Research Article

Robert Krasny* and Lei Wang

A treecode based on barycentric Hermite interpolation for electrostatic particle interactions

https://doi.org/10.1515/cmb-2019-0006
Received August 30, 2019; accepted November 24, 2019

Abstract: A particle-cluster treecode based on barycentric Hermite interpolation is presented for fast summation of electrostatic particle interactions in 3D. The interpolation nodes are Chebyshev points of the 2nd kind in each cluster. It is noted that barycentric Hermite interpolation is scale-invariant in a certain sense that promotes the treecode's efficiency. Numerical results for the Coulomb and screened Coulomb potentials show that the treecode run time scales like $O(N \log N)$, where N is the number of particles in the system. The advantage of the barycentric Hermite treecode is demonstrated in comparison with treecodes based on Taylor approximation and barycentric Lagrange interpolation.

Keywords: treecode; barycentric Hermite interpolation; electrostatic particle interactions

MSC: 65B99; 65D05; 65Y20; 65Z05

1 Introduction

Electrostatic effects play an important role in determining biomolecular structure, function, and dynamics [4, 16]. For example among a wide range of applications, biophysics researchers are interested in understanding the influence of electrostatic effects on protein folding and protein-ligand binding [30, 31]. Electrostatic effects are inherently multiscale and they pose an outstanding challenge in computational molecular biophysics [23]. Powerful computer hardware is certainly required, but advances in numerical algorithms are also necessary to achieve the goal of accurate and efficient simulations of large-scale biomolecular systems.

Here we consider the basic problem of computing the electrostatic potential due to a set of charged particles in 3D space,

$$\phi(\mathbf{x}_i) = \sum_{j=1, j\neq i}^{N} g(\mathbf{x}_i, \mathbf{x}_j) q_j, \quad i = 1, \dots, N, \tag{1}$$

where $\mathbf{x}_i$ is a target site, $\mathbf{x}_j$ is a source site of a particle with charge q_j, and $g(\mathbf{x}, \mathbf{y})$ is an interaction kernel. Two important examples are the Coulomb potential and screened Coulomb potential,

$$g(\mathbf{x}, \mathbf{y}) = \frac{1}{|\mathbf{x} - \mathbf{y}|}, \quad g(\mathbf{x}, \mathbf{y}) = \frac{e^{-\kappa|\mathbf{x}-\mathbf{y}|}}{|\mathbf{x} - \mathbf{y}|}, \tag{2}$$

where κ is the inverse Debye length. Computing the potential $\phi(\mathbf{x}_i)$ at all the target sites by direct summation requires $O(N^2)$ operations and many fast summation methods have been developed to reduce the cost including particle-mesh methods (e.g. PPPM [15], PME [8], MSM [13]) and tree-based methods (e.g. treecode [1],

***Corresponding Author: Robert Krasny:** Department of Mathematics, University of Michigan, Ann Arbor, MI 48109, USA, E-mail: krasny@umich.edu
Lei Wang: Department of Mathematical Sciences, University of Wisconsin-Milwaukee, Milwaukee, WI 53211, USA, E-mail: wang256@uwm.edu

FMM [11], panel clustering [12]). However there is continuing interest in exploring different approaches to improve the performance of these methods, especially in the context of parallel simulations of large-scale biomolecular systems.

Treecode algorithms are interesting due to their relatively simple structure and low memory consumption. In these algorithms, some of the particle-particle interactions are computed directly and others are computed by particle-cluster approximations. The resulting run time scales like $O(N \log N)$, where N is the number of particles in the system. Previously we developed a treecode based on Cartesian Taylor approximation (TTC) [18] and a kernel-independent treecode based on barycentric Lagrange interpolation (BLTC) [27]. Here we extend the latter method to obtain a treecode based on barycentric Hermite interpolation (BHTC); it interpolates the kernel and its first partial derivatives in the three coordinate directions at Chebyshev points of the 2nd kind. As we explain below, barycentric Hermite interpolation is scale-invariant in a certain sense that promotes the efficiency of the treecode. Although the BHTC it is not strictly kernel-independent, for many kernels of practical interest the necessary derivatives can be easily computed. In this work we demonstrate the BHTC in computing electrostatic potentials due to Coulomb and screened Coulomb interactions, and numerical results show the improved performance of BHTC in comparison with TTC and BLTC.

The paper is organized as follows. Section 2 reviews polynomial interpolation in 1D. Section 3 discusses the extension to polynomial interpolation in 3D. Section 4 presents the barycentric Hermite treecode. Section 5 displays numerical results. Conclusions are given in Section 6.

2 Polynomial interpolation in 1D

We review polynomial interpolation in 1D including Lagrange form, barycentric Lagrange form, and barycentric Hermite form [25].

Lagrange form. Given a function $f(x)$ and a set of $n + 1$ distinct nodes $\{x_0, \dots, x_n\}$, the Lagrange form of the interpolating polynomial is

$$p(x) = \sum_{k=0}^{n} L_k(x) f(x_k), \quad L_k(x) = \frac{\Pi_{j=0\,,\, j\neq k}^{n}(x - x_j)}{\Pi_{j=0\,,\, j\neq k}^{n}(x_k - x_j)}, \quad k = 0 : n, \tag{3}$$

where the Lagrange polynomials satisfy $L_k(x_j) = \delta_{jk}$ for $j, k = 0 : n$. This form is well-known and appears in many textbooks, however Berrut and Trefethen [3] pointed out the advantages of the barycentric Lagrange form.

Barycentric Lagrange form. The interpolating polynomial can also be written in barycentric Lagrange form,

$$p(x) = \frac{\sum\limits_{k=0}^{n} \frac{w_k}{x - x_k} f(x_k)}{\sum\limits_{k=0}^{n} \frac{w_k}{x - x_k}}, \quad w_k = \frac{1}{\Pi_{j=0\,,\, j\neq k}^{n}(x_k - x_j)}, \quad k = 0 : n, \tag{4}$$

where the removable singularity at the nodes is resolved by setting $p(x_k) = f(x_k)$. Here we choose the nodes to be Chebyshev points of the 2nd kind,

$$x_k = \cos\theta_k, \quad \theta_k = k\pi/n, \quad k = 0 : n. \tag{5}$$

In this case as shown in [22], the barycentric weights w_k in (4) may be taken as

$$w_k = (-1)^k \delta_k, \quad \delta_k = \begin{cases} 1, & k = 1 : n - 1, \\ 1/2, & k = 0, n. \end{cases} \tag{6}$$

Among the advantages of barycentric Lagrange interpolation at Chebyshev points are excellent approximation properties, low cost, and stability [3, 25, 14]. It also has a scale-invariance property in that the same weights (6) can be used for any interval $[a, b]$, as long as the Chebyshev points are adapted to the interval; this is particularly useful in the context of the treecode [27].

Barycentric Hermite form. Next we consider Hermite interpolation [24, 21, 2]. Given function values and first derivative values at the nodes, $\{f_k, f_k', k = 0 : n\}$, the Hermite interpolating polynomial can be written in the form

$$p(x) = \sum_{k=0}^{n} \left(H_k(x) f_k + \widehat{H}_k(x) f_k' \right), \tag{7}$$

where the Hermite basis functions $H_k(x), \widehat{H}_k(x)$ are polynomials of degree $2n + 1$ satisfying

$$H_k(x_j) = \delta_{jk}, \quad H_k'(x_j) = 0, \quad \widehat{H}_k(x_j) = 0, \quad \widehat{H}_k'(x_j) = \delta_{jk}, \quad j, k = 0 : n. \tag{8}$$

It can be verified that the polynomial (7) interpolates the prescribed function and derivative values at the nodes, $p(x_k) = f_k, p'(x_k) = f_k', k = 0 : n$. Following [24], the Hermite interpolating polynomial can be written in barycentric form,

$$p(x) = \frac{\displaystyle\sum_{k=0}^{n} \left(\frac{a_k}{x - x_k} + \frac{b_k}{(x - x_k)^2} \right) f_k + \frac{b_k}{x - x_k} f_k'}{\displaystyle\sum_{k=0}^{n} \left(\frac{a_k}{x - x_k} + \frac{b_k}{(x - x_k)^2} \right)}, \tag{9}$$

where again the removable singularity is resolved by setting $p(x_k) = f_k$. When the nodes are the Chebyshev points of the 2nd kind (5), it was shown in [2] that the barycentric Hermite weights may be taken as

$$a_k = \begin{cases} \dfrac{x_k}{1 - x_k^2}, & k = 1 : n - 1, \\ -\frac{1}{12}(1 + 2n^2), & k = 0, \\ \frac{1}{12}(1 + 2n^2), & k = n, \end{cases} , \qquad b_k = \begin{cases} 1, & k = 1 : n - 1, \\ 1/4, & k = 0, n. \end{cases} \tag{10}$$

For reference note that the Hermite basis functions are

$$H_k(x) = \frac{\dfrac{a_k}{x - x_k} + \dfrac{b_k}{(x - x_k)^2}}{\displaystyle\sum_{k=0}^{n} \left(\frac{a_k}{x - x_k} + \frac{b_k}{(x - x_k)^2} \right)}, \quad \widehat{H}_k(x) = \frac{\dfrac{b_k}{x - x_k}}{\displaystyle\sum_{k=0}^{n} \left(\frac{a_k}{x - x_k} + \frac{b_k}{(x - x_k)^2} \right)}. \tag{11}$$

In practice the removable singularity at $x = x_k$ can be handled following [3] (see also [27]); if the argument x is closer to a node x_k than some tolerance, this is flagged and the correct values $H_k(x_j) = \delta_{jk}, \widehat{H}_k(x_j) = 0$ are provided when evaluating the polynomial (7); details relevant to the treecode will be given further below.

Extension to a general interval. The Chebyshev points (5) are suitable for interpolation on the interval $[-1, 1]$. For a general interval $[a, b]$, the function $x \to (a + b)/2 + x(b - a)/2$ is used to map the Chebyshev points to the desired interval, but we must also consider what happens to the barycentric Hermite weights a_k, b_k in (10). First note that the barycentric expressions (9), (11) are scale-invariant; if the weights a_k, b_k have a common factor independent of k, then the factor can be cancelled and the expressions (9), (11) stay the same. In the case of general nodes, following the derivation of the barycentric Lagrange weights in [22], it can be shown that the barycentric Hermite weights satisfy

$$a_k = -2b_k \sum_{j=0, j\neq k}^{n} \frac{1}{x_k - x_j}, \quad b_k = \frac{1}{(\psi'(x_k))^2}, \quad \psi(x) = \Pi_{j=0}^{n}(x - x_j). \tag{12}$$

Hence under the mapping $[-1, 1] \to [a, b]$, the weights b_k acquire a factor $(2/(b - a))^{2n}$, while the weights a_k acquire a factor $(2/(b - a))^{2n+1}$. Due to scale-invariance, the factor $(2/(b - a))^{2n}$ can be cancelled from a_k, b_k, so that for a general interval $[a, b]$, the barycentric Hermite weights in (10) should be scaled so that $a_k \to 2a_k/(b - a)$, while the b_k stay the same. The scale-invariance property of barycentric Hermite interpolation facilitates the treatment of different clusters in the treecode and this promotes the scheme's efficiency.

3 Extension to 3D

Now consider a function $f(\mathbf{x}) = f(x_1, x_2, x_3)$, and a tensor product grid of Chebyshev points $\mathbf{s_k} = (s_{k_1}, s_{k_2}, s_{k_3}) \in [-1, 1]^3$, where $\mathbf{k} = (k_1, k_2, k_3)$ is a multi-index with $k_\ell = 0, \ldots, n$ for $\ell = 1, 2, 3$. Letting $p(x_1, x_2, x_3)$ denote the Hermite interpolating polynomial, there are eight interpolation conditions to be satisfied at each Chebyshev point $\mathbf{s_k}$,

$$\begin{aligned} &p = f, && p_1 = f_1, && p_2 = f_2, && p_3 = f_3, \\ &p_{12} = f_{12}, && p_{13} = f_{13}, && p_{23} = f_{23}, && p_{123} = f_{123}, \end{aligned} \tag{13}$$

where the subscript denotes a partial derivative, for example $p_{12} = \partial^2 p/\partial x_1 \partial x_2$; here and below we sometimes omit function arguments for clarity. The eight conditions (13) represent all combinations of first partial derivatives in three variables. Using the properties of the basis functions (8), it follows that the Hermite interpolating polynomial is

$$\begin{aligned} p(x_1, x_2, x_3) = \sum_{k_1=0}^{n}\sum_{k_2=0}^{n}\sum_{k_3=0}^{n} \Big[& H_{k_1}^{x_1}\Big(H_{k_2}^{x_2}\Big(H_{k_3}^{x_3} f^s + \widehat{H}_{k_3}^{x_3} f_3^s\Big) + \widehat{H}_{k_2}^{x_2}\Big(H_{k_3}^{x_3} f_2^s + \widehat{H}_{k_3}^{x_3} f_{23}^s\Big)\Big) \\ & + \widehat{H}_{k_1}^{x_1}\Big(H_{k_2}^{x_2}\Big(H_{k_3}^{x_3} f_1^s + \widehat{H}_{k_3}^{x_3} f_{13}^s\Big) + \widehat{H}_{k_2}^{x_2}\Big(H_{k_3}^{x_3} f_{12}^s + \widehat{H}_{k_3}^{x_3} f_{123}^s\Big)\Big)\Big], \end{aligned} \tag{14}$$

where $H_{k_\ell}^{x_\ell} = H_{k_\ell}(x_\ell)$, $\widehat{H}_{k_\ell}^{x_\ell} = \widehat{H}_{k_\ell}(x_\ell)$, and superscript s indicates that f and its partial derivatives are evaluated at the Chebyshev points $(s_{k_1}, s_{k_2}, s_{k_3})$. It can be verified that (14) satisfies the interpolation conditions (13).

This result is applied in the treecode to approximate a kernel $g(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$. In that case, holding $\mathbf{x}$ fixed and interpolating with respect to $\mathbf{y}$ yields,

$$\begin{aligned} g(\mathbf{x}, \mathbf{y}) \approx \sum_{k_1=0}^{n}\sum_{k_2=0}^{n}\sum_{k_3=0}^{n} \Big[& H_{k_1}^{y_1}\Big(H_{k_2}^{y_2}\Big(H_{k_3}^{y_3} g^s + \widehat{H}_{k_3}^{y_3} g_3^s\Big) + \widehat{H}_{k_2}^{y_2}\Big(H_{k_3}^{y_3} g_2^s + \widehat{H}_{k_3}^{y_3} g_{23}^s\Big)\Big) \\ & + \widehat{H}_{k_1}^{y_1}\Big(H_{k_2}^{y_2}\Big(H_{k_3}^{y_3} g_1^s + \widehat{H}_{k_3}^{y_3} g_{13}^s\Big) + \widehat{H}_{k_2}^{y_2}\Big(H_{k_3}^{y_3} g_{12}^s + \widehat{H}_{k_3}^{y_3} g_{123}^s\Big)\Big)\Big], \end{aligned} \tag{15}$$

where now the target variables (x_1, x_2, x_3) are parameters in the kernel and its partial derivatives, the Hermite basis functions are evaluated at the source variables (y_1, y_2, y_3), while the partial derivatives of the kernel are taken with respect to (y_1, y_2, y_3) and are evaluated at the Chebyshev points $(s_{k_1}, s_{k_2}, s_{k_3})$.

4 Barycentric Hermite treecode

In a treecode the particles $\mathbf{x}_i$ are partitioned into a hierarchy of clusters C, and the sum (1) is written as

$$\phi(\mathbf{x}_i) = \sum_{j=1}^{N} g(\mathbf{x}_i, \mathbf{x}_j) q_j = \sum_{C} \phi(\mathbf{x}_i, C), \quad i = 1, \ldots, N, \tag{16}$$

where

$$\phi(\mathbf{x}_i, C) = \sum_{\mathbf{y}_j \in C} g(\mathbf{x}_i, \mathbf{y}_j) q_j, \tag{17}$$

is the interaction between a target particle $\mathbf{x}_i$ and a cluster of source particles $C = \{\mathbf{y}_j\}$. The sum over C in (16) denotes a suitable set of clusters depending on the target particle. In our implementation, the clusters are rectangular boxes.

Particle-cluster approximation. The particle-cluster interaction (17) can be approximated using barycentric Hermite interpolation as follows. We substitute the kernel approximation (15) into the particle-cluster interaction (17) and switch the order of summation to obtain the particle-cluster approximation,

$$\phi(\mathbf{x}_i, C) \approx \sum_{k_1=0}^{n}\sum_{k_2=0}^{n}\sum_{k_3=0}^{n} (g^s, g_1^s, g_2^s, g_3^s, g_{12}^s, g_{13}^s, g_{23}^s, g_{123}^s) \cdot (\widehat{q}, \widehat{q}_1, \widehat{q}_2, \widehat{q}_3, \widehat{q}_{12}, \widehat{q}_{13}, \widehat{q}_{23}, \widehat{q}_{123}), \tag{18}$$

where the modified charges are

$$\begin{aligned}
&\widehat{q} = \sum_{\mathbf{y}_j \in C} H_{k_1}^{y_{j1}} H_{k_2}^{y_{j2}} H_{k_3}^{y_{j3}} q_j, \quad \widehat{q}_{123} = \sum_{\mathbf{y}_j \in C} \widehat{H}_{k_1}^{y_{j1}} \widehat{H}_{k_2}^{y_{j2}} \widehat{H}_{k_3}^{y_{j3}} q_j, \\
&\widehat{q}_1 = \sum_{\mathbf{y}_j \in C} \widehat{H}_{k_1}^{y_{j1}} H_{k_2}^{y_{j2}} H_{k_3}^{y_{j3}} q_j, \quad \widehat{q}_2 = \sum_{\mathbf{y}_j \in C} H_{k_1}^{y_{j1}} \widehat{H}_{k_2}^{y_{j2}} H_{k_3}^{y_{j3}} q_j, \quad \widehat{q}_3 = \sum_{\mathbf{y}_j \in C} H_{k_1}^{y_{j1}} H_{k_2}^{y_{j2}} \widehat{H}_{k_3}^{y_{j3}} q_j, \\
&\widehat{q}_{12} = \sum_{\mathbf{y}_j \in C} \widehat{H}_{k_1}^{y_{j1}} \widehat{H}_{k_2}^{y_{j2}} H_{k_3}^{y_{j3}} q_j, \quad \widehat{q}_{13} = \sum_{\mathbf{y}_j \in C} \widehat{H}_{k_1}^{y_{j1}} H_{k_2}^{y_{j2}} \widehat{H}_{k_3}^{y_{j3}} q_j, \quad \widehat{q}_{23} = \sum_{\mathbf{y}_j \in C} H_{k_1}^{y_{j1}} \widehat{H}_{k_2}^{y_{j2}} \widehat{H}_{k_3}^{y_{j3}} q_j,
\end{aligned} \tag{19}$$

and $\mathbf{y}_j = (y_{j1}, y_{j2}, y_{j3})$ are the source particles. Note that in the exact particle-cluster interaction (17) the target $\mathbf{x}_i$ interacts with the source particles and charges $\{\mathbf{y}_j, q_j\}$, while in the particle-cluster approximation (18) the target $\mathbf{x}_i$ interacts with the Chebyshev grid points and modified charges $\{\mathbf{s_k}, \widehat{q}_\mathbf{k}\}$. It should also be noted that the Chebyshev grid points are mapped to the cluster C using the linear function mentioned above in the discussion of general intervals $[a, b]$.

The particle-cluster approximation has two advantages. First, the operation count for the exact interaction (17) is $O(N_c)$, where N_c is the number of particles in cluster C, while the operation count for the approximation (18) is $O(n^3)$, hence the approximation is faster when $n^3 << N_c$. Second, the modified charges are independent of the target $\mathbf{x}_i$, so they can be precomputed and reused in different particle-cluster approximations.

Treecode algorithm. The treecode algorithm has two stages. Stage 1 starts by building a hierarchical tree of particle clusters. The root cluster is a rectangular box containing all the particles. The root is bisected along its long edges, yielding 2, 4, or 8 child clusters, and these are similarly bisected until a cluster has fewer than a user-specified number of particles N_0. A long edge is defined to be any edge whose length exceeds $\ell_{max}/\sqrt{2}$, where ℓ_{max} is the maximum edge length of the cluster. Once the tree is built, the modified charges (19) for each cluster are computed; the procedure is explained below. This completes stage 1.

In stage 2, the treecode cycles through the target particles and computes particle-cluster interactions starting at the root. Well-separated particle-cluster interactions are computed using the approximation (18). The criterion for being well-separated is $r/R < \theta$, where r is the cluster radius, $R = |\mathbf{x}_i - \mathbf{y}_c|$ is the distance between the target particle and the cluster center, and θ is a user-specified parameter; this is the multipole acceptance criterion (MAC). If the MAC fails, then the code descends to the child clusters, until the leaves are reached at which point the particle-cluster interaction is computed by direct summation (17). This is essentially the Barnes-Hut treecode algorithm [1], extended to barycentric Hermite interpolation; references can be consulted for further details [18, 27].

Computation of modified charges. We explain some details of how the modified charges (19) for a given cluster are computed. Note that the Hermite basis functions appearing there can be written concisely in barycentric form (11) as

$$H_{k_\ell}^{y_{j\ell}} = H_{k_\ell}(y_{j\ell}) = \frac{c_{j,k_\ell,\ell}}{\sum_{k=0}^{n} c_{j,k,\ell}}, \quad c_{j,k_\ell,\ell} = \frac{a_{k_\ell}}{y_{j\ell} - s_{k_\ell}} + \frac{b_{k_\ell}}{(y_{j\ell} - s_{k_\ell})^2}, \tag{20a}$$

$$\widehat{H}_{k_\ell}^{y_{j\ell}} = \widehat{H}_{k_\ell}(y_{j\ell}) = \frac{d_{j,k_\ell,\ell}}{\sum_{k=0}^{n} c_{j,k,\ell}}, \quad d_{j,k_\ell,\ell} = \frac{b_{k_\ell}}{y_{j\ell} - s_{k_\ell}}, \tag{20b}$$

where (y_{j1}, y_{j2}, y_{j3}) are the source particles in the cluster and $(s_{k_1}, s_{k_2}, s_{k_3})$ are the Chebyshev points adapted to the cluster. First the temporary variables $c_{j,k,\ell}$, $d_{j,k,\ell}$ are computed; there are three loops, one loop over source particles $j = 1 : N_c$, one loop over Chebyshev points $k = 0 : n$, and one loop over coordinate indices $\ell = 1, 2, 3$; this requires $O(nN_c)$ operations and a storage array of size $O(nN_c)$ which is reused for different clusters. To handle the removable singularity, we adopt the procedure in [3] (page 510), see also [27]; the code checks whether a source particle coordinate is close to a Chebyshev point; the condition is $|y_{jl} - s_{k_\ell}| \le$ DBL_MIN, where DBL_MIN = 2.22507e-308 is the smallest positive IEEE double precision floating point number; if this condition is met, then the temporary variables $c_{j,k,\ell}$, $d_{j,k,\ell}$ are adjusted to enforce the interpolation conditions (8). Having computed the temporary variables $c_{j,k,\ell}$, $d_{j,k,\ell}$, the code computes the Hermite basis functions in (20), and then the modified charges are computed as indicated in (19). For each cluster this process requires $O(n^3 N_c)$ operations and $O(n^3)$ storage. The modified charges are computed for each cluster when the tree is constructed and in practice this requires only a small fraction of the total treecode run time.

Relation to other methods. We briefly discuss the relation of the barycentric Hermite treecode (BHTC) to some other tree-based fast summation methods. Makino [19] proposed using pseudoparticles in a treecode or FMM to reproduce the multipole moments of the actual particles in a cluster; in some sense the Chebyshev grid points in the BHTC can be considered as pseudoparticles, although their charges are determined by interpolating the kernel and its derivatives, rather than moment matching. Fong and Darve [9] developed the black-box FMM which uses polynomial interpolation at Chebyshev points of the 1st kind to approximate cluster-cluster interactions, where the interpolating polynomial is expressed in terms of Chebyshev polynomials; by contrast, the BHTC uses Hermite polynomial interpolation at Chebyshev points of the 2nd kind to approximate particle-cluster interactions, where the Hermite interpolating polynomial is expressed in barycentric form. Finally note that the BHTC is not strictly kernel-independent in the same sense as the KIFMM [28, 29] and bbFMM [9], which require only kernel evaluations, but for many kernels of practical interest the first partial derivatives needed in BHTC are easily computed. The methods mentioned here have various similarities and differences, and a detailed performance comparison is an interesting topic for future investigation.

5 Numerical results

The BHTC was programmed in C++ in double precision and compiled using the Intel compiler icpc with -O2 optimization. The computations were performed on the University of Wisconsin-Milwaukee Mortimer Faculty Research Cluster, where each node is a Dell PowerEdge R430 server with two 12-core Intel Xeon E5-2680 v3 processors at 2.50 GHz and 64 GB RAM. Parallel runs used OpenMP with up to 24 cores on a single node. The source code is available for download (github.com/Treecodes/barycentric-hermite-treecode). We start with a single particle-cluster interaction and proceed to the full treecode. The particle charges are random numbers in the interval $[-1, 1]$ and the screened Coulomb potential uses $\kappa = 0.1$ for the inverse Debye length.

5.1 A single particle-cluster interaction

Consider a target particle $\mathbf{x} = (1.214, 1.214, 1.214)$ and a cluster of $N = 1000$ randomly located source particles in the unit cube $[0, 1]^3$; this corresponds to MAC $\theta = 0.7$. The potential is computed at the target using Taylor approximation [18], barycentric Lagrange interpolation [27], and barycentric Hermite interpolation as described above. Figure 1 plots the error as a function of the degree parameter $n = 1 : 2 : 31$, where the exact value is computed by direct summation (17). Note that the degree parameter n has two different meanings; in the case of the Taylor approximation, the expansion is carried out at the cluster center $\mathbf{y}_c = (0.5, 0.5, 0.5)$ and includes terms $(\mathbf{x} - \mathbf{y}_c)^{\mathbf{k}}$ satisfying $k_1 + k_2 + k_3 \leq n$, whereas in the case of the barycentric approximations, n is the degree of the interpolating polynomial in each direction. Bearing this in mind, Fig. 1 shows that the error decreases most rapidly for barycentric Hermite interpolation, followed by barycentric Lagrange interpolation and then Taylor approximation. Results for the Coulomb potential and screened Coulomb potential are very close. Note however that the three approximations have different cost, so next we consider the error and run time in the full treecode.

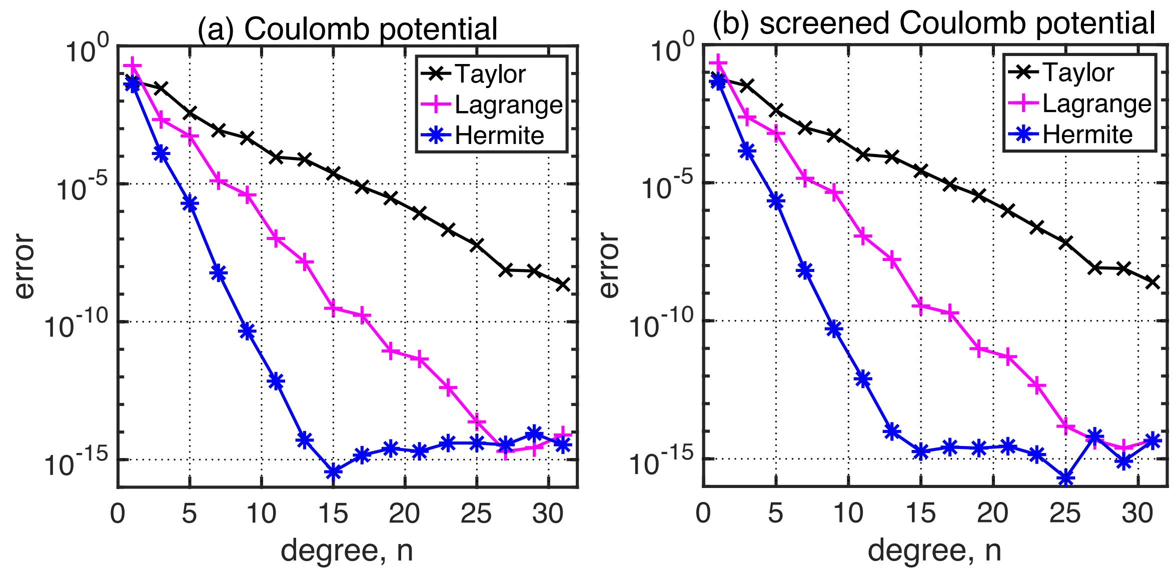

Figure 1: A single particle-cluster interaction, target particle at (1.214, 1.214, 1.214), cluster of $N = 1000$ particles in $[0, 1]^3$, MAC $\theta = 0.7$, error is plotted versus degree parameter $n = 1 : 2 : 31$ for Taylor approximation, barycentric Lagrange interpolation, barycentric Hermite interpolation, (a) Coulomb potential, (b) screened Coulomb potential.

5.2 Treecode results

We compute the electrostatic potential (1) for N charged particles randomly located in a cube. The maximum leaf size is $N_0 = 2000$. The error is measured by

$$E = \left(\sum_{i=1}^{N} |\phi^d(\mathbf{x}_i) - \phi^t(\mathbf{x}_i)|^2 \Big/ \sum_{i=1}^{N} |\phi^d(\mathbf{x}_i)|^2 \right)^{1/2}, \tag{21}$$

where $\phi^d(\mathbf{x}_i)$ is computed by direct summation and $\phi^t(\mathbf{x}_i)$ is the treecode approximation.

Run time versus error. Figure 2 compares the BHTC with treecodes based on Taylor approximation [18] and barycentric Lagrange interpolation [27]. Results are shown for the Coulomb and screened Coulomb potentials (a,b), with system size N=1e6. The run time is shown versus error for treecode MAC $\theta = 0.5, 0.7$ and degree n = 1:2:15 (T), n=1:2:9 (L,H). With these parameters, the Taylor and Lagrange treecodes reach error 1e-8, while the BHTC reaches error 1e-13. The direct sum run time is the red horizontal line (4306 s in (a), 8146 s in (b)), and all three treecodes are faster than direct summation. For both the direct sum and treecode, the run time for the screened Coulomb potential is roughly twice that for the Coulomb potential; this is attributed to the exponential function evaluation in the former. For a given error E, the Taylor and Lagrange treecodes have comparable run time, but the BHTC is faster than both, especially as the error decreases. Focusing on the BHTC, MAC $\theta = 0.7$ is more efficient for error above 1e-9 and MAC $\theta = 0.5$ is more efficient for error below 1e-9.

Error and run time versus system size. Figure 3 presents the BHTC error (a,b) and run time (c,d) for system size N=1e5, 1e6, 1e7, with MAC $\theta = 0.7$ and degree $n = 1 : 2 : 9$. Results are shown for the Coulomb (a,c) and screened Coulomb (b,d) potentials. Since the direct sum for the largest system size N=1e7 is prohibitively expensive, in that case the error is computed at 2000 sample target particles and the run time is extrapolated from the smaller systems. Figure 3a,b shows that for a given degree n, the treecode error is nearly independent of system size N, and for a given system size N, the error decreases as the degree n increases. The errors for the two potentials are comparable. Figure 3c,d shows that for a given degree n, the treecode run time increases at a slightly superlinear rate consistent with $O(N \log N)$ scaling. For a given system size N, the treecode run time increases as the degree n increases, but for these parameters it remains below the direct sum run time.

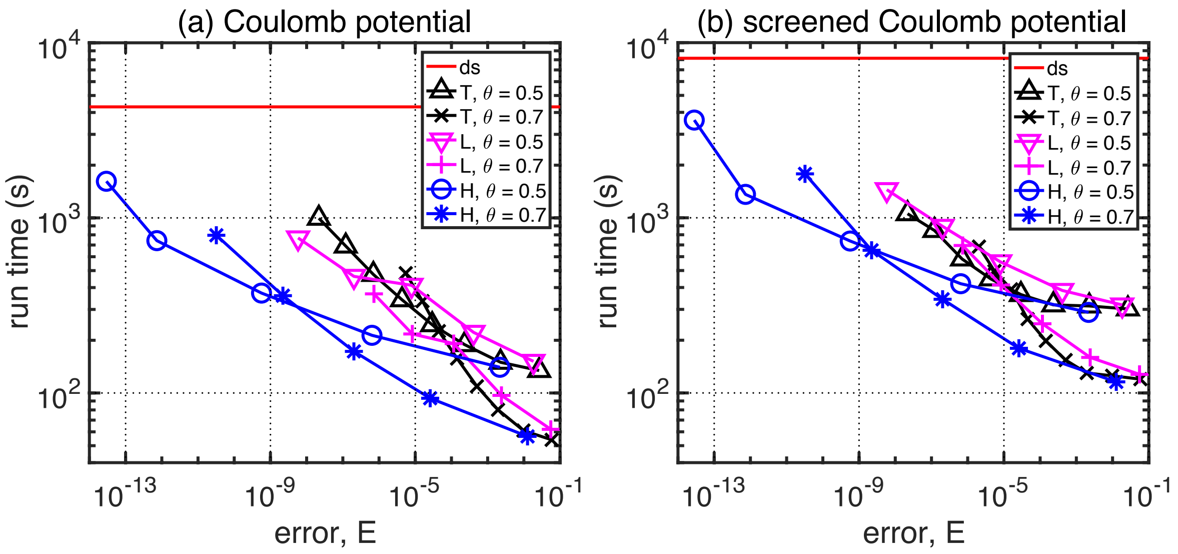

Figure 2: Comparison of treecodes based on Taylor approximation (T), barycentric Lagrange interpolation (L), barycentric Hermite interpolation (H), random particles in a cube, system size N=1e6, run time (s) versus error, (a) Coulomb potential, (b) screened Coulomb potential, direct sum run time (red horizontal line), treecode results use MAC $\theta = 0.5, 0.7$, degree n = 1:2:15 (T), n = 1:2:9 (L,H), degree increases from right to left.

Speedup. Table 1 quantifies the speedup defined as the ratio of run time for the direct sum (d) and BHTC (t). Results are shown for treecode parameters MAC $\theta = 0.7$ and degree $n = 3$, yielding error E between 7.65e-6 and 2.59e-5. The speedup is similar for the Coulomb potential and screened Coulomb potential, and it increases with system size; the speedup is approximately 7× for N=1e5, 45× for N=1e6, and 400× for N=1e7.

Table 1: Barycentric Hermite treecode, random particles in a cube, Coulomb potential, screened Coulomb potential, system size N, run time (s) for direct sum (d), treecode (t), speedup (d/t), error, E, treecode MAC $\theta = 0.7$, degree $n = 3$.

N	direct sum (s), d	treecode (s), t	speedup, d/t	error, E
		Coulomb potential		
1e5	42.62	5.62	7.58	7.65e-6
1e6	4305.93	93.31	46.15	2.58e-5
1e7	430593.00	1078.43	399.28	1.97e-5
		screened Coulomb potential		
1e5	81.13	11.07	7.33	8.36e-6
1e6	8146.08	179.87	45.29	2.59e-5
1e7	814608.00	1911.27	426.21	2.11e-5

Memory consumption. Table 2 records the memory (MB) consumed by direct summation and the three treecodes (TTC, BLTC, BHTC); this was obtained using the Valgrind Massif tool (www.valgrind.org). Results are shown for the Coulomb potential with MAC $\theta = 0.7$, but the same results hold for the screened Coulomb potential and different θ.

Table 2 shows how the memory consumption depends on the system size N and degree n; these parameters contribute as follows. First, the particle coordinates and charges are stored in arrays of size $O(N)$; this memory is common to all methods and can be seen in the direct sum column in Table 2. Second, the treecodes require memory for the tree data structure. A certain portion of this memory is overhead common to all three treecodes and is $O(N)$; this accounts for the increase between the direct sum column and the $n = 1$ column in

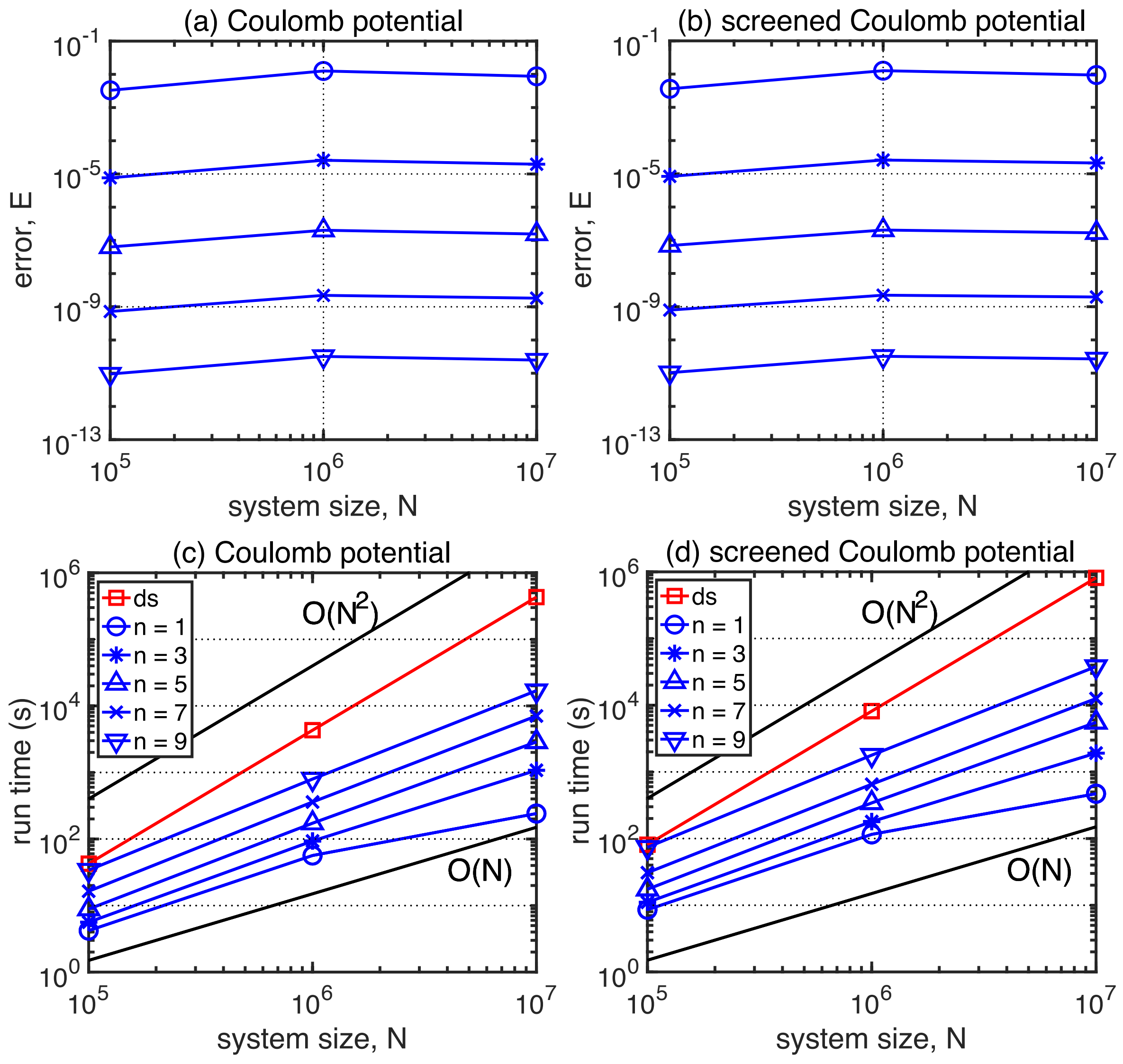

Figure 3: Barycentric Hermite treecode, random particles in a cube, system size N=1e5, 1e6, 1e7, (a,c) Coulomb potential, (b,d) screened Coulomb potential, treecode MAC $\theta = 0.7$, degree $n = 1 : 2 : 9$, (a,b) error, (c,d) run time, direct sum run time (red line).

Table 2. The treecodes differ however in the memory required for each cluster. The TTC stores cluster moments in arrays of asymptotic size $\frac{1}{6}n^3$; the factor $\frac{1}{6}$ arises from the sum $k_1 + k_2 + k_3 \leq n$. The BLTC stores modified charges in arrays of asymptotic size n^3; this is due to the tensor product nature of the Chebyshev grid points. The BHTC stores modified charges in arrays of asymptotic size $8n^3$; the factor 8 arises because there are eight modified charges at each Chebyshev grid point as shown in Eq. (19). Hence the treecode memory consumption is $O(N) + O(Nn^3)$, where the first term is the particle memory and tree data structure overhead, and the second term is the cluster memory with a prefactor depending on the treecode version.

These trends can be seen in Table 2. For degree $n = 1$ all three treecodes use the same memory to within two decimal digits; in this case the $O(N)$ particle/overhead memory dominates the $O(Nn^3)$ cluster memory. As the degree n increases, the TTC memory remains the same, while the BLTC memory increases, and the BHTC memory increases even more; in this case the $O(Nn^3)$ cluster memory eventually dominates the $O(N)$ particle/overhead memory. We checked that for larger degree n, the TTC memory also shows the expected $O(Nn^3)$ increase. Note that even with the largest system size N=1e7 and degree $n = 9$, the BHTC still uses less than 7 GB of memory. The larger memory consumption of the BHTC in comparison with the BLTC and TTC is compensated by its better performance as shown in Fig. 2.

Table 2: Memory consumption (MB), random particles in a cube, system size N = 1e5, 1e6, 1e7, results shown for direct sum, Taylor treecode (TTC), barycentric Lagrange treecode (BLTC), barycentric Hermite treecode (BHTC), treecode degree $n = 1, 3, 5, 7, 9$.

N	direct sum	$n = 1$	$n = 3$	$n = 5$	$n = 7$	$n = 9$
		TTC				
1e5	5.60	7.45	7.45	7.45	7.45	7.45
1e6	56.02	72.42	72.42	72.42	72.42	72.42
1e7	560.20	782.67	782.67	782.67	782.67	782.67
		BLTC				
1e5	5.60	7.45	7.45	7.45	7.56	7.65
1e6	56.20	72.42	72.42	72.42	72.42	81.81
1e7	560.20	782.67	782.67	782.67	995.05	1383.54
		BHTC				
1e5	5.60	7.45	7.56	8.57	12.23	18.24
1e6	56.02	72.42	72.52	99.84	158.33	254.57
1e7	560.20	782.67	995.05	1960.78	3832.49	6912.16

Parallel simulations. Table 3 records parallel results for the direct sum and BHTC using OpenMP with up to 24 cores on a single node. The system size is N=1e6 and the treecode uses MAC $\theta = 0.7$, degree $n = 3$, yielding error about 2.6e-5. As before, the run time for the screened Coulomb potential is about twice that for the Coulomb potential. Direct summation parallelizes very well; the parallel efficiency remains above 86% up to 24 cores. The treecode also parallelizes well considering its complexity; with 24 cores the parallel efficiency is 61% for the Coulomb potential and 68% for the screened Coulomb potential. With 1 core the treecode is 45× faster than direct summation, and even with 24 cores the treecode is still more than 32× faster than direct summation.

6 Conclusions

We presented a particle-cluster treecode based on barycentric Hermite interpolation (BHTC) for fast summation of electrostatic particle interactions in 3D. The interpolating polynomial matches the kernel and its first partial derivatives in the three coordinate directions at a tensor product grid of Chebyshev points in each cluster. Barycentric Hermite interpolation is scale-invariant in a certain sense that facilitates the treatment of different clusters in the treecode and this promotes the scheme's efficiency. The treecode run time scales like $O(N \log N)$, where N is the number of particles in the system. We demonstrated the method for the Coulomb potential and screened Coulomb potential. Numerical results show that for these two kernels, the BHTC is more efficient than the Taylor treecode [18] and barycentric Lagrange treecode [27] especially in the high accuracy regime. The parallel efficiency of the BHTC was demonstrated using up to 24 cores on a single node.

There are several directions for future research. In principle the BHTC can be extended to interpolate higher-order derivatives, but it remains to be seen whether this is advantageous in practice. Although we demonstrated the method for electrostatic potentials, it can be extended to compute electrostatic forces arising from Coulomb and screened Coulomb potentials, as needed in molecular dynamics simulations. It would also be interesting to consider whether the BHTC is advantageous for more complicated kernels such as those arising in the Method of Regularized Stokeslets [5, 6, 20]. We plan to install the BHTC in the

Table 3: Barycentric Hermite treecode parallel performance, random particles in a cube, system size N=1e6, treecode parameters MAC $\theta = 0.7$, degree $n = 3$, Coulomb potential (error E = 2.58e-5), screened Coulomb potential (error E = 2.59e-5), number of cores (nc), run time (s) is shown for direct sum and treecode (d, t), parallel speedup = d_1/d_{nc}, t_1/t_{nc}, parallel efficiency (PE %) = $(d_1/d_{nc})/nc$, $(t_1/t_{nc})/nc$, parallel treecode speedup (d/t).

nc	d time (s)	d_1/d_{nc}	d PE (%)	t time (s)	t_1/t_{nc}	t PE (%)	d/t
			Coulomb potential				
1	4305.07	1.00	100.00	93.62	1.00	100.00	45.98
2	2167.15	1.99	99.33	47.32	1.98	98.92	45.80
4	1093.24	3.94	98.45	24.41	3.84	95.88	44.79
8	599.26	7.18	89.80	13.76	6.80	85.05	43.55
12	412.14	10.45	87.05	10.25	9.13	76.11	40.21
24	206.31	20.87	86.95	6.35	14.74	61.43	32.49
			screened Coulomb potential				
1	8159.46	1.00	100.00	180.11	1.00	100.00	45.30
2	4082.25	1.00	99.94	91.13	1.98	98.82	44.80
4	2053.37	3.97	99.34	46.40	3.88	97.04	44.25
8	1131.98	7.21	90.10	25.94	6.97	86.79	43.63
12	765.24	10.66	88.86	19.02	9.47	78.91	40.23
24	383.77	21.26	88.59	11.00	16.37	68.22	34.89

treecode-accelerated boundary integral (TABI) Poisson-Boltzmann solver [10] and the Adaptive Poisson-Boltzmann Solver (APBS) [17] for electrostatics of solvated biomolecules. Work is also proceeding on a GPU implementation of the BHTC [26].

Acknowledgments: We thank the reviewers for suggestions that helped improve the article. We are grateful for a computer allocation on the University of Wisconsin-Milwaukee Mortimer Faculty Research Cluster. Partial support is acknowledged from start-up funds provided by the University of Wisconsin-Milwaukee, NSF grant DMS-1819094, the Michigan Institute for Computational Discovery and Engineering (MICDE), and the Mcubed program at the University of Michigan.

References

[1] J.E. Barnes, P. Hut, A hierarchical $O(N \log N)$ force-calculation algorithm, Nature **324** (1986), 446–449.
[2] E. Berriochoa, A. Cachafeiro, J. Díaz, J. Illán, Algorithms and convergence for Hermite interpolation based on extended Chebyshev nodal systems, Appl. Math. Comput. **234** (2014), 223–236.
[3] J.-P. Berrut, L.N. Trefethen, Barycentric Lagrange interpolation, SIAM Rev. **46** (2004), 501–517.
[4] G.A. Cisneros, M. Karttunen, P. Ren, C. Sagui, Classical electrostatics for biomolecular simulations, Chem. Rev. **114** (2014), 779–814.
[5] R. Cortez, The method of regularized Stokeslets, SIAM J. Sci. Comput., **23** (2001), 1204–1225.
[6] R. Cortez, L. Fauci, A. Medovikov, The method of regularized Stokeslets in three dimensions: Analysis, validation, and application to helical swimming, Phys. Fluids **17** (2005), 031504.
[7] M.E. Davis, J.A. McCammon, Electrostatics in biomolecular structure and dynamics, Chem. Rev. **90** (1990), 509–521.
[8] U. Essmann, L. Perera, M. Berkowitz, T. Darden, H. Lee, L. Pedersen, A smooth particle mesh Ewald method, J. Chem. Phys. **103** (1995), 8577–8593.
[9] W. Fong, E. Darve, The black-box fast multipole method, J. Comput. Phys. **228** (2009), 8712–8725.
[10] W.-H. Geng, R. Krasny, A treecode-accelerated boundary integral Poisson-Boltzmann solver for solvated biomolecules, J. Comput. Phys. **247** (2013), 62–78.
[11] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. **73** (1987), 325–348.
[12] W. Hackbusch, Z.P. Nowak, On the fast matrix multiplication in the boundary element method by panel clustering, Numer. Math. **54** (1989), 463–491.

[13] D.J. Hardy, Z. Wu, J.C. Phillips, J.E. Stone, R.D. Skeel, K. Schulten, Multilevel summation method for electrostatic force evaluation, J. Chem. Theory Comput. **11** (2015), 766–779.

[14] N.J. Higham, The numerical stability of barycentric Lagrange interpolation, IMA J. Numer. Anal. **24** (2004), 547–556.

[15] R.W. Hockney, J.W. Eastwood, Computer Simulation Using Particles, Taylor & Francis, Bristol (1988).

[16] B. Honig, A. Nicholls, Classical electrostatics in biology and chemistry, Science **268** (1995), 1144–1149.

[17] E. Jurrus, D. Engel, K. Star, K. Monson, J. Brandi, L.E. Felberg, D.H. Brookes, L. Wilson, J. Chen, K. Liles, M. Chun, P. Li, D.W. Gohara, T. Dolinsky, R. Konecny, D.R. Koes, J.E. Nielsen, T. Head-Gordon, W.H. Geng, R. Krasny, G.-W. Wei, M.J. Holst, J.A. McCammon, N.A. Baker, Improvements to the APBS biomolecular solvation software suite, Protein Science 27 (2018) 112–128.

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

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

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

[21] B. Sadiq, D. Viswanath, Barycentric Hermite interpolation, SIAM J. Sci. Comput. **35** (2013), A1254–A1270.

[22] H.E. Salzer, Lagrangian interpolation at the Chebyshev points $x_{n,\nu} = \cos(\nu\pi/n)$, $\nu = 0(1)n$; some unnoted advantages, Comput. J. **15** (1972), 156–159.

[23] T. Schlick, Molecular Modeling and Simulation: An Interdisciplinary Guide, Springer, New York (2010), 2nd edition.

[24] C. Schneider, W. Werner, Hermite interpolation: The Barycentric approach, Computing **46** (1991), 35–51.

[25] L.N. Trefethen, Approximation Theory and Approximation Practice, SIAM, Philadelphia (2013).

[26] N. Vaughn, L. Wilson, L.Wang, R. Krasny, GPU-accelerated barycentric treecodes, in preparation

[27] L. Wang, R. Krasny, S. Tlupova, A kernel-independent treecode based on barycentric Lagrange interpolation, arXiv:1902.02250

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

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

[30] Z. Zhang, S, Witham, E. Alexov, On the role of electrostatics in protein-protein interactions, Phys. Biol. **8** (2011), 035001.

[31] H.-X. Zhou, X. Pang, Electrostatic interactions in protein structure, folding, binding, and condensation, Chem. Rev. **118** (2018), 1691–1741.