathcal{K}}{\partial y_c \partial z_c} = \left\{\mathcal{P}_{yz} + \left(y\mathcal{T}_z + z\mathcal{T}_y + \frac{5}{3}yzq(r)\Psi(r)\right)q(r)\right\}H_2(r), \tag{A.26}$$

with

$$\mathcal{P}_{xz} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \qquad \text{and} \qquad \mathcal{P}_{yz} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{A.27}$$

A.2.3 Third partial derivative

Let $\mathcal{G} = \mathcal{P}_{yz} + \left(y\mathcal{T}_z + z\mathcal{T}_y + \frac{5}{3}yzq(r)\Psi(r)\right)q(r) = \mathcal{P}_{yz} + \mathcal{J}_{yz}q(r)$, then

$$\frac{\partial^3 \mathcal{K}}{\partial x_c \partial y_c \partial z_c} = H_2(r)\frac{\partial}{\partial x_c}\mathcal{G} + \mathcal{G}\frac{\partial}{\partial x_c}H_2(r). \tag{A.28}$$

Consequently,

$$\frac{\partial}{\partial x_c}\mathcal{G} = \frac{\partial \mathcal{P}_{yz}}{\partial x_c} + q(r)\frac{\partial}{\partial x_c}\mathcal{J}_{yz} + \mathcal{J}_{yz}\frac{\partial q}{\partial x_c}, \tag{A.29}$$

with

$$\frac{\partial \mathcal{P}_{yz}}{\partial x_c} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{A.30}$$

and

$$\frac{\partial}{\partial x_c}\mathcal{J}_{yz} = y\frac{\partial \mathcal{T}_z}{\partial x_c} + z\frac{\partial \mathcal{T}_y}{\partial x_c} + \frac{5}{3}yz\left(q(r)\frac{\partial \Psi}{\partial x_c} + \Psi(r)\frac{\partial q}{\partial x_c}\right). \tag{A.31}$$

Since $\dfrac{\partial \mathcal{T}_y}{\partial x_c} = \mathcal{P}_{xy}$ and $\dfrac{\partial \mathcal{T}_z}{\partial x_c} = \mathcal{P}_{xz}$, we find

$$\frac{\partial^3 \mathcal{K}}{\partial x_c \partial y_c \partial z_c} = \left\{ x\mathcal{P}_{yz} + y\mathcal{P}_{xz} + z\mathcal{P}_{xy} + \frac{5}{3}q(r)\left(yz\mathcal{T}_x + xz\mathcal{T}_y + xy\mathcal{T}_z + \frac{7}{3}xyzq(r)\Psi(r) \right) \right\} q(r)H_2(r). \tag{A.32}$$

References

[1] 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.

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

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

[4] 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.

[5] H. A. Boateng, *Periodic Coulomb Tree Method: An Alternative to Parallel Particle Mesh Ewald*, J. Chem. Theory Comput., 16, 1 (2020), pp. 7–17.

[6] H. A. Boateng and S. Tlupova, *A treecode algorithm based on tricubic interpolation*, (2022), submitted.

[7] H. Boateng and S. Tlupova, *Comparison of tricubic treecodes for the Coulomb kernel*, https://github.com/haboateng/comparison-of-tricubic-treecodes-Coulomb-kernel, Last assessed: 08-10-2022

[8] H. Boateng and S. Tlupova, *Comparison of tricubic treecodes for the regularized Stokeslet kernel*, https://github.com/haboateng/comparison-of-tricubic-treecodes-regularized-stokeslet-kernel, Last assessed: 08-10-2022

[9] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee and L. G. Pedersen, *A smooth particle mesh Ewald method*, J. Chem. Phys., 103, 19 (1995), pp. 8577–8593.

[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] R. Cortez, L. Fauci and A. Medovikov, *The method of regularized Stokeslets in three dimensions: Analysis, validation, and application to helical swimming*, Phys. Fluids, 17 (2005), Article 031504.

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

[13] 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.

[14] 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.

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

[16] 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.

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

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

[19] 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.

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

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

[22] J. LaGrone, R. Cortez, W. Yan and L. Fauci, *Complex dynamics of long, flexible fibers in shear*, J. Non-Newton. Fluid Mech., 269 (2019), pp. 73–81.

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

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

[25] 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.

[26] M. W. Rostami and S. D. Olson, *Kernel-independent fast multipole method within the framework of regularized Stokeslets*, J. Fluid Struct., 67 (2016), pp. 60–84.

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

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

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

[30] 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.

[31] B. K. Tapley, H. I. Andersson, E. Celledoni and B. Owren, *Computational geometric methods for preferential clustering of particle suspensions*, J. Comput. Phys., 448(2022), 110725.

[32] 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.

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

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

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