Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

Towards an adaptive treecode for N-body problems

Benjamin W. Ong · Satyen Dhamankar

Received: date / Accepted: date

Abstract N-body problems are notoriously expensive to compute. For N bodies, evaluating a sum directly scales like $\mathcal{O}(N^2)$. A treecode approximation to the N-body problem is highly desirable because for a given level of accuracy, the computation scales instead like $\mathcal{O}(N \log N)$. A main component of the treecode approximation, is computing the Taylor coefficients and moments of a cluster–particle approximation. For the two-parameter family of regularized kernels previously introduced (Ong et al, 2017), computing the Taylor coefficients directly is algebraically messy and undesirable. This work derives two useful recurrence relationships for computing the Taylor coefficients. The treecode is implemented in Cartesian coordinates, and numerical results verify that the recurrence relationships facilitate computation of $G^{\epsilon,n}(\mathbf{x})$ and its derivatives.

Keywords Kernel Regularization · Singular Kernels · N-body systems · Fast Summation · Taylor Series

Mathematics Subject Classification (2000) 41A58, 41A63, 70F10

Benjamin W. Ong
Michigan Technological University Department of Mathematical Sciences
Tel.: +1-906-487-3367
Fax: +1-906-487-3133
E-mail: ongbw@mtu.edu

Satyen V. Dhamankar
Michigan Technological University Department of Mathematical Sciences
E-mail: sdhamank@mtu.edu

1 Introduction

The N-body dynamical system,

$$\ddot{\mathbf{x}}_j = -w_j \sum_{m=1}^{N} w_m \nabla G(\mathbf{x}_j - \mathbf{x}_m), \tag{1}$$

models many physical phenomena, such as vortex motion [8,12], planetary movement [1] and plasma kinetics [7]. In $\mathbb{R}^3$, the function $G(\mathbf{x})$ is the familiar fundamental solution to the Laplace operator,

$$G(\mathbf{x}) = \begin{cases} \dfrac{1}{4\pi \|\mathbf{x}\|_2}, & \mathbf{x} \neq \mathbf{0} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Recently, a two-parameter family of regularized kernels was introduced [10] to better control the error in the Hamiltonian. Instead of solving eq. (1), one solves the approximate equations

$$\ddot{\mathbf{x}}_j = -w_j \sum_{m=1}^{N} w_m \nabla G^{\epsilon,n}(\mathbf{x}_j - \mathbf{x}_m), \tag{3}$$

where the regularized kernels,

$$G^{\epsilon,n}(\mathbf{x}) = \frac{1}{4\pi} \sum_{\ell=0}^{n} \binom{-\frac{1}{2}}{\ell} \left(-\epsilon^2\right)^{\ell} \left(\|\mathbf{x}\|_2^2 + \epsilon^2\right)^{-1/2-\ell}, \tag{4}$$

are derived by truncating a Taylor expansions of the non-regularized kernel, $G(\mathbf{x})$, about $(\|\mathbf{x}|_2^2 + \epsilon^2)$. We seek to construct an adaptive treecode algorithm/software package that uses the two-parameter family of regularized kernels in place of the algebraic regularized kernels [9,3]. The adaptive treecode algorithm will provide the ability to control the Hamiltonian error and the modeling error when using these regularized kernels.

Although treecode algorithms have been demonstrated to be computationally efficient for high-resolution simulations [11], one needs to evaluate derivatives of the kernels, eqs. (2) and (4), in the treecode algorithm. Evaluating derivatives of the kernel, eq. (4) is algebraically messy; for $n = 0$, a recurrence relation can be used to compute the derivatives without complicated algebra [9]. An this work, we develop recurrence relations for computing derivatives of eq. (4) for any n.

The manuscript proceeds as follows. In section 2, we review the treecode algorithm, discussing the recurrence relations in section 3. Section 4 presents numerical experiments demonstrating that our recurrence relations indeed facilitate computation of $G^{\epsilon,n}(\mathbf{x})$ and its derivatives.

2 Fast Summation Treecode Algorithms

The N-body systems, eqs. (1) and (3), are computationally expensive to simulate. To compute the force on each particle (body), the contributions from the other $N-1$ particles results in an $\mathcal{O}(N)$ cost. Since there are N particles and the force on each particle is required, the computational cost for the simulation balloons to the $\mathcal{O}(N^2)$. There are several approaches to approximate the force on each particle. A common approach is the Particle-In-Cell (PIC) method [2], where the particles are mapped to a uniform grid, the potential is evaluated using the mapped particles, and the gradient of the potential is evaluated at each particle location. Unfortunately, the resolution of the simulation is restricted to the resolution of the underlying uniform grid. An alternative class of methods are so-called fast summation methods, which approximate the long-range interactions using clusters of particles and their moments. Perhaps most popular is the fast-multipole method [5], which uses high-order spherical harmonics expansions and a sophisticated evaluation procedure. This cluster–cluster approximation comes with a high computational cost for moving particles. An alternative fast-summation method is the Boundary Integral Treecode (BIT) algorithm. Treecode algorithms have been very successful in particle simulations, and there is ongoing interest to optimize their performance [6].

2.1 Cluster Approximation

We begin by considering a non-overlapping partition of particles into clusters, C_r. Then eq. (3) can be expressed as

$$\ddot{\mathbf{x}}_j = -w_j \sum_{m=1}^{N} w_m \nabla G^{\epsilon,n}(\mathbf{x}_j - \mathbf{x}_m) = -w_j \sum_{C_r} \nabla \phi(\mathbf{x}_j, C_r),$$

where

$$\phi(\mathbf{y}, C_r) = \sum_{\mathbf{x}_i \in C_r} w_i G^{\epsilon,n}(\mathbf{y} - \mathbf{x}_i). \tag{5}$$

The function $\phi(\mathbf{y}, C_r)$ can be interpreted as the potential at the point $\mathbf{y}$ due to particles in cluster C_r. Performing a Taylor expansion of the regularized Green's function about the cluster center $\mathbf{x}_{cr}$ gives

$$\begin{aligned}
\phi(\mathbf{y}, C_r) &\approx \sum_{\mathbf{x}_i \in C_r} \sum_{\|\mathbf{k}\|_1=0}^{p} \frac{1}{\mathbf{k}!} \partial_{\mathbf{x}}^{\mathbf{k}} G^{\epsilon,n}(\mathbf{y} - \mathbf{x}_{cr}) \left(w_i (\mathbf{x}_i - \mathbf{x}_{cr})^{\mathbf{k}} \right) \\
&= \sum_{\|\mathbf{k}\|_1=0}^{p} \frac{1}{\mathbf{k}!} \partial_{\mathbf{x}}^{\mathbf{k}} G^{\epsilon,n}(\mathbf{y} - \mathbf{x}_{cr}) \sum_{\mathbf{x}_i \in C_r} w_i (\mathbf{x}_i - \mathbf{x}_{cr})^{\mathbf{k}} \\
&= \sum_{\|\mathbf{k}\|_1=0}^{p} a^{\mathbf{k}}(\mathbf{y} - \mathbf{x}_{cr}) M^{\mathbf{k}}(C_r),
\end{aligned} \tag{6}$$

where p is the order of Taylor approximation, the term

$$a^{\mathbf{k}}(\mathbf{x}) = \frac{1}{\mathbf{k}!}\partial_{\mathbf{x}}^{\mathbf{k}} G^{\epsilon,n}(\mathbf{x}), \tag{7}$$

is the $\mathbf{k}$th Taylor coefficient of the Green's function, and

$$M^{\mathbf{k}}(C_r) = \sum_{\mathbf{x}_i \in C_r} w_i (\mathbf{x}_i - \mathbf{x}_{cr})^{\mathbf{k}},$$

is the $\mathbf{k}$th moment of cluster C_r. We have utilized multi-index notation in eq. (6). For $\mathbf{k} = (k_1, k_2, k_3)$ with $k_i \geq 0$, $\mathbf{k}! = k_1!\, k_2!\, k_3!$, $\partial_{\mathbf{x}}^{\mathbf{k}} = \partial_{x_1}^{k_1}\, \partial_{x_2}^{k_2}\, \partial_{x_3}^{k_3}$, and $\mathbf{x}^{\mathbf{k}} = x_1^{k_1}\, x_2^{k_2}\, x_3^{k_3}$.

2.2 Recurrsive Evaluation Strategy

Similar to the treecode algorithm described in [9], a tree is first constructed by adaptively shrinking cells at each step in the construction using a binary tree. The potential (or force) at a point $\mathbf{y}$ is then evaluated using a recursive divide-and-conquer strategy [4]. Specifically, one checks if $\phi(\mathbf{y}, C_r)$ can be evaluated using a cluster (Taylor) approximation, eq. (6). In our code and simulations, the multipole acceptance criterion (MAC) is used, i.e., the Taylor expansion is used only if $\frac{r}{R} \leq \theta$, where r is the radius of the cluster, R is the distance between the $\mathbf{y}$ and the cluster, and θ is a user-specified parameter. If the cluster approximation cannot be used, then the potential is computed by (recursively) taking a linear combination of potentials from children clusters if they exist, or resorting to direct summation, eq. (5) if no children nodes exist.

3 Taylor Coefficients

Evaluating derivatives of the regularized kernel, eq. (4), is algebraically messy. A recurrence relation was previously found [9] to facilitate computation of the derivatives of $G^{\epsilon,0}(\mathbf{x})$ without complicated algebra. In section 3.1, we review the main results for the previously found recurrence relations before discussing and proving our new recurrence relations in section 3.2.

3.1 Previous Work

In [9] the authors consider the regularized potential,

$$\phi(\mathbf{x}) = \frac{1}{4\pi} \frac{1}{\left(|\mathbf{x}|_2^2 + \epsilon^2\right)^{1/2}},$$

which is indeed $G^{\epsilon,0}(\mathbf{x})$. A pth-order Taylor approximation of $\phi(\mathbf{x})$ at $\mathbf{x} = \bar{\mathbf{x}}$ can be expressed as

$$\phi(\mathbf{x}) \approx \sum_{\|\mathbf{k}\|=0}^{p} T^{\mathbf{k}}(\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{k}},$$

where

$$T^{\mathbf{k}}(\bar{\mathbf{x}}) = \frac{1}{\mathbf{k}!}\partial_{\mathbf{x}}^{\mathbf{k}}\phi(\bar{\mathbf{x}})$$

is the $\mathbf{k}$th Taylor coefficient. A key result in the paper is the derivation of the recurrence relation,

$$\begin{aligned}\|\mathbf{k}\|_1\left(\|\mathbf{x}\|_2^2+\epsilon^2\right)T^{\mathbf{k}}(\mathbf{x}) + (2\|\mathbf{k}\|_1-1)\sum_{i=1}^{3} x_i\, T^{\mathbf{k}-\mathbf{e}_i}(\mathbf{x}) + \cdots \\ (\|\mathbf{k}\|_1-1)\sum_{i=1}^{3} T^{\mathbf{k}-2\mathbf{e}_i}(\mathbf{x}) = 0,\end{aligned} \tag{8}$$

where $\mathbf{e}_1, \mathbf{e}_2$ and $\mathbf{e}_3$ are the standard Cartesian basis vectors. We refer the reader to [13] for a derivation and proof of eq. (8).

3.2 New Recurrence Relations

We seek a recurrence relation to compute the Taylor coefficients, eq. (7). To simplify the algebra, we adopt the following notation for the regularized kernels, eq. (4). We let

$$G^{\epsilon,n}(\mathbf{x}) = \sum_{\ell=0}^{n} G_\ell^\epsilon(\mathbf{x}), \tag{9}$$

where

$$G_\ell^\epsilon(\mathbf{x}) = \frac{1}{4\pi}\binom{-\frac{1}{2}}{\ell}\left(-\epsilon^2\right)^\ell\left(\|\mathbf{x}\|_2^2+\epsilon^2\right)^{-1/2-\ell}. \tag{10}$$

Further, we denote

$$a_\ell^{\mathbf{k}}(\mathbf{x}) = \frac{1}{\mathbf{k}!}\partial_{\mathbf{x}}^{\mathbf{k}}\, G_\ell^\epsilon(\mathbf{x}). \tag{11}$$

Proposition 1 *Let $f(y_1, y_2, y_3) : \mathbb{R}^3 \to \mathbb{R}$. Then for any integer $m > 0$,*

$$\partial_{y_i}^m\left((y_1+y_2+y_3)\, f\right) = m\,\partial_{y_i}^{m-1} f + (y_1+y_2+y_3)\,\partial_{y_i}^m f \tag{12}$$

Proof We shall prove proposition 1 by induction. Let $m = 1$. Applying the product rule,

$$\partial_{y_1}\left(\left(y_1+y_2+y_3\right)f\right) = f + \left(y_1+y_2+y_3\right)\partial_{y_i} f,$$

which satisfies eq. (12). Suppose that eq. (12) is true for any m. Then, taking the derivative of both sides of eq. (12) with respect to y_i gives

$$\begin{aligned}
\partial_{y_i}\left(\partial_{y_i}^m\left(\left(y_1+y_2+y_3\right)f\right)\right) &= \partial_{y_i}\left(m\,\partial_{y_i}^{m-1} f + \left(y_1+y_2+y_3\right)\partial_{y_i}^m f\right) \\
\partial_{y_i}^{m+1}\left(\left(y_1+y_2+y_3\right)f\right) &= m\,\partial_{y_i}^m f + \partial_{y_i}^m f + \left(y_1+y_2+y_3\right)\partial_{y_i}^{m+1} f \\
&= \left(m+1\right)\partial_{y_i}^m f + \left(y_1+y_2+y_3\right)\partial_{y_i}^{m+1} f,
\end{aligned}$$

which completes the proof. □

Lemma 1 *For $\ell \geq 0$, the Taylor coefficients, $a_\ell^{\mathbf{k}}(\mathbf{x})$, satisfy*

$$\frac{\epsilon^2}{\left(2\ell+2\right)}\sum_{i=1}^{3}\left(k_i+1\right)a_\ell^{\mathbf{k}+\mathbf{e}_i}(\mathbf{x}) + \sum_{i=1}^{3}\left(a_{\ell+1}^{\mathbf{k}-\mathbf{e}_i}(\mathbf{x}) + x_i\,a_{\ell+1}^{\mathbf{k}}(\mathbf{x})\right) = 0,$$

where $a_{\ell+1}^{-1,k_2,k_3} = a_{\ell+1}^{k_1,-1,k_3} = a_{\ell+1}^{k_1,k_2,-1} := 0$.

Proof Define the coefficient,

$$q(\ell) := \frac{1}{4\pi}\binom{-\frac{1}{2}}{\ell}\left(-\epsilon^2\right)^\ell,$$

so that eq. (10) satisfies

$$G_\ell^\epsilon(\mathbf{x}) = q(\ell)\left(\|\mathbf{x}\|_2^2+\epsilon^2\right)^{-1/2-\ell}. \tag{13}$$

Taking the derivative of eq. (13) with respect to x_i gives

$$\begin{aligned}
\partial_{x_i} G_\ell^\epsilon(\mathbf{x}) &= x_i\left(-1-2\ell\right)q(\ell)\left(\|\mathbf{x}\|_2^2+\epsilon^2\right)^{-3/2-\ell} \\
&= x_i\left(-1-2\ell\right)\frac{q(\ell)}{q(\ell+1)}G_{\ell+1}^\epsilon(\mathbf{x}).
\end{aligned}$$

Hence,

$$\left(\partial_{x_1}G_\ell^\epsilon(\mathbf{x}) + \partial_{x_2}G_\ell^\epsilon(\mathbf{x}) + \partial_{x_3}G_\ell^\epsilon(\mathbf{x})\right) = \left(x_1+x_2+x_3\right)\left(-1-2\ell\right)\frac{q(\ell)}{q(\ell+1)}G_{\ell+1}^\epsilon(\mathbf{x}).$$

Applying $\partial_{x_1}^{k_1}$ and invoking proposition 1,

$$\begin{aligned}
&\partial_{x_1}^{k_1}\left(\partial_{x_1}G_\ell^\epsilon(\mathbf{x}) + \partial_{x_2}G_\ell^\epsilon(\mathbf{x}) + \partial_{x_3}G_\ell^\epsilon(\mathbf{x})\right) \\
&= \partial_{x_1}^{k_1+1}G_\ell^\epsilon(\mathbf{x}) + \partial_{x_1,x_2}^{k_1+1,1}G_\ell^\epsilon(\mathbf{x}) + \partial_{x_1,x_3}^{k_1,0,1}G_\ell^\epsilon(\mathbf{x}) \\
&= k_1(-1-2\ell)\frac{q(\ell)}{q(\ell+1)}\partial_{x_1}^{k_1-1}G_{\ell+1}^\epsilon(\mathbf{x}) + \left(x_1+x_2+x_3\right)\left(-1-2\ell\right)\frac{q(\ell)}{q(\ell+1)}\partial_{x_1}^{k_1}G_{\ell+1}^\epsilon(\mathbf{x}).
\end{aligned}$$

In the same fashion, applying $\partial_{x_2}^{k_2}$ and invoking proposition 1,

$$\begin{aligned}
&\partial_{x_2}^{k_2}\left(\partial_{x_1}^{k_1+1}G_\ell^\epsilon(\mathbf{x}) + \partial_{x_2}G_\ell^\epsilon(\mathbf{x}) + \partial_{x_3}G_\ell^\epsilon(\mathbf{x})\right)\\
&= \partial_{x_1,x_2}^{k_1+1,k_2}G_\ell^\epsilon(\mathbf{x}) + \partial_{x_1,x_2}^{k_1,k_2+1}G_\ell^\epsilon(\mathbf{x}) + \partial_{x_1,x_2,x_3}^{k_1,k_2,1}G_\ell^\epsilon(\mathbf{x})\\
&= (-1-2\ell)\,\frac{q(\ell)}{q(\ell+1)}\Big(k_1\partial_{x_1,x_2}^{k_1-1,k_2}G_{\ell+1}^\epsilon(\mathbf{x}) + k_2\partial_{x_1,x_2}^{k_1,k_2-1}G_{\ell+1}^\epsilon(\mathbf{x}) + \cdots\\
&\qquad (x_1+x_2+x_3)\;\partial_{x_1,x_2}^{k_1,k_2}G_{\ell+1}^\epsilon(\mathbf{x})\Big).
\end{aligned}$$

Finally, applying $\partial_{x_3}^{k_3}$ to both sides of the above equation and invoking proposition 1,

$$\begin{aligned}
&\partial_{\mathbf{x}}^{\mathbf{k}+\mathbf{e}_1}G_\ell^\epsilon(\mathbf{x}) + \partial_{\mathbf{x}}^{\mathbf{k}+\mathbf{e}_2}G_\ell^\epsilon(\mathbf{x}) + \partial_{\mathbf{x}}^{\mathbf{k}+\mathbf{e}_2}G_\ell^\epsilon(\mathbf{x})\\
&= (-2\ell-1)\frac{q(\ell)}{q(\ell+1)}\Big(k_1\partial_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1}G_{\ell+1}^\epsilon(\mathbf{x}) + k_2\partial_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_2}G_{\ell+1}^\epsilon(\mathbf{x}) + k_3\partial_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_3}G_{\ell+1}^\epsilon(\mathbf{x}) + \cdots\\
&\quad (x_1+x_2+x_3)\partial_{\mathbf{x}}^{\mathbf{k}}G_{\ell+1}^\epsilon(\mathbf{x})\Big).
\end{aligned}$$

Since

$$\frac{q(l+1)}{q(l)} = \frac{\epsilon^2(2l+1)}{2l+2},$$

we can write this more compactly as

$$\sum_{i=1}^{3}\partial_{\mathbf{x}}^{\mathbf{k}+\mathbf{e}_i}G_\ell^\epsilon(\mathbf{x}) = \frac{2\ell+2}{(-\epsilon^2)}\sum_{i=1}^{3}\left(k_i\partial_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}G_{\ell+1}^\epsilon(\mathbf{x}) + x_i\partial_{\mathbf{x}}^{\mathbf{k}}G_{\ell+1}^\epsilon(\mathbf{x})\right). \tag{14}$$

Dividing both sides of eq. (14) by $k_1!\,k_2!\,k_3!$ and using the definition in eq. (11) recovers the desired result after some algebraic manipulation. □

Lemma 1 gives a recurrence relation that connects terms involving ℓ and $(\ell+1)$. We also require a generalized result of eq. (8). We first prove some useful lemmas before stating and proving a generalized recurrence relationship.

Proposition 2 *For any positive integer k, the function $G_\ell^\epsilon(\mathbf{x})$ satisfies the following relationship,*

$$\begin{aligned}
(\|\mathbf{x}\|_2^2+\epsilon^2)\,\partial_{x_i}^k G_\ell^\epsilon(\mathbf{x}) + (2k+2\ell-1)\,x_i\partial_{x_i}^{k-1}G_\ell^\epsilon(\mathbf{x}) + \cdots\\
(k-1)(k+2\ell-1)\,\partial_{x_i}^{k-2}G_\ell^\epsilon(\mathbf{x}) = 0,
\end{aligned} \tag{15}$$

where $\partial_{x_i}^{-1}G_\ell^\epsilon(\mathbf{x}) := 0$.

Proof We shall establish eq. (15) by induction. Applying the operator ∂_{x_i} to eq. (13), establishes eq. (15) for $k=1$,

$$(\|\mathbf{x}\|_2^2+\epsilon^2)\partial_{x_i}G_\ell^\epsilon(\mathbf{x}) + (2\ell+1)x_iG_\ell^\epsilon(\mathbf{x}) = 0. \tag{16}$$

Applying the operator ∂_{x_i} to eq. (16) establishes eq. (15) for $k = 2$.

$$(\|\mathbf{x}\|_2^2 + \epsilon^2)\partial_{x_i}^2 G_\ell^\epsilon(\mathbf{x}) + (2\ell+3)x_i\partial_{x_i} G_\ell^\epsilon(\mathbf{x}) + (2\ell+1)G_\ell^\epsilon(\mathbf{x}) = 0.$$

Suppose that eq. (15) holds for all $k \leq n$. Since

$$\begin{aligned}(\|\mathbf{x}\|_2^2 + \epsilon^2)\,\partial_{x_i}^n G_\ell^\epsilon(\mathbf{x}) + (2n+2\ell-1)\,x_i\partial_{x_i}^{n-1} G_\ell^\epsilon(\mathbf{x}) + \cdots \\ (n-1)(k+2\ell-1)\,\partial_{x_i}^{n-2} G_\ell^\epsilon(\mathbf{x}) = 0,\end{aligned} \tag{17}$$

applying ∂_{x_i} to eq. (17) gives

$$\begin{aligned}(\|\mathbf{x}\|_2^2 + \epsilon^2)\partial_{x_i}^{n+1} G_\ell^\epsilon(\mathbf{x}) + 2x_i\partial_{x_i}^n G_\ell^\epsilon(\mathbf{x}) + (2n+2\ell-1)\partial_{x_i}^{n-1} G_\ell^\epsilon(\mathbf{x}) + \cdots \\ (2n+2\ell-1)x_i\partial_{x_i}^n G_\ell^\epsilon(\mathbf{x}) + (n-1)(n+2\ell-1)\partial_{x_i}^{n-1} G_\ell^\epsilon(\mathbf{x}) = 0.\end{aligned}$$

Collecting terms and simplifying,

$$\begin{aligned}(\|\mathbf{x}\|_2^2 + \epsilon^2)\partial_{x_i}^{n+1} G_\ell^\epsilon(\mathbf{x}) + (2(n+1)+2\ell-1)x_i\partial_{x_i}^n G_\ell^\epsilon(\mathbf{x}) + \cdots \\ (n)(n+2\ell)\partial_{x_i}^{n-1} G_\ell^\epsilon(\mathbf{x}) = 0.\end{aligned}$$

This is exactly eq. (15) with $n = k+1$, establishing proposition 2. □

Proposition 3 *For any positive integers k_1 and k_2, the function $G_\ell^\epsilon(\mathbf{x})$ satisfies*

$$\begin{aligned}(\|\mathbf{x}\|_2^2 + \epsilon^2)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2} G_\ell^\epsilon(\mathbf{x}) + 2k_2x_2\partial_{x_1}^{k_1}\partial_{x_2}^{k_2-1} G_\ell^\epsilon(\mathbf{x}) + \cdots \\ k_2(k_2-1)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2-2} G_\ell^\epsilon(\mathbf{x}) + (2k_1+2\ell-1)x_1\partial_{x_1}^{k_1-1}\partial_{x_2}^{k_2} G_\ell^\epsilon(\mathbf{x}) + \cdots \\ (k_1-1)(k_1+2\ell-1)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2-2} G_\ell^\epsilon(\mathbf{x}) = 0,\end{aligned} \tag{18}$$

where $\partial_{x_1}^{-1}\partial_{x_2}^{k_2} G_\ell^\epsilon(\mathbf{x}) = \partial_{x_1}^{k_1}\partial_{x_2}^{-1} G_\ell^\epsilon(\mathbf{x}) := 0$.

Proof Invoke Proposition 2 with $i = 1$ and $k = k_1 > 0$. Applying the operator ∂_{x_2},

$$\begin{aligned}(\|\mathbf{x}\|_2^2 + \epsilon^2)\partial_{x_1}^{k_1}\partial_{x_2} G_\ell^\epsilon(\mathbf{x}) + 2x_2\partial_{x_1}^{k_1} G_\ell^\epsilon(\mathbf{x}) + \cdots \\ (2k_1+2\ell-1)x_1\partial_{x_1}^{k_1-1}\partial_{x_2} G_\ell^\epsilon(\mathbf{x}) + (k_1-1)(k_1+2\ell-1)\partial_{x_1}^{k_1-2}\partial_{x_2} G_\ell^\epsilon(\mathbf{x}) = 0,\end{aligned}$$

which shows that eq. (18) is true for $k_1 > 0$ and $k_2 = 1$. To prove by induction, suppose that eq. (18) is true for $k_1 > 0$ and $0 < k_2 \leq n$. Then,

$$\begin{aligned}(\|\mathbf{x}\|_2^2 + \epsilon^2)\partial_{x_1}^{k_1}\partial_{x_2}^{n} G_\ell^\epsilon(\mathbf{x}) + 2\,n\,x_2\,\partial_{x_1}^{k_1}\partial_{x_2}^{n-1} G_\ell^\epsilon(\mathbf{x}) + \cdots \\ n(n-1)\partial_{x_1}^{k_1}\partial_{x_2}^{n-2} G_\ell^\epsilon(\mathbf{x}) + (2k_1+2\ell-1)x_1\partial_{x_1}^{k_1-1}\partial_{x_2}^{n} G_\ell^\epsilon(\mathbf{x}) + \cdots \\ (k_1-1)(k_1+2\ell-1)\partial_{x_1}^{k_1}\partial_{x_2}^{n-2} G_\ell^\epsilon(\mathbf{x}) = 0.\end{aligned} \tag{19}$$

Applying the operator ∂_{x_2} operator to eq. (19),

$$\begin{aligned}(\|\mathbf{x}\|_2^2 + \epsilon^2)\partial_{x_1}^{k_1}\partial_{x_2}^{n+1} G_\ell^\epsilon(\mathbf{x}) + 2x_2\partial_{x_1}^{k_1}\partial_{x_2}^{n} G_\ell^\epsilon(\mathbf{x}) + 2nx_2\partial_{x_1}^{k_1}\partial_{x_2}^{n} G_\ell^\epsilon(\mathbf{x}) + \cdots \\ 2n\partial_{x_1}^{k_1}\partial_{x_2}^{n-1} G_\ell^\epsilon(\mathbf{x}) + n(n-1)\partial_{x_1}^{k_1}\partial_{x_2}^{n-1} G_\ell^\epsilon(\mathbf{x}) + \cdots \\ (2k_1+2\ell-1)x_1\partial_{x_1}^{k_1-1}\partial_{x_2}^{n+1} G_\ell^\epsilon(\mathbf{x}) + (k_1-1)(k_1+2\ell-1)\partial_{x_1}^{k_1}\partial_{x_2}^{n-1} G_\ell^\epsilon(\mathbf{x}) = 0.\end{aligned}$$

Collecting terms and simplifying,

$$\begin{aligned}(\|\mathbf{x}\|_2^2+\epsilon^2)\partial_{x_1}^{k_1}\partial_{x_2}^{n+1}G_\ell^\epsilon(\mathbf{x})+2(n+1)x_2\partial_{x_1}^{k_1}\partial_{x_2}^{n}G_\ell^\epsilon(\mathbf{x})+\cdots\\ (n+1)(n)\partial_{x_1}^{k_1}\partial_{x_2}^{n-1}G_\ell^\epsilon(\mathbf{x})+(2k_1+2\ell-1)x_1\partial_{x_1}^{k_1-1}\partial_{x_2}^{n+1}G_\ell^\epsilon(\mathbf{x})+\\ (k_1-1)(k_1+2\ell-1)\partial_{x_1}^{k_1}\partial_{x_2}^{n-1}G_\ell^\epsilon(\mathbf{x})=0.\end{aligned}$$

This is exactly eq. (18) with $n = k_2 + 1$, establishing proposition 3 for all positive k_1 and k_2. □

Proposition 4 *For any positive integers k_1, k_2 and k_3, the function $G_\ell^\epsilon(\mathbf{x})$ satisfies*

$$\begin{aligned}(\|\mathbf{x}\|_2^2+\epsilon^2)\partial_{\mathbf{x}}^{\mathbf{k}}G_\ell^\epsilon(\mathbf{x})+2k_3x_3\partial_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_3}G_\ell^\epsilon(\mathbf{x})+k_3(k_3-1)\partial_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_3}G_\ell^\epsilon(\mathbf{x})+\cdots\\ 2k_2x_2\partial_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_2}G_\ell^\epsilon(\mathbf{x})+k_2(k_2-1)\partial_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_2}G_\ell^\epsilon(\mathbf{x})+(2k_1+2\ell-1)x_1\partial_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1}G_\ell^\epsilon+\cdots\\ (k_1-1)(k_1+2\ell-1)\partial_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_1}G_\ell^\epsilon(\mathbf{x})=0,\end{aligned}\tag{20}$$

where $\mathbf{e}_i$ are the Cartesian basis vectors, and

$$\partial_{x_1}^{-1}\partial_{x_2}^{k_2}\partial_{x_3}^{k_3}G_\ell^\epsilon(\mathbf{x})=\partial_{x_1}^{k_1}\partial_{x_2}^{-1}\partial_{x_3}^{k_3}G_\ell^\epsilon(\mathbf{x})=\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}^{-1}G_\ell^\epsilon(\mathbf{x}):=0.$$

Proof Applying ∂_{x_3} to eq. (18),

$$\begin{aligned}(\|\mathbf{x}\|_2^2+\epsilon^2)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}G_\ell^\epsilon(\mathbf{x})+2x_3\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}G_\ell^\epsilon(\mathbf{x})+2k_2x_2\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}G_\ell^\epsilon(\mathbf{x})+\cdots\\ k_2(k_2-1)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2-2}\partial_{x_3}G_\ell^\epsilon(\mathbf{x})+(2k_1+2\ell-1)x_1\partial_{x_1}^{k_1-1}\partial_{x_2}^{k_2}\partial_{x_3}G_\ell^\epsilon(\mathbf{x})+\cdots\\ (k_1-1)(k_1+2\ell-1)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2-2}\partial_{x_3}G_\ell^\epsilon(\mathbf{x})=0\end{aligned}$$

which establishes eq. (20) for $k_1 > 0$, $k_2 > 0$, and $k_3 = 1$. To proceed by induction, suppose that eq. (20) is true for $k_1 > 0, k_2 > 0$ and $0 < k_3 \leq n$. Then,

$$\begin{aligned}(\|\mathbf{x}\|_2^2+\epsilon^2)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}^{n}G_\ell^\epsilon(\mathbf{x})+2n\,x_3\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}^{n-1}G_\ell^\epsilon(\mathbf{x})+\cdots\\ n(n-1)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}^{n-2}G_\ell^\epsilon(\mathbf{x})+2k_2x_2\partial_{x_1}^{k_1}\partial_{x_2}^{k_2-1}\partial_{x_3}^{n}G_\ell^\epsilon(\mathbf{x})+\cdots\\ k_2(k_2-1)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2-2}\partial_{x_3}^{n}G_\ell^\epsilon(\mathbf{x})+(2k_1+2\ell-1)x_1\partial_{x_1}^{k_1-1}\partial_{x_2}^{k_2}\partial_{x_3}^{n}G_\ell^\epsilon+\cdots\\ (k_1-1)(k_1+2\ell-1)\partial_{x_1}^{k_1-2}\partial_{x_2}^{k_2}\partial_{x_3}^{n}G_\ell^\epsilon(\mathbf{x})=0.\end{aligned}$$

Applying the operator ∂_{x_3} gives

$$\begin{aligned}(\|\mathbf{x}\|_2^2+\epsilon^2)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}^{n+1}G_\ell^\epsilon(\mathbf{x})+2x_3\,\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}^{n}G_\ell^\epsilon(\mathbf{x})+\cdots\\ 2n\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}^{n-1}G_\ell^\epsilon(\mathbf{x})+2n\,x_3\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}^{n}G_\ell^\epsilon(\mathbf{x})+n(n-1)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2}\partial_{x_3}^{n-1}G_\ell^\epsilon(\mathbf{x})+\cdots\\ 2k_2x_2\partial_{x_1}^{k_1}\partial_{x_2}^{k_2-1}\partial_{x_3}^{n+1}G_\ell^\epsilon(\mathbf{x})+k_2(k_2-1)\partial_{x_1}^{k_1}\partial_{x_2}^{k_2-2}\partial_{x_3}^{n+1}G_\ell^\epsilon(\mathbf{x})+\cdots\\ (2k_1+2\ell-1)x_1\partial_{x_1}^{k_1-1}\partial_{x_2}^{k_2}\partial_{x_3}^{n+1}G_\ell^\epsilon(\mathbf{x})+\cdots\\ (k_1-1)(k_1+2\ell-1)\partial_{x_1}^{k_1-2}\partial_{x_2}^{k_2}\partial_{x_3}^{n+1}G_\ell^\epsilon(\mathbf{x})=0.\end{aligned}$$

Collecting terms and simplifying gives eq. (20) with $n = k_3 + 1$, establishing proposition 4 for all positive k_1, k_2 and k_3. □

Lemma 2 *For $\ell \geq 0$, the Taylor coefficients, $a_\ell^{\mathbf{k}}$ satisfy*

$$\begin{aligned}\left(\|\mathbf{x}\|_2^2+\epsilon^2\right)\sum_{i=1}^{3} k_i\, a_\ell^{\mathbf{k}}(\mathbf{x}) + \left(2\left(\|\mathbf{k}|_1\right)+2\ell-1\right)\sum_{i=1}^{3} x_i a_\ell^{\mathbf{k}-\mathbf{e}_i}(\mathbf{x}) + \cdots \\ \left(\|\mathbf{k}\|_1+2\ell-1\right)\sum_{i=1}^{3} a_\ell^{\mathbf{k}-2\mathbf{e}_i}(\mathbf{x}) = 0,\end{aligned} \tag{21}$$

where

$$a_\ell^{-1,k_2,k_3}(\mathbf{x}) = a_\ell^{k_1,-1,k_3}(\mathbf{x}) = a_\ell^{k_1,k_2,-1}(\mathbf{x}) := 0.$$

Proof Dividing eq. (20) by $\mathbf{k}! = k_1!k_2!k_3!$ and using the definition in eq. (11),

$$\begin{aligned}(\|\mathbf{x}\|_2^2+\epsilon^2)a_\ell^{\mathbf{k}}(\mathbf{x}) + 2\sum_{i=1}^{3} x_i a_\ell^{\mathbf{k}-\mathbf{e}_i}(\mathbf{x}) + \sum_{i=1}^{3} a_\ell^{\mathbf{k}-2\mathbf{e}_i}(\mathbf{x}) + \cdots \\ \frac{2\ell-1}{k_1}\left(x_1 a_\ell^{\mathbf{k}-\mathbf{e}_1}(\mathbf{x}) + a_\ell^{\mathbf{k}-2\mathbf{e}_1}(\mathbf{x})\right) = 0.\end{aligned}$$

More generally, the following result can be obtained by permuting the order in which the differential operators, $\{\partial_{x_1}^{k_1}, \partial_{x_2}^{k_2}, \partial_{x_3}^{k_3}\}$, are applied in Propositions 3 and 4,

$$\begin{aligned}(\|\mathbf{x}\|_2^2+\epsilon^2)a_\ell^{\mathbf{k}}(\mathbf{x}) + 2\sum_{i=1}^{3} x_i a_\ell^{\mathbf{k}-\mathbf{e}_i}(\mathbf{x}) + \sum_{i=1}^{3} a_\ell^{\mathbf{k}-2\mathbf{e}_i}(\mathbf{x}) + \cdots \\ \frac{2\ell-1}{k_j}\left(x_j a_\ell^{\mathbf{k}-\mathbf{e}_j}(\mathbf{x}) + a_\ell^{\mathbf{k}-2\mathbf{e}_j}(\mathbf{x})\right) = 0, \qquad j=1,2,3.\end{aligned} \tag{22}$$

Summing up eq. (22) for $j = 1, 2, 3$ and collecting terms recovers eq. (21) after some algebraic manipulation, establishing Lemma 2. □

If $\ell = 0$, eq. (21) simplifies to eq. (8).

3.3 Computing the Taylor Coefficients

At first glance, it might appear that computing the Taylor coefficients, eq. (11), using Lemmas 1 and 2 will require solving systems of equations. In fact, by carefully ordering the computation of the Taylor coefficients, each Taylor coefficient can be explicitly computed.

Suppose that we require a p-th order Taylor expansion of $G^{\epsilon,n}(\mathbf{x})$ in eq. (6), i.e., we are interested in $a_\ell^{\mathbf{k}}(\mathbf{x})$ for $\ell = 0, \ldots, n$ and $\|\mathbf{k}\|_1 = k_1+k_2+k_3 \leq p$. Algorithm 1 provides one approach for computing the desired Taylor coefficients. Line 8 in Algorithm 1 can be computed using a parallel for-loop since the data $\{a_\ell^{\mathbf{k}-\mathbf{e}_j}(\mathbf{x}), a_\ell^{\mathbf{k}-2\mathbf{e}_j}(\mathbf{x}),\ j=1,2,3\}$ has already been computed and is available. If further parallelism and memory optimization is desired, one can simultaneously computed multiple ℓ's in a pipeline-parallel fashion. A dependency flow-chart is given in fig. 1.

Algorithm 1: Pseudocode for using Lemmas 1 and 2 to compute Taylor coefficients.

```
Specify a_0^{(0,0,0)}(x);
Initialize a^k(x) = 0;
for ℓ = 0, ..., p do
    if ℓ > 0 then
        Use Lemma 1 to compute a_ℓ^{(0,0,0)}(x);
    end
    for c = 1, ..., p do
        for all ||k||_1 = c do
            Use Lemma 2 to compute a_ℓ^k(x);
        end
    end
    a^k(x) ← a^k(x) + a_ℓ^k(x);
end
```

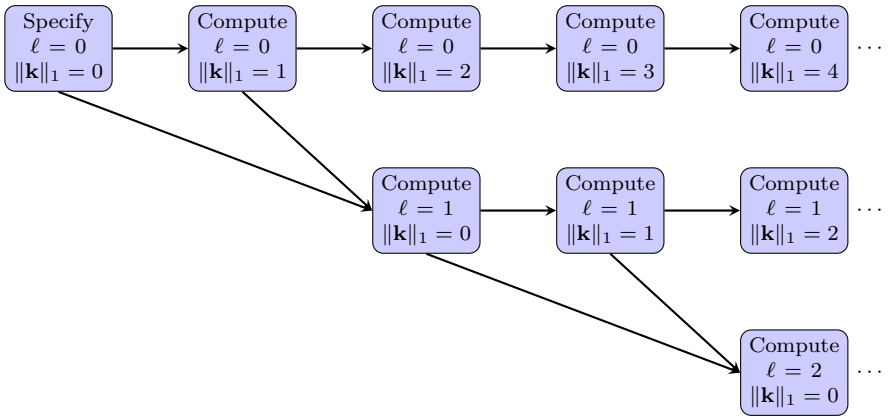

Fig. 1 Taylor coefficients can be computed in a pipeline parallel fashion – purple tasks in each column can be simultaneously computed once dependencies (arrows) are satisfied.

4 Numerical experiments

In the first numerical experiment, we use the recurrence relations, lemma 1, to compute gradients of the regularized kernels, eq. (4). We place a unit mass at the origin, $(0, 0, 0)$. The true force asserted by the unit mass at $\mathbf{x} = (0.1, 0.1, 0.1)$ is given by the gradient of the unregularized kernel, eq. (2). The convergence of the force due to the regularized system is shown in fig. 2. Specifically, we compute $\|\nabla G(\mathbf{x}) - \nabla G^{\epsilon,n}(\mathbf{x})\|_2$ for various n and ϵ using the recurrence relations in Lemma 1. In Figure 2. we observe convergence as $n \to \infty$ and as $\epsilon \to 0$.

In the second numerical experiment, we validate the recurrence relations, Lemmas 1 and 2, for computing higher-order derivatives of the regularized kernels, eq. (4). We initialize a cluster of ten unit masses whose locations are sampled from the unit normal distribution, $X^{10\times 3} \sim N(0, 1)$. The force due to

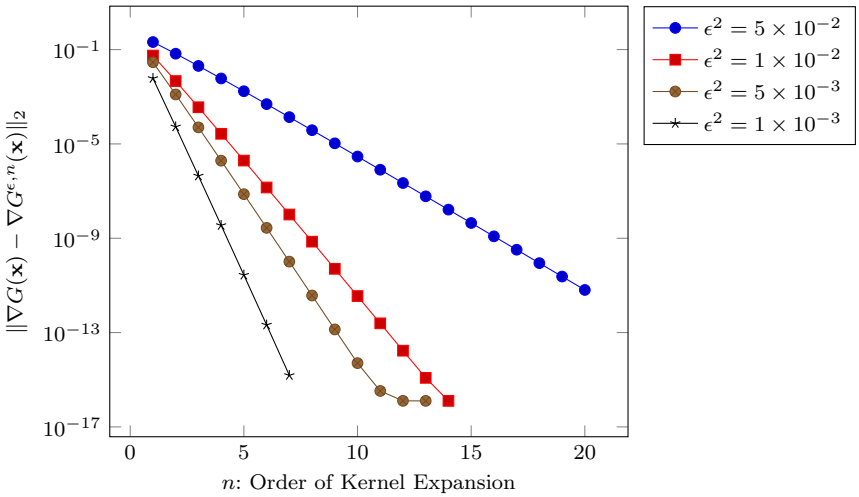

Fig. 2 Convergence of the gradient of the regularized kernels, $\nabla G^{\epsilon,n}$, to the gradient of the unregularized kernels for various n and ϵ. The recurrence relation in Lemma 1 are used to compute the values. Convergence to machine precision is observed as $n \to \infty$ and as $\epsilon \to 0$.

these ten particles can be computed using a direct summation, $\sum_{i=1}^{10} \nabla G(\mathbf{x} - \mathbf{x}_i)$. We are interested in approximating the force exerted by these particles at $\mathbf{x} = (5, 5, 5)$, where a cluster is appropriate. The pth-order cluster approximation using the regularized kernel $G^{\epsilon,n}(\mathbf{x})$ is given by eq. (6). Using the recurrence relations, Lemmas 1 and 2, we compute the error as a function of p: the order of the cluster (Taylor) approximation, for various regularized kernels $G^{\epsilon,m}(\mathbf{x})$. In Figure 3, $\epsilon^2 = 10^{-2}$; in Figure 4, $\epsilon^2 = 5 \times 10^{-4}$. For each (ϵ, n) pairing, the error stagnates as the order of the cluster approximation is improved. The value at which the error stagnates decreases as n is increased or as $\epsilon is decreased$. Both Figures 3 and 4 indicate that the recurrence relations correctly give the desired cluster approximations. Comparing the two figures leads to the observation that for fixed n, the value at which the error stagnates is smaller when ϵ is reduced.

Lastly, we perform some timing runs that illustrate the benefits of treecode approximations and the potential computational overhead in using regularized kernels. The research-grade C++ code used to generate these timing studies is available at `http://github.org/ongbw/bit`. We initialize N particles, sampled uniformly in $[0, 1]^3$, and compute the force exerted on each particle using direct summation and the treecode approximation. For the treecode approximation, a MAC criterion of $\theta = 0.5$ was used; we constrain the tree to have no more than 10^5 leaves/children, and each leaf is not sub-divided if there are fewer than 100 particles. As fig. 5 illustrates, the direct sum computation takes $\mathcal{O}(N^2)$ operations, which is significantly more expensive than the treecode approximation for a large number of particles. The plots also show the increased computational cost when the higher-order regularized kernels are used within the treecode.

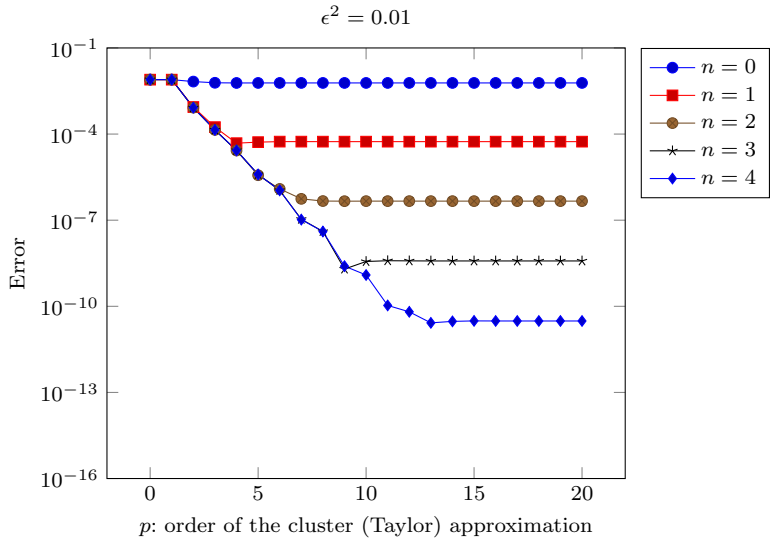

Fig. 3 Error versus p:. Error: difference between the direct-sum, unregularized kernels, and the cluster-approximation, regularized kernels. p: the order of the cluster (Taylor) approximation. For each regularized kernel, $G^{0.1,n}(\mathbf{x})$ used in the cluster approximation, the error stagnates as p increases. The value at which the error stagnates decreases as n is increased.

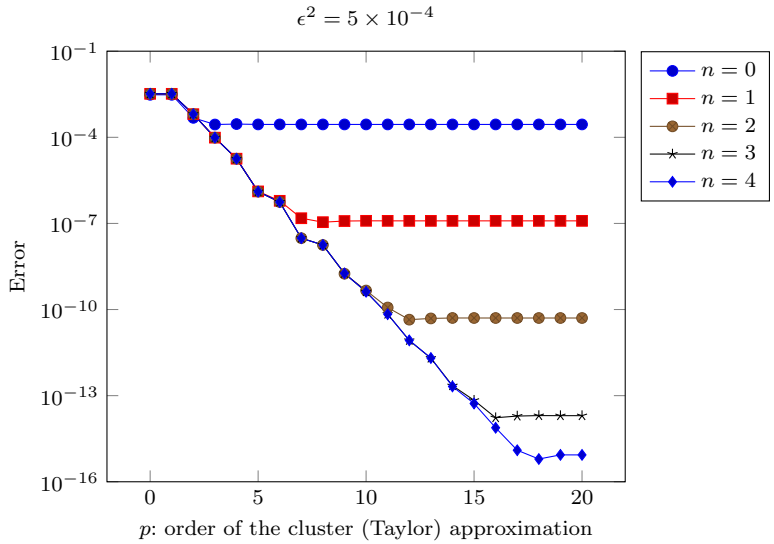

Fig. 4 Error versus p:. Error: difference between the direct-sum, unregularized kernels, and the cluster-approximation, regularized kernels. p: the order of the cluster (Taylor) approximation. For each regularized kernel, $G^{\sqrt{0.0005},n}(\mathbf{x})$ used in the cluster approximation, the error stagnates as p increases. The value at which the error stagnates decreases as n is increased.

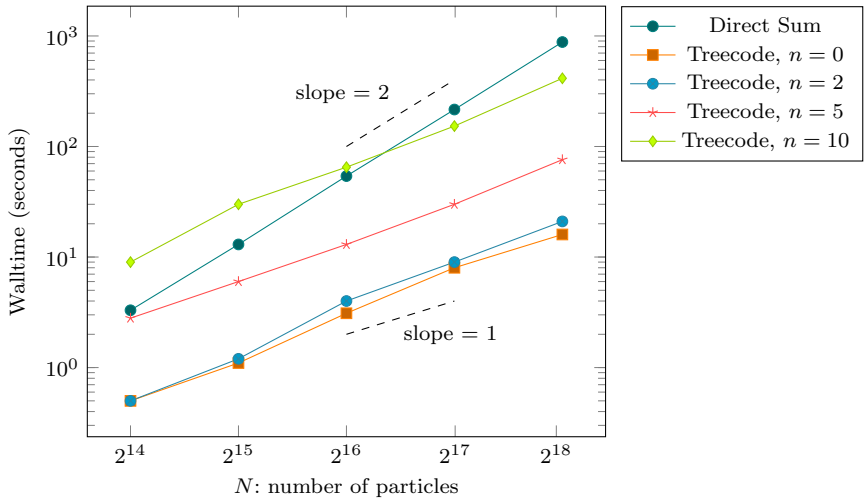

Fig. 5 Timing study for a direct-sum computation and the treecode algorithm with various regularized kernels.

Conclusions

The main contribution of this work is the derivation of two useful recurrence relationships for computing the Taylor coefficients of high-order regularized kernels within a treecode approximation; these high-order regularized kernels have previously been shown to be useful for controlling modeling error in N-body simulation. The research-grade treecode software, available at `http://github.com/ongbw/bit`, implements the recurrence relations to compute the cluster approximations. The expected convergence behavior is observed, as well as the desired speedup that a treecode approximation provides. The software can be significantly improved by computing the Taylor coefficients in a pipeline parallel fashion, as this minimizes storage overhead in addition to accelerating the treecode computation. Additionally, one could use the error estimates previously derived in [10] to generate an adaptive-order treecode algorithm, where additional moments are used as greater fidelity is desired. A future research direction that the authors are pursuing, is the development of high-order regularized kernels for the non-oscillatory Helmholtz equation.

References

1. Bate, R.R., Mueller, D.D., White, J.E.: Fundamentals of astrodynamics. Dover Publications (1971)
2. Birdsall, C.K., Langdon, A.B.: Plasma Physics via Computer Simulation. CRC Press (2004). DOI 10.1201/9781315275048. URL `https://doi.org/10.1201/9781315275048`
3. Christlieb, A., Krasny, R., Verboncoeur, J.: A treecode algorithm for simulating electron dynamics in a penning–malmberg trap. Computer Physics Communications **164**(1), 306 – 310 (2004). DOI https://doi.org/10.1016/j.cpc.2004.06.076. URL

http://www.sciencedirect.com/science/article/pii/S0010465504002966. Proceedings of the 18th International Conferene on the Numerical Simulation of Plasmas
4. Christlieb, A.J., Krasny, R., Verboncoeur, J.P., Emhoff, J.W., Boyd, I.D.: Grid-free plasma simulation techniques. IEEE Transactions on Plasma Science **34**(2), 149–165 (2006). DOI 10.1109/TPS.2006.871104
5. Coifman, R., Rokhlin, V., Wandzura, S.: The fast multipole method for the wave equation: a pedestrian prescription. IEEE Antennas and Propagation Magazine **35**(3), 7–12 (1993). DOI 10.1109/74.250128
6. van Elteren, A., Bédorf, J., Portegies Zwart, S.: Multi-scale high-performance computing in astrophysics: simulating clusters with stars, binaries and planets. Philosophical Transactions of the Royal Society A (2018). DOI 10.1098/rsta.2018.0153. URL https://doi.org/10.1098/rsta.2018.0153
7. Jackson, J.D.: Classical electrodynamics. Wiley (1999)
8. Leonard, A.: Vortex methods for flow simulation. J. Comput. Phys. **37**(3), 289–335 (1980). DOI 10.1016/0021-9991(80)90040-6. URL http://dx.doi.org/10.1016/0021-9991(80)90040-6
9. Lindsay, K., Krasny, R.: A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow. J. Comput. Phys. **172**(2), 879–907 (2001). DOI 10.1006/jcph.2001.6862
10. Ong, B.W., Christlieb, A.J., Quaife, B.D.: A new family of regularized kernels for the harmonic oscillator. J. Sci. Comput. **71**(3), 1212–1237 (2017). DOI 10.1007/s10915-016-0336-0. URL https://doi.org/10.1007/s10915-016-0336-0
11. Speck, R., Arnold, L., Gibbon, P.: Towards a petascale tree code: Scaling and efficiency of the pepc library. Journal of Computational Science **2**(2), 138 – 143 (2011). DOI https://doi.org/10.1016/j.jocs.2011.01.011. URL http://www.sciencedirect.com/science/article/pii/S1877750311000184. Simulation Software for Supercomputers
12. Winckelmans, G.S., Leonard, A.: Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows. J. Comput. Phys. **109**(2), 247–273 (1993). DOI 10.1006/jcph.1993.1216. URL http://dx.doi.org/10.1006/jcph.1993.1216
13. Zhong-Hui, D., Robert, K.: An adaptive treecode for computing nonbonded potential energy in classical molecular systems. Journal of Computational Chemistry **22**(2), 184–195 (2001). DOI 10.1002/1096-987X(20010130)22:2⟨184::AID-JCC6⟩3.0.CO;2-7. First discussion of recurrence relation in 3d for laplace regularized kernel.