Journal of Computational Physics 228 (2009) 3858–3868

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A Cartesian treecode for screened coulomb interactions

Peijun Li [a,*,1], Hans Johnston [b,2], Robert Krasny [c,2,3]

[a] Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907, United States
[b] Department of Mathematics and Statistics, University of Massachusetts, Amherst, MA 01003, United States
[c] Department of Mathematics, University of Michigan, Ann Arbor, MI 48109, United States

ARTICLE INFO

Article history:
Received 10 August 2008
Received in revised form 5 February 2009
Accepted 16 February 2009
Available online 28 February 2009

MSC:
65Z05
82-08

PACS:
02.70.Ns
02.30.Mv

Keywords:
Screened coulomb potential
Yukawa potential
Cartesian treecode
Particle–cluster interaction
Recurrence relations

ABSTRACT

A treecode algorithm is presented for evaluating electrostatic potentials in a charged particle system undergoing screened Coulomb interactions in 3D. The method uses a far-field Taylor expansion in Cartesian coordinates to compute particle–cluster interactions. The Taylor coefficients are evaluated using new recurrence relations which permit efficient computation of high order approximations. Two types of clusters are considered, uniform cubes and adapted rectangular boxes. The treecode error, CPU time and memory usage are reported and compared with direct summation for randomly distributed particles inside a cube, on the surface of a sphere and on an 8-sphere configuration. For a given order of Taylor approximation, the treecode CPU time scales as $O(N \log N)$ and the memory usage scales as $O(N)$, where N is the number of particles. Results show that the treecode is well suited for non-homogeneous particle distributions as in the sphere and 8-sphere test cases.

Published by Elsevier Inc.

1. Introduction

Electrostatic effects induced by charged particle systems are important in many areas of biology, chemistry and physics [16]. In the case of a dielectric medium, a common form of pairwise interaction is the screened Coulomb potential,

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

where $\mathbf{x},\mathbf{y} \in \mathbb{R}^3$ and $\kappa \geqslant 0$ is the Debye–Hückel parameter. We interpret $\mathbf{x}$ as a target point and $\mathbf{y}$ as a source point. The Debye–Hückel parameter is defined by

* Corresponding author.
E-mail addresses: lipeijun@math.purdue.edu (P. Li), johnston@math.umass.edu (H. Johnston), krasny@umich.edu (R. Krasny).

[1] The research was supported in part by a University of Michigan Research Fellowship and NSF Grant EAR-0724656.
[2] The research was supported in part by Michigan Life Sciences Corridor Grant #1515.
[3] The research was supported in part by NSF Grants DMS-0510162, ATM-0723440.

0021-9991/$ - see front matter Published by Elsevier Inc.
doi:10.1016/j.jcp.2009.02.022

$$\kappa^2 = \frac{8\pi q^2 I}{\epsilon k_B T}, \tag{2}$$

where q and I are the ionic charge and concentration in the medium, ϵ is the dielectric constant, k_B is the Boltzmann constant and T is the temperature. For example in the case of a solvent, $\kappa > 0$ accounts for the presence of dissolved electrolytes which tend to weaken the electric field.

The screened Coulomb potential (1) arises in many applications. It is the free-space Green's function of the linear Poisson–Boltzmann equation,

$$\Delta\phi - \kappa^2\phi = 0, \tag{3}$$

which is used in implicit solvent models for biomolecular simulations [1,21]. The same potential function appears in studies of DNA structure [24] and dusty plasmas [3,17,27] and it is known as the Yukawa potential in nuclear physics where it describes the binding of nucleons by meson exchange [6].

Here we consider systems of particles with locations $\mathbf{x}_i$ and charges q_i, for $i = 1 : N$, interacting through Eq. (1). In this case, the electrostatic potential at a given particle $\mathbf{x}_i$ due to all the other particles is

$$V_i = \sum_{j=1, j\neq i}^{N} q_j \phi(\mathbf{x}_i, \mathbf{x}_j). \tag{4}$$

These potentials are used to investigate basic properties of the system such as conformational stability and binding affinity and their efficient computation is a major concern. The cost of evaluating V_i for $i = 1 : N$ by direct summation is $O(N^2)$, which is prohibitively expensive when N is large. Despite the exponential decay in Eq. (1), simply truncating the potential outside some radius is not a viable option in applications where the values of κ and $|\mathbf{x} - \mathbf{y}|$ are not sufficiently large. As a result, various more sophisticated methods have been developed to reduce the cost of evaluating sums like Eq. (4). For example in the case $\kappa = 0$, particle-mesh methods transfer charge from the particles to a regular mesh where the sums can be rapidly computed using the fast Fourier transform (FFT) [15].

The present work is concerned with an alternative class of tree-based fast summation methods that avoid using a mesh. These include the Barnes–Hut treecode [2] and the Greengard–Rokhlin Fast Multipole Method (FMM) [12–14]. Tree-based methods divide the particles into a hierarchy of clusters having a tree structure. In a treecode, the particle–particle interactions are replaced by particle–cluster interactions which are evaluated using either a far-field multipole expansion or direct summation. The FMM is a more elaborate procedure that uses near-field and far-field expansions. In principle, for a given order of expansion, treecodes require $O(N \log N)$ execution time and the FMM requires $O(N)$ execution time. In practice, several factors affect the performance including the number of levels in the tree, the density of the particle distribution, and the cache size of the computer. Aside from accuracy and execution time, different implementations can be compared in terms of memory requirements, coding complexity and parallelizability. Optimizing these methods is still an active area of research.

Most tree-based fast summation methods for electrostatics have dealt with the case $\kappa = 0$, where classical expansions of the Coulomb potential are available. For example, the original FMM used spherical harmonics expansions [12] and the new version of the FMM incorporates plane-wave expansions as well [7]. In order to extend these methods to the case $\kappa > 0$, it is necessary to identify suitable expansions of the screened Coulomb potential. This was recently accomplished for the FMM using modified spherical Bessel functions [4] and an appropriate plane-wave expansion [13]. These developments led to FMM-accelerated boundary element simulations of the Poisson–Boltzmann equation [5,19,20]. A key goal of research in this area is to develop an efficient Poisson–Boltzmann solver which can be used in each timestep of a molecular dynamics simulation [25]. Another suitable approach for the case $\kappa > 0$ is the kernel-independent FMM which uses equivalent particle densities in place of analytic series expansions [29].

The FMM is a relatively complicated algorithm and it is worthwhile to investigate simpler alternatives. Here we present a particle–cluster treecode for screened Coulomb interactions using a far-field Taylor expansion in Cartesian coordinates. Several Cartesian FMMs have been developed for the case $\kappa = 0$, for example [30,8,28,26], but we are not aware of any extensions to the case $\kappa > 0$. One challenge in using Cartesian Taylor series for either a treecode or FMM is the computational complexity of explicit formulas for higher derivatives of the potential function $\phi(\mathbf{x}, \mathbf{y})$, but we avoid this problem by deriving new recurrence relations for the Taylor coefficients of the screened Coulomb potential. The recurrence relations are valid for $\kappa \geqslant 0$ and they enable us to evaluate the Taylor approximations in a simple and efficient manner. This approach was previously employed in particle simulations involving several other potential functions including regularized Biot-Savart kernels [9,22,18], the real-space potential in Ewald summation [10] and power law potentials [11]. After describing the treecode algorithm, we present results for $\kappa = 0$ and $\kappa = 1$ for three test cases, randomly distributed particles inside a cube, on the surface of a sphere and on an 8-sphere configuration. We also compare two types of particle clusters, uniform cubes on each level as in the original treecode [2] and adapted rectangular boxes [18]. For a given order of Taylor approximation, the treecode CPU time scales as $O(N \log N)$ and the memory usage scales as $O(N)$. The version using adapted rectangular boxes is well suited for non-homogeneous particle distributions as in the sphere and 8-sphere test cases.

In Section 2 we review how the treecode uses particle–cluster interactions to evaluate electrostatic potentials in particle system. In Section 3 we derive recurrence relations for the Taylor coefficients of the screened Coulomb potential with respect

to Cartesian coordinates. In Section 4 we give details of the code implementation. Numerical results are presented in Section 5 and a summary is given in Section 6.

2. Particle–cluster interactions

In this section we review how the treecode uses particle–cluster interactions to evaluate electrostatic potentials in a particle system [2,11]. Assume the particles have been divided into a hierarchy of clusters (the procedure will be described later). The treecode evaluates the potential in Eq. (4) as a sum of particle–cluster interactions,

$$V_i = \sum_c V_{i,c}, \tag{5}$$

where

$$V_{i,c} = \sum_{\mathbf{y}_j \in c} q_j \phi(\mathbf{x}_i, \mathbf{y}_j) \tag{6}$$

is the interaction between a target particle $\mathbf{x}_i$ and a cluster of sources $c = \{\mathbf{y}_j\}$. A particle–cluster interaction is shown schematically in Fig. 1, where $\mathbf{y}_c$ is the cluster center, R is the particle–cluster distance and r_c is the cluster radius.

If particle $\mathbf{x}_i$ and cluster c are well-separated, the terms in Eq. (6) can be expanded in a Taylor series with respect to $\mathbf{y}$ about $\mathbf{y}_c$,

$$\phi(\mathbf{x}_i, \mathbf{y}_j) = \sum_{||\mathbf{k}||=0}^{\infty} \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \phi(\mathbf{x}_i, \mathbf{y}_c)(\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}}, \tag{7}$$

where Cartesian multi-index notation has been used with $\mathbf{k} = (k_1, k_2, k_3),\ k_i \in \mathbb{N},\ ||\mathbf{k}|| = k_1 + k_2 + k_3,\ \mathbf{k}! = k_1!k_2!k_3!,$ $\mathbf{y} = (y_1, y_2, y_3),\ y_i \in \mathbb{R},\ \mathbf{y}^{\mathbf{k}} = y_1^{k_1} y_2^{k_2} y_3^{k_3},\ D_{\mathbf{y}}^{\mathbf{k}} = D_{y_1}^{k_1} D_{y_2}^{k_2} D_{y_3}^{k_3}$. The Taylor series (7) converges for $R > r_c$ and hence it plays the role of a far-field expansion in the treecode, analogous to the far-field spherical harmonics multipole expansion in the FMM. Substituting (7) into (6) yields

$$V_{i,c} = \sum_{\mathbf{y}_j \in c} q_j \sum_{||\mathbf{k}||=0}^{\infty} \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \phi(\mathbf{x}_i, \mathbf{y}_c)(\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} \tag{8}$$

$$= \sum_{||\mathbf{k}||=0}^{\infty} \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \phi(\mathbf{x}_i, \mathbf{y}_c) \sum_{\mathbf{y}_j \in c} q_j (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} \tag{9}$$

$$\approx \sum_{||\mathbf{k}||=0}^{p} a^{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_c) m_c^{\mathbf{k}}, \tag{10}$$

where

$$a^{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_c) = \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \phi(\mathbf{x}_i, \mathbf{y}_c), \tag{11}$$

is the $\mathbf{k}$th Taylor coefficient of the screened Coulomb potential (1),

$$m_c^{\mathbf{k}} = \sum_{\mathbf{y}_j \in c} q_j (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}}, \tag{12}$$

is the $\mathbf{k}$th moment of cluster c and the Taylor series has been truncated at order p. Note that the Taylor coefficients $a^{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_c)$ are independent of the sources $\mathbf{y}_j$ in cluster c and the cluster moments $m_c^{\mathbf{k}}$ are independent of the target $\mathbf{x}_i$. These features lead to a savings in execution time.

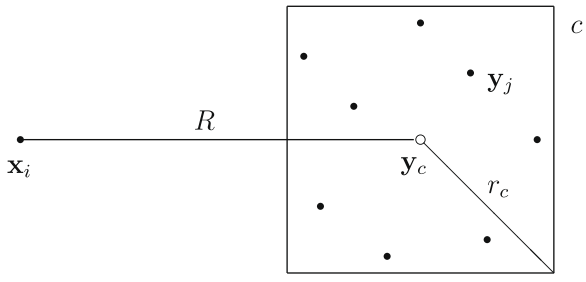

Fig. 1. Schematic of particle–cluster interaction between particle $\mathbf{x}_i$ and cluster $c = \{\mathbf{y}_j\}$; $\mathbf{y}_c$: cluster center, R: particle–cluster distance and r_c: cluster radius.

The treecode has two options for computing a particle–cluster interaction $V_{i,c}$. It can use direct summation as in the definition (6), or Taylor approximation as in (10). In practice, the Taylor approximation is used if the following multipole acceptance criterion (MAC) is satisfied,

$$\frac{r_c}{R} \leqslant \theta, \tag{13}$$

where θ is a user-specified parameter for controlling the error [2,23]. If the MAC is not satisfied, the code examines the children of cluster c, or it performs direct summation if c is a leaf of the tree.

Here we focus on the problem of evaluating the electrostatic potential V_i, but similar considerations apply to the electric field $E_i = -\nabla V_i$ and hence the treecode can be applied to compute electrostatic forces as well. In the next section we derive recurrence relations for evaluating the Taylor coefficients.

3. Recurrence relations for Taylor coefficients

Explicit formulas for the Taylor coefficients of the screened Coulomb potential (11) are cumbersome to work with. Instead we derive recurrence relations that permit simple and efficient computation of these coefficients to high order. It is convenient to define an auxiliary function and its Taylor coefficients,

$$\psi(\mathbf{x},\mathbf{y}) = e^{-\kappa|\mathbf{x}-\mathbf{y}|}, \quad b^{\mathbf{k}}(\mathbf{x},\mathbf{y}) = \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \psi(\mathbf{x},\mathbf{y}). \tag{14}$$

In the remainder of this section we omit the arguments $(\mathbf{x},\mathbf{y})$ for clarity. We will show that the Taylor coefficients $a^{\mathbf{k}}, b^{\mathbf{k}}$ satisfy the recurrence relations,

$$||\mathbf{k}||\mathbf{x}-\mathbf{y}|^2 a^{\mathbf{k}} - (2||\mathbf{k}||-1)\sum_{i=1}^{3}(x_i-y_i)a^{\mathbf{k}-\mathbf{e}_i} + (||\mathbf{k}||-1)\sum_{i=1}^{3} a^{\mathbf{k}-2\mathbf{e}_i} = \kappa\left(\sum_{i=1}^{3}(x_i-y_i)b^{\mathbf{k}-\mathbf{e}_i} - \sum_{i=1}^{3} b^{\mathbf{k}-2\mathbf{e}_i}\right), \tag{15}$$

$$||\mathbf{k}||b^{\mathbf{k}} = \kappa\left(\sum_{i=1}^{3}(x_i-y_i)a^{\mathbf{k}-\mathbf{e}_i} - \sum_{i=1}^{3} a^{\mathbf{k}-2\mathbf{e}_i}\right), \tag{16}$$

for $||\mathbf{k}|| \geqslant 2$, where $\mathbf{e}_i$ are the Cartesian basis vectors. Before proceeding to the derivation, note that even though the recurrence relations are coupled, the structure is such that they can be solved by explicit marching; the values of $a^{\mathbf{k}}, b^{\mathbf{k}}$ for $||\mathbf{k}|| = 0, 1$ are computed from the definitions and then the recurrence relations can be applied to compute the coefficients for $||\mathbf{k}|| \geqslant 2$.

First we derive the recurrence relation for $a^{\mathbf{k}}$. From Eqs. (1) and (14) we obtain

$$|\mathbf{x}-\mathbf{y}|\phi = \psi. \tag{17}$$

Then applying D_{y_1} yields

$$|\mathbf{x}-\mathbf{y}|^2 D_{y_1}\phi - (x_1-y_1)\phi = \kappa(x_1-y_1)\psi. \tag{18}$$

Next applying $D_{y_1}^{k_1-1}$ and using Leibnitz's rule for repeated differentiation of a product yields

$$|\mathbf{x}-\mathbf{y}|^2 D_{y_1}^{k_1}\phi - (2k_1-1)(x_1-y_1)D_{y_1}^{k_1-1}\phi + (k_1-1)^2 D_{y_1}^{k_1-2}\phi = \kappa((x_1-y_1)D_{y_1}^{k_1-1}\psi - (k_1-1)D_{y_1}^{k_1-2}\psi). \tag{19}$$

Next applying $D_{y_2}^{k_2} D_{y_3}^{k_3}$ yields

$$\begin{aligned} &|\mathbf{x}-\mathbf{y}|^2 D_{\mathbf{y}}^{\mathbf{k}}\phi - 2\sum_{i=1}^{3} k_i(x_i-y_i)D_{\mathbf{y}}^{\mathbf{k}-\mathbf{e}_i}\phi + \sum_{i=1}^{3} k_i(k_i-1)D_{\mathbf{y}}^{\mathbf{k}-2\mathbf{e}_i}\phi + (x_1-y_1)D_{\mathbf{y}}^{\mathbf{k}-\mathbf{e}_1}\phi - (k_1-1)D_{\mathbf{y}}^{\mathbf{k}-2\mathbf{e}_1}\phi \\ &\quad = \kappa((x_1-y_1)D_{\mathbf{y}}^{\mathbf{k}-\mathbf{e}_1} - (k_1-1)D_{\mathbf{y}}^{\mathbf{k}-2\mathbf{e}_1}). \end{aligned} \tag{20}$$

Dividing by $\mathbf{k}!$ and substituting the definitions of $a^{\mathbf{k}}, b^{\mathbf{k}}$ yields

$$|\mathbf{x}-\mathbf{y}|^2 a^{\mathbf{k}} - 2\sum_{i=1}^{3}(x_i-y_i)a^{\mathbf{k}-\mathbf{e}_i} + \sum_{i=1}^{3} a^{\mathbf{k}-2\mathbf{e}_i} + \frac{1}{k_1}((x_1-y_1)a^{\mathbf{k}-\mathbf{e}_1} - a^{\mathbf{k}-2\mathbf{e}_1}) = \frac{\kappa}{k_1}((x_1-y_1)b^{\mathbf{k}-\mathbf{e}_1} - b^{\mathbf{k}-2\mathbf{e}_1})). \tag{21}$$

Index 1 plays a special role in Eq. (21) and two similar equations are obtained by permuting indices. Multiplying these equations by k_1, k_2, k_3, respectively and summing the results yields (15). In practice, Eq. (21) is used instead of the symmetric form (15) to gain a slight reduction in CPU time.

Regarding the coefficients $b^{\mathbf{k}}$, explicit calculation shows that

$$D_{y_1}\psi = \kappa(x_1-y_1)\phi, \tag{22}$$

and then following the same steps as above yields Eq. (16). This completes the derivation of the recurrence relations for the Taylor coefficients of the screened Coulomb potential.

Note that in the limit $\kappa \to 0$, the recurrence relations (15) and (16) reduce to

$$||\mathbf{k}||\mathbf{x}-\mathbf{y}|^2 a^{\mathbf{k}} - (2||\mathbf{k}|| - 1)\sum_{i=1}^{3}(x_i - y_i)a^{\mathbf{k}-\mathbf{e}_i} + (||\mathbf{k}|| - 1)\sum_{i=1}^{3} a^{\mathbf{k}-2\mathbf{e}_i} = 0, \tag{23}$$

$$||\mathbf{k}||b^{\mathbf{k}} = 0. \tag{24}$$

Eq. (23) is the recurrence relation for the Taylor coefficients of the Coulomb potential (this can be seen by setting $\kappa = 0$ in the previous derivation) and Eq. (24) is a simple consequence of the fact that $\psi(\mathbf{x}, \mathbf{y}) \to 1$ as $\kappa \to 0$. Hence the recurrence relations (15) and (16) remain valid in the limit $\kappa \to 0$.

We conclude this section by reporting the accuracy of the Taylor approximation for a sample particle–cluster interaction. Consider a target particle $\mathbf{x} = (-0.5, -0.5, -0.5)$ and a cluster of $N = 10^3$ source particles randomly distributed in the cube $[0, 0.5]^3$ with random charges $|q_i| \leqslant 0.5$. The relative error in potential is defined by

$$E_1 = \frac{|V - \widehat{V}|}{|V|}, \tag{25}$$

where V is the exact value given by direct summation (6) and $\widehat{V}$ is the Taylor approximation (10). The Taylor coefficients are computed using the recurrence relations derived above. Fig. 2 presents E_1 as a function of the screening parameter, $0 \leqslant \kappa \leqslant 5$, and the approximation order, $0 \leqslant p \leqslant 20$. In this case we have $r_c/R = 1/3$, so the Taylor series converges, but the error is non-montonic with respect to order p (this was previously observed for the Coulomb potential [30]). Note that the error E_1 apparently vanishes for some special values of κ and p (there are four occurrences in Fig. 2), but aside from these values some general trends can be seen. For a given order p, the error generally increases as κ increases. For example with $p = 6$, the error is 4.73e−6 for $\kappa = 0$ and 4.31e−3 for $\kappa = 5$; see circles ($\circ$) in Fig. 2. Conversely, for a given value of κ, the error generally decreases as the order p increases. For example with $\kappa = 1$, the error is 2.89e−1 for $p = 0$ and 1.71e−13 for $p = 20$. In summary, the results show that the Taylor approximation (10) can be used to evaluate well-separated particle–cluster interactions of the screened Coulomb potential to a required level of accuracy.

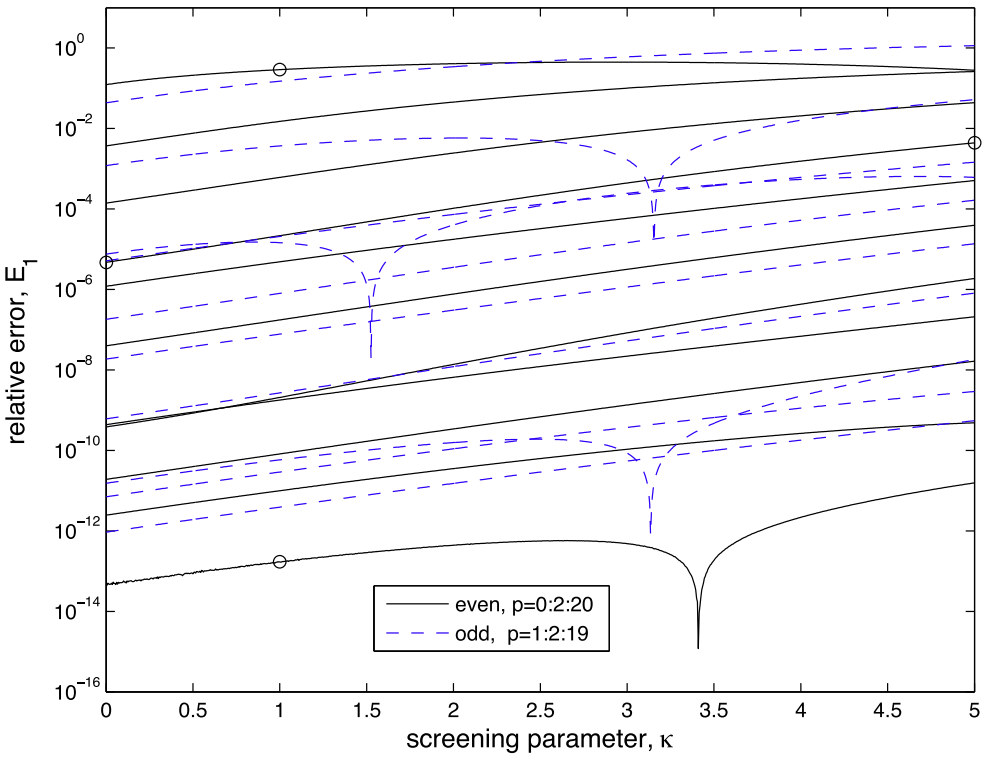

Fig. 2. Relative error in particle–cluster potential E_1, Eq. (25), computed by Taylor approximation (10), as a function of screening parameter κ and approximation order p; target particle $\mathbf{x} = (-0.5, -0.5, -0.5)$, cluster of $N = 10^3$ source particles randomly distributed in the cube $[0.0, 0.5]^3$ with random charges $|q_i| \leqslant 0.5$; $0 \leqslant p \leqslant 20$, order increases from top to bottom, even (solid line), odd (dashed line); circles ($\circ$) indicate E_1 value for $\kappa = 0, 5, p = 6$ and $\kappa = 1, p = 0, 20$.

Table 1
Outline of treecode algorithm. θ: MAC parameter, p: order of Taylor approximation and N_0: maximum number of particles in a leaf.

```
1  program main
2  input particle positions and weights: x_i, q_i, i = 1 : N
3  input user-specified parameters: θ, p, N_0
4  construct tree
5  do i = 1, N
6      compute-potential(x_i, root)
7  end do
8  end main
9  subroutine compute-potential(x, c)
10 if MAC is satisfied
11     compute and store moments of c (unless they are already available)
12     compute particle–cluster interaction by Taylor approximation
13 else
14     if c is a leaf
15         compute particle–cluster interaction by direct summation
16     else
17         do j = 1, number of children of c
18             compute-potential(x, c%child(j))
19         end do
20     end if
21 end if
22 return
23 end compute-potential
```

4. Code implementation

First we comment on the data structures used in the treecode. The particle coordinates and charges are stored in a linear array. Each cluster is defined by a data structure containing pointers to the particles contained in the cluster, the intervals in space defining the cluster, the cluster moments and pointers to the children of the cluster.

Next in Table 1 we present an outline of the algorithm. After inputting the particle data and user-specified parameters, the code constructs the tree by dividing the particles into a hierarchy of clusters [2]. Each cluster is a cube with sides parallel to the Cartesian axes and the cubes on a given level are uniform in size. The root cluster is the smallest cube containing all the particles. The root is bisected in each coordinate direction yielding eight child clusters. Subdivision is then applied to the children and this continues until the number of particles in a cluster is less than a user-specified value N_0. The remaining undivided clusters form the leaves of the tree.

Next the code loops over the particles $\mathbf{x}_i$ to compute the potentials V_i in Eq. (5). The clusters c appearing in (5) are determined by traversing the tree for each particle $\mathbf{x}_i$. This is done by calling a recursive subroutine **compute-potential**$(\mathbf{x}_i, c)$ which computes the particle–cluster interaction $V_{i,c}$ in Eq. (6). In the initial call, c is set to the root cluster. If the MAC (13) is satisfied, then $V_{i,c}$ is computed by Taylor approximation (the moments of cluster c are first computed and stored, unless they are available from a previous particle–cluster approximation). If the MAC is not satisfied, then there are two options. If c is a leaf, then $V_{i,c}$ is computed by direct summation. If c is not a leaf, then the code calls **compute-potential**$(\mathbf{x}_i, c\%\text{child}(j))$, where index j runs over the children of cluster c.

In a particle–cluster treecode such as this, the number of operations is $O(N \log N)$, where the factor N comes from the loop over particles and the factor $\log N$ is the depth of the tree [2]. In the present Cartesian implementation the prefactor is $O(p^3)$, representing the number of multi-indices satisfying $0 \leqslant ||\mathbf{k}|| \leqslant p$ (this is the number of moments required for each cluster, the number of Taylor coefficients required for a particle–cluster approximation and the number of operations required to compute the approximation).

The computations were run on an Intel Pentium 4 processor (3.2 GHz, 2 MB level 2 cache, 1536 MB DDR2 memory). The code was written in Fortran90 using double precision arithmetic and was compiled using the ifort compiler. In the CPU results given below, the treecode timings include the entire computation (constructing the tree, computing the moments, evaluating the potential). The direct sum was coded as two nested loops running over all particles. The code is available by contacting one of the authors.

5. Numerical results

In this section we examine the error, CPU time and memory usage of the treecode in comparison with direct summation for the problem of computing the potentials V_i, for $i = 1 : N$ in Eq. (4). Fig. 3 shows the three test cases considered, randomly distributed particles (a) inside a cube, (b) on the surface of a sphere and (c) on an 8-sphere configuration (motivated by [29]). We used a set of representative parameter values for the treecode, $\theta = 0.5$ for the MAC parameter, p = 0:2:10 for the order of Taylor approximation and $N_0 = 500$ for the maximum number of particles in a leaf. The system size was $N = 2^m \cdot 10^3$ for m = 0:10, starting at N = 1k and ending at N = 1024k.

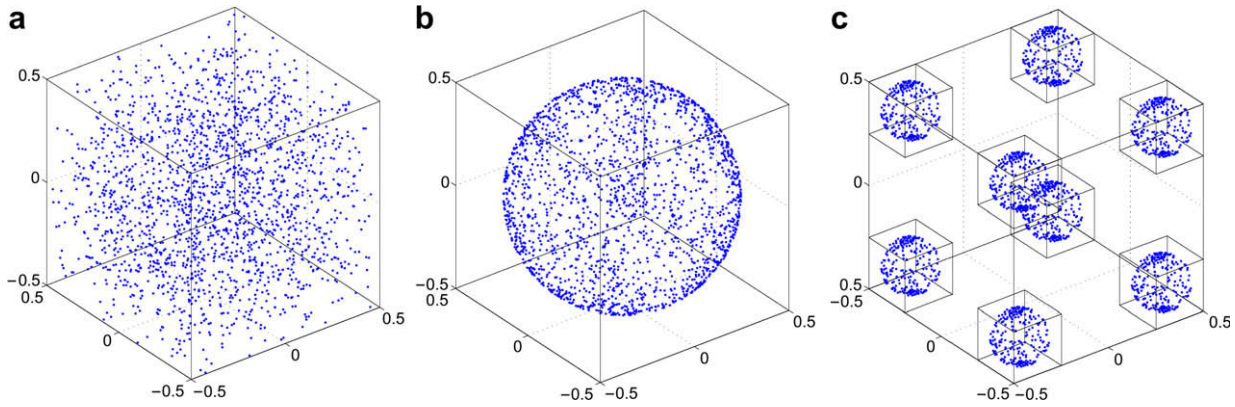

Fig. 3. Test cases, particles distributed randomly (a) inside a cube, (b) on the surface of a sphere, (c) on a configuration of 8 spheres at the corners of a cube (motivated by [29]).

5.1. Particles inside a cube

First we report the treecode performance for randomly distributed particles in the cube $[-0.5, 0.5]^3$ with random charges $|q_i| \leqslant 0.5$. Results are presented for the Coulomb potential, $\kappa = 0$ and the screened Coulomb potential with $\kappa = 1$.

The relative error in potential is defined by

$$E_2 = \left(\frac{\sum_{i=1}^{N} |V_i - \widehat{V}_i|^2}{\sum_{i=1}^{N} |V_i|^2} \right)^{1/2}, \tag{26}$$

where V_i is the exact value given by direct summation (6) and $\widehat{V}_i$ is the value computed by the treecode. Fig. 4 shows the relative error in potential E_2 as a function of the number of particles N and the approximation order p. The error varies little with N and it decreases steadily as p increases. The error is slightly higher for $\kappa = 1$ than for $\kappa = 0$, consistent with the trend in Fig. 2.

Fig. 5 shows the CPU time as a function of the number of particles N and the approximation order p. The CPU time is $O(N^2)$ for direct summation and is consistent with $O(N \log N)$ for the treecode. The breakeven point depends on the order p and we discuss this below, but for sufficiently large N, the treecode is faster than direct summation. The CPU times for direct summation and the treecode are slightly higher for $\kappa = 1$ than for $\kappa = 0$, due to the expense of computing the exponential in the screened potential (1). The treecode CPU time displays a periodic variation with N which is more pronounced for low order p. This is presumably due to details of the code implementation and computer cache size, rather than an intrinsic property of the method.

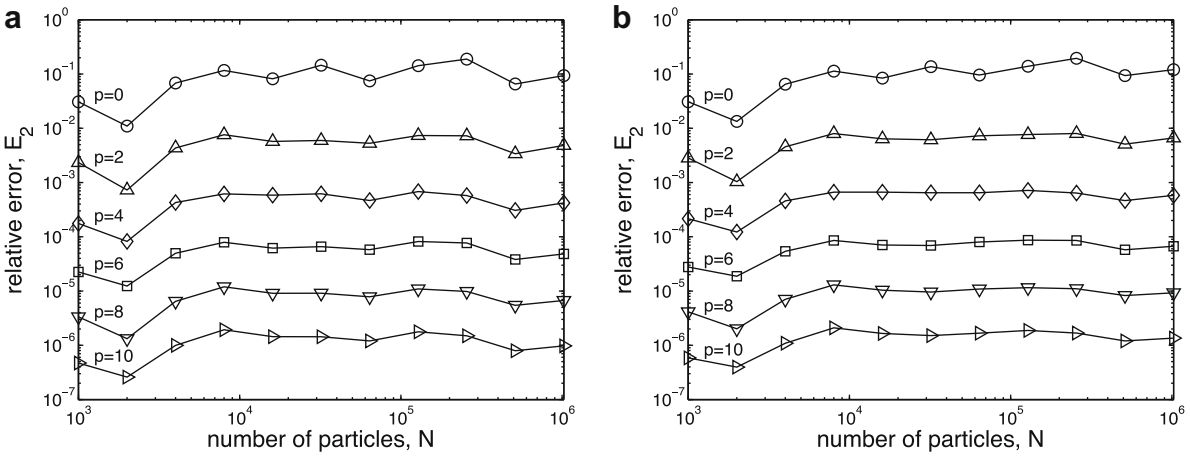

Fig. 4. Random particles inside a cube; relative error in potential E_2, Eq. (26) computed by the treecode with approximation order p; (a) Coulomb potential, $\kappa = 0$ and (b) screened Coulomb potential, $\kappa = 1$.

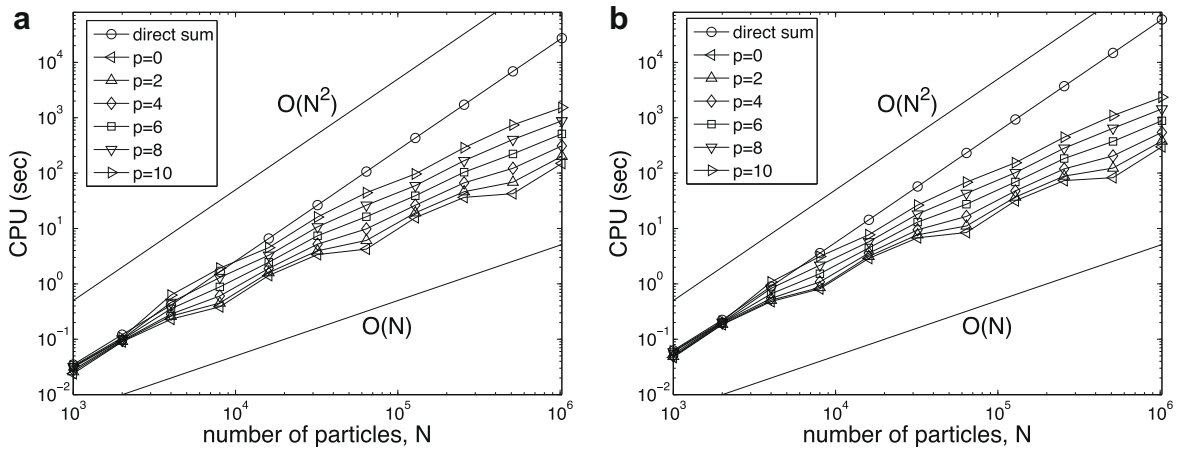

Fig. 5. Random particles inside a cube; CPU time required by direct summation and the treecode with order p; (a) Coulomb potential, $\kappa = 0$ and (b) screened Coulomb potential, $\kappa = 1$.

Although the treecode is intended for problems requiring large values of N, we want to comment on the results in Fig. 5 for small N. These results indicate that direct summation and the treecode have comparable CPU times for $N = 1k$, $2k$ and $p \leqslant 10$. This can be understood by the following considerations. For $N = 1k$, $2k$ and $N_0 = 500$, there are only two levels in the tree. In this case, with our choice of MAC parameter $\theta = 0.5$, most of the particle–cluster interactions are computed by direct summation, but a few of them are computed by Taylor approximation and these compensate for the overhead in the treecode. Since relatively few particle–cluster interactions are computed by Taylor approximation, the treecode CPU time depends only weakly on the order p for $N = 1k$, $2k$, as seen in Fig. 5. For $N = 4k$ there is a change in behaviour. In this case the tree has three levels and more of the particle–cluster interactions are computed by Taylor approximation, causing the treecode CPU time to vary more strongly with order p. As N increases still further, the tree acquires more levels and even more of the particle–cluster interactions are computed by Taylor approximation, enabling the treecode to achieve its asymptotic $O(N \log N)$ behaviour.

Fig. 6(a) shows the memory usage as a function of the number of particles N and the approximation order p. Results are presented only for $\kappa = 1$, since the memory usage is independent of κ. The treecode requires more memory than direct summation, primarily to store the cluster moments. The memory usage is $O(N)$ for both the treecode and direct summation, but it varies periodically with N for the treecode, again presumably due to details of the code implementation and computer cache size. Fig. 6(b) shows the memory usage ratio (treecode/direct summation). The treecode requires about twice as much memory as direct summation for $p = 0$ and less than five times as much for $p = 10$.

5.2. Comparison of three test cases

Next we examine the treecode performance for three test cases shown in Fig. 3, randomly distributed particles (a) inside a cube, (b) on the surface of a sphere and (c) on an 8-sphere configuration. In proceeding from (a) to (b) to (c) the particle den-

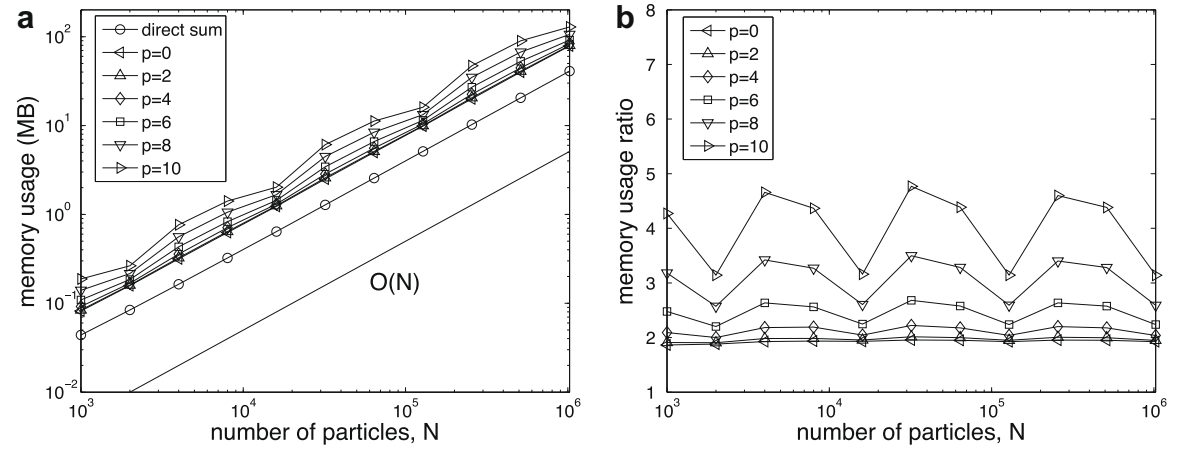

Fig. 6. Random particles inside a cube; memory usage for direct summation and the treecode with order p, for screened Coulomb potential, $\kappa = 1$; (a) memory usage and (b) memory usage ratio (treecode/direct sum).

sity becomes increasingly non-homogeneous. In the cube test case, the particles were randomly distributed as in the previous subsection. In the sphere test cases, particles inside a cube were projected radially onto the surface of a sphere and these were replicated for the 8-sphere configuration. The reason for considering the sphere test cases is that many biological applications involve surface charge distributions. For example in a dielectric medium, a surface charge develops at a discontinuity of the dielectric function and specific instances include solute–solvent interfaces, ion channels and cell membranes. The results presented below are meant to indicate the treecode's potential capability for these applications.

We also compare two types of particle clusters. The first type are uniform cubes on each level, as in the previous subsection and the original treecode [2]. The second type are adapted rectangular boxes obtained by shrinking the clusters so they fit the particles they contain [18]. A schematic in 2D is shown in Fig. 7. In both cases empty clusters are omitted and the tree has the same logical structure, but the adapted rectangular boxes have smaller radii and this affects the results for the sphere and 8-sphere test cases as we will see.

Table 2 reports the treecode performance for representative parameter values $\kappa = 1, p = 6, \theta = 0.5, N_0 = 500$. Results using uniform cubes are labeled "treecode-1" and those using adapted rectangular boxes are labeled "treecode-2". The direct sum CPU time and memory usage are also given; the values depend on N, but otherwise are independent of the particle density and hence are the same for all three test cases.

First consider the error. The error has the same order of magnitude for all three test cases, between 10^{-4} and 10^{-5}. The error is slightly smaller in the sphere and 8-sphere test cases than in the cube test case. This is attributed to the fact that the tree has more empty clusters in the sphere and 8-sphere test cases and these clusters contribute nothing to the error. Next note that within each test case, the error is slightly larger for treecode-2 than for treecode-1. This is attributed to the fact that the adapted rectangular boxes have smaller radii than the uniform cubes, implying that the MAC is satisfied more often and as a result, some particle–cluster interactions that were computed by direct sum in treecode-1 are instead computed by Taylor approximation in treecode-2.

Next consider the CPU time. The treecode CPU time decreases on proceeding from the cube to the sphere to the 8-sphere configuration. To help understand this, note for example that the sphere test case has more empty clusters than the cube test case and consequently each non-empty cluster in the sphere test case contains more particles on average for a given value of N. This leads to greater efficiency since the cost of computing a particle–cluster interaction by Taylor approximation is independent of the number of particles in the cluster (the cluster moments will be available most of the time from a previous interaction). Next note that within each test case, treecode-2 is faster than treecode-1. This is attributed again to the fact that the adapted rectangular boxes have smaller radii than the uniform cubes, so the MAC is satisfied at a higher level in the tree where the clusters contain more particles. The gain in efficiency due to using adapted rectangular boxes increases with N. In the 8-sphere test case with $N = 1024k$, treecode-1 is about 130 times faster than direct summation and treecode-2 is about 170 times faster.

Finally consider the memory usage. The memory usage is roughly the same for all three test cases and for the two versions of the treecode. There is a slight increase in memory usage for the sphere and 8-sphere test cases relative to the cube test case for $N = 1024k$, but even so, the treecode (with $p = 6$ in Table 2) uses roughly only 2.5 times the memory required by direct summation.

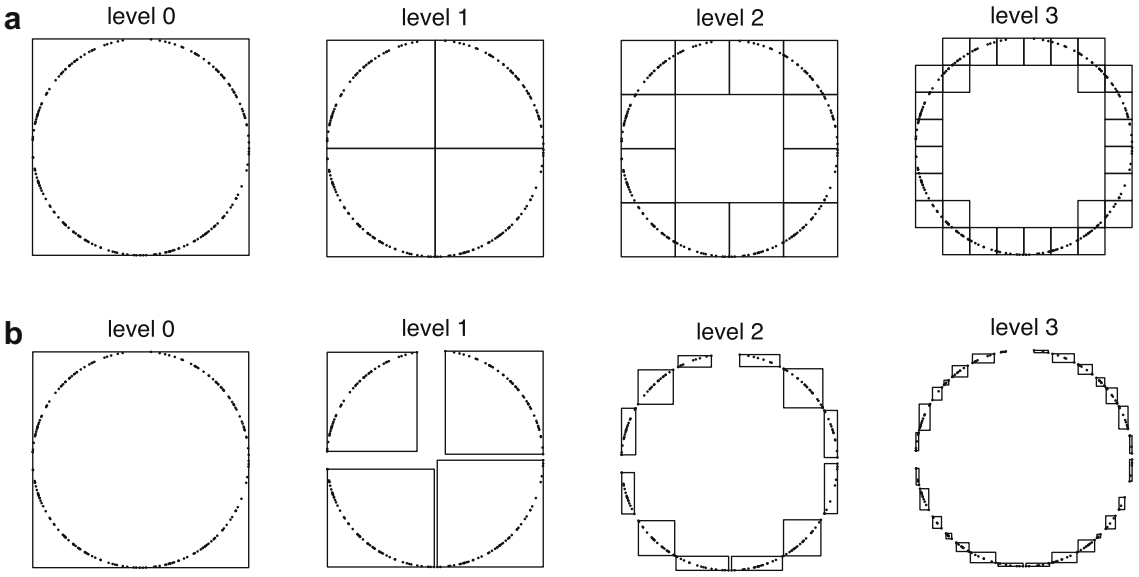

Fig. 7. Schematic of tree structure for random particles on a circle in 2D. The clusters are (a) uniform squares on each level, or (b) adapted rectangular boxes obtained by shrinking the squares.

Table 2
Treecode performance for the cube, sphere and 8-sphere test cases, $\kappa = 1, p = 6, \theta = 0.5, N_0 = 500$, error given by Eq. (26), CPU time in seconds (sec), memory usage in megabytes (MB); the clusters were either uniform cubes (treecode-1) or adapted rectangular boxes (treecode-2).

	Test case	Method	$N = 16k$	$N = 64k$	$N = 256k$	$N = 1024k$
Error	Cube	treecode-1	7.05e−5	8.03e−5	8.56e−5	6.69e−5
		treecode-2	7.41e−5	8.37e−5	8.81e−5	6.77e−5
	Sphere	treecode-1	1.54e−5	2.29e−5	3.12e−5	2.97e−5
		treecode-2	2.41e−5	3.85e−5	5.01e−5	5.21e−5
	8-Sphere	treecode-1	1.57e−5	2.11e−5	2.20e−5	2.32e−5
		treecode-2	1.61e−5	3.02e−5	4.13e−5	4.15e−5
CPU (s)	Cube	treecode-1	4.44	27.39	183.29	883.12
		treecode-2	4.43	27.11	181.30	875.99
	Sphere	treecode-1	4.05	22.51	118.58	590.15
		treecode-2	3.46	17.99	91.48	444.70
	8-Sphere	treecode-1	1.85	12.41	83.49	452.86
		treecode-2	1.80	11.21	67.05	345.94
	Cube, sphere, 8-sphere	direct sum	14.35	230.95	3705.07	59439.65
Memory (MB)	Cube	treecode-1	1.44	6.60	26.98	91.65
		treecode-2	1.44	6.60	26.98	91.65
	Sphere	treecode-1	1.41	6.34	25.59	100.13
		treecode-2	1.41	5.78	23.98	96.22
	8-Sphere	treecode-1	1.43	6.43	25.96	98.46
		treecode-2	1.43	6.41	25.73	100.87
	Cube, sphere, 8-sphere	direct sum	0.64	2.56	10.24	40.96

6. Summary

We presented a treecode algorithm for evaluating electrostatic potentials in a charged particle system undergoing screened Coulomb interactions in 3D. The method follows the Barnes–Hut approach in which the particle–particle interactions are replaced by particle–cluster interactions [2]. We employed a far-field Cartesian Taylor expansion of the screened Coulomb potential to compute particle–cluster approximations. The Taylor coefficients were evaluated using new recurrence relations which permit efficient computation of high order approximations. Two types of clusters were considered, uniform cubes and adapted rectangular boxes. The treecode error, CPU time and memory usage were reported and compared with direct summation for randomly distributed particles inside a cube, on the surface of a sphere and on an 8-sphere configuration. For a given order of Taylor approximation, the treecode CPU time scales as $O(N \log N)$ and the memory usage scales as $O(N)$. The results show that the treecode is well suited for non-homogeneous particle distributions as in the sphere and 8-sphere test cases. We plan to employ the treecode in boundary element simulations of the Poisson–Boltzmann equation, which involve screened charge distributions on a molecular surface [1,5,19,20].

Researchers needing to use a fast summation method for screened Coulomb interactions have several options to consider including the analytic FMM [4,13] and the kernel-independent FMM [29]. The present work proposes the Cartesian treecode as a viable alternative. The key advantage of the treecode is its relative simplicity in comparison to the FMM. Treecodes in general use only a far-field expansion, while the FMM uses a more elaborate procedure which converts the far-field expansion to a near-field expansion [14,12]. The Cartesian treecode proposed here uses elementary Taylor series, while the analytic FMM uses special function expansions. In principle, for a given order of expansion, the treecode requires $O(N \log N)$ execution time, while the FMM requires $O(N)$ execution time, but these estimates are based on the number of floating point operations and do not account for memory access costs. These estimates also assume that the particle distribution is homogeneous, which is not the case in many applications. In conclusion then, our point of view is that the Cartesian treecode enriches the range of choices available for computing electrostatic potentials and forces.

Acknowledgments

The research was supported in part by NSF Grants DMS-0510162, ATM-0723440, EAR-0724656, Michigan Life Sciences Corridor Grant #1515, and a University of Michigan Research Fellowship. The authors thank the reviewers for suggestions which improved the manuscript.

References

[1] N.A. Baker, Improving implicit solvent simulations: a Poisson-centric view, Curr. Opin. Struct. Biol. 15 (2005) 137–143.
[2] J. Barnes, P. Hut, A hierarchical $O(N \log N)$ force-calculation algorithm, Nature 324 (1986) 446–449.
[3] M. Bonitz, D. Block, O. Arp, V. Golubnychiy, H. Baumgartner, P. Ludwig, A. Piel, A. Filinov, Structural properties of screened Coulomb balls, Phys. Rev. Lett. 96 (2006) 075001.
[4] A.H. Boschitsch, M.O. Fenley, W.K. Olson, A fast adaptive multipole algorithm for calculating screened Coulomb (Yukawa) interactions, J. Comput. Phys. 151 (1999) 212–241.
[5] A.H. Boschitsch, M.O. Fenley, H.-X. Zhou, Fast boundary element method for the linear Poisson–Boltzmann equation, J. Phys. Chem. B 106 (2002) 2741–2754.
[6] G.E. Brown, A.D. Jackson, The Nucleon–Nucleon Interaction, North-Holland, Amsterdam, 1976.

[7] H. Cheng, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, J. Comput. Phys. 155 (1999) 468–498.
[8] H.-Q. Ding, N. Karasawa, W.A. Goddard III, Atomic level simulations on a million particles: the cell multipole method for Coulomb and London nonbond interactions, J. Chem. Phys. 97 (1992) 4309–4315.
[9] C.I. Draghicescu, M. Draghicescu, A fast algorithm for vortex blob interactions, J. Comput. Phys. 11 (1995) 69–78.
[10] Z.-H. Duan, R. Krasny, An Ewald summation based multipole method, J. Chem. Phys. 113 (2000) 3492–3495.
[11] Z.-H. Duan, R. Krasny, An adaptive treecode for computing nonbonded potential energy in classical molecular systems, J. Comput. Chem. 22 (2001) 184–195.
[12] L. Greengard, The Rapid Evaluation of Potential Fields in Particle Systems, MIT Press, Cambridge, MA, 1988.
[13] L. Greengard, J. Huang, A new version of the fast multipole method for screened Coulomb interactions in three dimensions, J. Comput. Phys. 180 (2002) 642–658.
[14] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (1987) 325–348.
[15] R.W. Hockney, J.W. Eastwood, Computer Simulation Using Particles, Taylor and Francis, New York, 1988.
[16] B. Honig, A. Nicholls, Classical electrostatics in biology and chemistry, Science 268 (1995) 1144–1149.
[17] U. Konopka, G.E. Morfill, L. Ratke, Measurement of the interaction potential of microspheres in the sheath of a RF discharge, Phys. Rev. Lett. 84 (2000) 891–894.
[18] K. Lindsay, R. Krasny, A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow, J. Comput. Phys. 172 (2001) 879–907.
[19] B. Lu, X. Cheng, J. Huang, J.A. McCammon, Order N algorithm for computation of electrostatic interactions in biomolecular systems, Proc. Nat. Acad. Sci. 103 (2006) 19314–19319.
[20] B. Lu, X. Cheng, J.A. McCammon, New-version-fast-multipole-method accelerated electrostatic calculations in biomolecular systems, J. Comput. Phys. 226 (2007) 1348–1366.
[21] B.Z. Lu, Y.C. Zhou, M.J. Holst, J.A. McCammon, Recent progress in numerical methods for the Poisson–Boltzmann equation in biophysical applications, Commun. Comput. Phys. 3 (2008) 973–1009.
[22] T. Sakajo, H. Okamoto, An application of Draghicescu's fast summation method to vortex sheet motion, J. Phys. Soc. Jpn 67 (1998) 462–470.
[23] J.K. Salmon, M.S. Warren, Skeletons from the treecode closet, J. Comput. Phys. 111 (1994) 136–155.
[24] T. Schlick, B. Li, W.K. Olson, The influence of salt on the structure and energetics of supercoiled DNA, Biophys. J. 67 (1994) 2146–2166.
[25] T. Schlick, Molecular Modeling and Simulation: An Interdisciplinary Guide, Springer, New York, 2002.
[26] B. Shanker, H. Huang, Accelerated Cartesian expansions – A fast method for computing of potentials of the form R^{-v} for all real v, J. Comput. Phys. 226 (2007) 732–753.
[27] T.E. Sheridan, W.L. Theisen, Study of two-dimensional Debye clusters using Brownian motion, Phys. Plasmas 13 (2006) 062110.
[28] J. Shimada, H. Kaneko, T. Takada, Performance of fast multipole methods for calculating electrostatic interactions in biomacromolecular simulations, J. Comput. Chem. 15 (1994) 28–43.
[29] 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.
[30] F. Zhao, An O(N) Algorithm for three-dimensional N-body simulations, Technical Report AI-TR-995, M.I.T. AI Lab, 1987.