SIAM J. SCI. COMPUT.
Vol. 33, No. 5, pp. 2341–2355

FAST EVALUATION OF MULTIQUADRIC RBF SUMS BY A CARTESIAN TREECODE*

ROBERT KRASNY† AND LEI WANG‡

Abstract. A treecode is presented for evaluating sums defined in terms of the multiquadric radial basis function (RBF), $\phi(\mathbf{x}) = (|\mathbf{x}|^2 + c^2)^{1/2}$, where $\mathbf{x} \in \mathbb{R}^3$ and $c \geq 0$. Given a set of N nodes, evaluating an RBF sum directly requires CPU time that scales like $O(N^2)$. For a given level of accuracy, the treecode reduces the CPU time to $O(N \log N)$ using a far-field expansion of $\phi(\mathbf{x})$. We consider two options for the far-field expansion: (1) a Laurent series previously used in applications of the Fast Multipole Method to multiquadric RBFs, and (2) a certain Taylor series previously used in treecode particle simulations, but not yet in the context of multiquadric RBFs. It is known that the Laurent series converges when the RBF parameter c lies in an interval $0 \leq c \leq \bar{c}$, where $\bar{c}$ is proportional to the minimum node spacing, but here we show that the Taylor series converges uniformly for $c \geq 0$. We implement the treecode in Cartesian coordinates and use a recurrence relation to compute the Taylor coefficients. Numerical results exhibit the treecode error, CPU time, and memory usage in two test cases, random nodes in a cube and on the surface of a sphere. The treecode approach presented here is applicable to generalized multiquadrics in any dimension.

Key words. fast summation, multiquadric radial basis function, treecode, Taylor series

AMS subject classifications. 65D99, 41A58, 41A63

DOI. 10.1137/090779851

1. Introduction.

Radial basis function (RBF) methods were developed for scattered data interpolation, and now they are also applied in the numerical solution of differential equations [7, 14]. Here we consider the multiquadric RBF,

$$\phi(\mathbf{x}) = (|\mathbf{x}|^2 + c^2)^{1/2}, \tag{1.1}$$

where $\mathbf{x} \in \mathbb{R}^d$ and $c \geq 0$ is the RBF parameter. An RBF approximation has the form

$$s(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j \phi(\mathbf{x} - \mathbf{x}_j), \tag{1.2}$$

where λ_j and $\mathbf{x}_j$ are the coefficients and nodes, respectively, and s is either the interpolant to a set of data values or the numerical solution of a differential equation. It is known that large values of c yield spectral accuracy [8, 9], but the problem of determining the coefficients becomes ill-conditioned in that regime and this is exacerbated when the nodes are closely spaced [35]. Several approaches have been developed to address the ill-conditioning, e.g., [3, 12, 15, 17, 18], but this issue is beyond the scope of the present work. Here we assume the coefficients and nodes are given and focus on the cost of evaluating the sums

$$s(\mathbf{x}_i) = \sum_{j=1}^{N} \lambda_j \phi(\mathbf{x}_i - \mathbf{x}_j), \quad i = 1 : N, \tag{1.3}$$

*Submitted to the journal's Methods and Algorithms for Scientific Computing section December 11, 2009; accepted for publication (in revised form) June 27, 2011; published electronically September 22, 2011. This work was supported by NSF grant ATM-0723440 and UCAR grant S07-59369.
http://www.siam.org/journals/sisc/33-5/77985.html

†Department of Mathematics, University of Michigan, Ann Arbor, MI 48109-1043 (krasny@umich.edu).

‡Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4844 (lwang@mcs.anl.gov).

which can also be viewed as a matrix-vector multiplication. Note that the evaluation points and nodes are assumed to be the same, but this is not an essential restriction in our approach, and we sometimes also refer to these collectively as particles. The straightforward direct summation method loops over i and j to evaluate the sums in (1.3), and this requires $O(N^2)$ operations. Since this is prohibitively expensive when N is large, various approaches have been developed to reduce the cost, as discussed below.

Before proceeding, we note that an alternative form of the multiquadric is sometimes used,

$$\phi(\mathbf{x}) = ((\epsilon|\mathbf{x}|)^2 + 1)^{1/2}, \tag{1.4}$$

where $\epsilon > 0$. The two forms (1.1) and (1.4) are equivalent, up to a scaling factor, by setting $\epsilon = c^{-1}$. There is also interest in the generalized multiquadric,

$$\phi(\mathbf{x}) = (|\mathbf{x}|^2 + c^2)^{\nu/2}, \tag{1.5}$$

where $\nu \in \mathbb{R}$. In this work we focus on (1.1), but our approach can also be applied to (1.4) and (1.5).

1.1. Fast summation methods. The literature on fast RBF methods is growing rapidly and here we focus on work related to multiquadrics. Evaluating the RBF sums in (1.3) resembles the problem of evaluating potentials and fields induced by long-range particle interactions in computational physics [23]. Examples include the gravitational and electrostatic potentials induced by point masses and point charges, respectively. If the RBF function values $\phi(\mathbf{x})$ decay sufficiently fast for large $|\mathbf{x}|$, then the CPU time can be reduced by applying a cutoff; i.e., the terms $\phi(\mathbf{x}_i - \mathbf{x}_j)$ can be omitted when particles $\mathbf{x}_i$ and $\mathbf{x}_j$ are sufficiently far apart. For example, Frank and Reich [19] developed a particle method for the shallow water equations using a generalized multiquadric with $\nu = -8$ to represent the vorticity, and they employed a cutoff in computations with $N \geq 500{,}000$. If the error induced by the cutoff is acceptable, then this is a viable approach to reducing the CPU time, but if the error is not acceptable, then it becomes necessary to account for the omitted terms beyond the cutoff radius.

One approach to handling long-range particle interactions leads to a class of particle-mesh methods [25], but here we are concerned with an alternative class of tree-based multipole methods. Early contributions in this context include the Barnes–Hut treecode [1] and the Greengard–Rokhlin fast multipole method (FMM) [21]. These algorithms divide the particles into a hierarchy of clusters having a tree structure. The treecode replaces some of the particle-particle interactions by particle-cluster interactions which are evaluated by a far-field multipole approximation. The FMM combines the far-field approximations into a near-field approximation before performing the evaluation. For a given level of accuracy, the treecode requires $O(N \log N)$ operations and in principle the FMM requires $O(N)$ operations. However, these estimates assume that the particles are distributed homogeneously in space, and the actual CPU run time may vary due to boundary effects and details of the particle distribution. In addition to the operation count, one may also compare these methods in terms of algorithmic complexity, communication costs, memory usage, and parallelizability. Optimizing these tree-based fast summation methods is still an active area of research.

The treecode and FMM were originally developed for the Newtonian potential, given by (1.5) with $\nu = -1, c = 0$. In that case ϕ is a harmonic function and classical

multipole series can be used for the far-field and near-field expansions. However, the multiquadric (1.1) is nonharmonic for $c \neq 0$ and the classical expansions do not apply. Several approaches have been developed to overcome this obstacle. In particular, Beatson and Newsam [5] and Beatson and Greengard [4] developed an FMM for the multiquadric (1.1) using a far-field Laurent series and a near-field Taylor series, and Cherrie, Beatson, and Newsam [10] implemented that approach for the generalized multiquadric (1.5). We will discuss these expansions in more detail below, but here we note that the Laurent series converges for a restricted set of RBF parameter values, $0 \leq c \leq \bar{c}$, where $\bar{c}$ is proportional to the minimum node spacing [4, 30].

A number of other fast summation methods have been developed for multiquadric RBFs. Roussos and Baxter [31] expressed the multiquadric as a Gaussian integral and applied suitable quadrature rules and the fast Gauss transform [22]. Ying [38] applied the kernel-independent FMM, which employs equivalent particle densities in place of analytic series expansions [37]. Beatson and Newsam [6] and Livne and Wright [28] developed methods based on polynomial interpolation and multilevel summation. Gumerov and Duraiswami [24] developed an FMM for the multiquadric in dimension $d = 2$ by relating it to the biharmonic kernel in dimension $d = 3$.

1.2. The present work. Here we describe a treecode algorithm for evaluating the multiquadric sums (1.3) using a far-field expansion arising from a certain Taylor series. We shall show that the Taylor series converges for all RBF parameter values $c \geq 0$, in contrast with the Laurent series. The treecode is implemented in Cartesian coordinates, and we use a recurrence relation to compute the Taylor coefficients. The method is applied to two test cases: random particles in a cube and on the surface of a sphere. For a given level of accuracy, the treecode CPU time scales like $O(N \log N)$, and it achieves a substantial speedup over direct summation for large systems. This type of Cartesian far-field Taylor series treecode was previously used in computational vortex methods [11, 27, 29, 32, 33] and molecular dynamics [13, 26], but not yet in the context of multiquadric RBFs. Alternative tree-based methods using Cartesian coordinates include an FMM for the Newtonian potential [39] and an FMM using Cartesian tensors for general power law potentials [36].

The paper is organized as follows. In section 2 we explain the role of the far-field expansion in a treecode. In section 3 we consider the Laurent and Taylor expansions of the multiquadric RBF in dimension $d = 1$. In section 4 we consider the generalization to dimension $d = 3$ and derive a recurrence relation for the Taylor coefficients. In section 5 we describe the treecode in detail. In section 6 we present numerical results. Conclusions are given in section 7.

2. The role of the far-field expansion in a treecode. Assume the particles have been divided into disjoint clusters and write the RBF sum (1.3) as a sum of particle-cluster interactions,

$$s(\mathbf{x}_i) = \sum_{j=1}^{N} \lambda_j \phi(\mathbf{x}_i - \mathbf{x}_j) = \sum_{C} \sum_{\mathbf{y}_j \in C} \lambda_j \phi(\mathbf{x}_i - \mathbf{y}_j) = \sum_{C} s(\mathbf{x}_i, C), \tag{2.1}$$

where C denotes a cluster (not to be confused with the RBF parameter c) and

$$s(\mathbf{x}_i, C) = \sum_{\mathbf{y}_j \in C} \lambda_j \phi(\mathbf{x}_i - \mathbf{y}_j) \tag{2.2}$$

is the interaction between particle $\mathbf{x}_i$ and cluster C. The scheme for choosing the clusters will be explained later. There are several options for evaluating the particle-

cluster interaction (2.2). If particle $\mathbf{x}_i$ and cluster C are not well-separated (the precise criterion is given later), then direct summation is used, but if they are well-separated, then we apply a far-field expansion of the form

$$\phi(\mathbf{x}_i - \mathbf{y}_j) \approx \sum_{k=0}^{p} f_k(\mathbf{x}_i) g_k(\mathbf{y}_j), \tag{2.3}$$

which is assumed to be valid for $|\mathbf{x}_i| > |\mathbf{y}_j|$. Substituting in (2.2) and rearranging the result yields

$$s(\mathbf{x}_i, C) \approx \sum_{k=0}^{p} f_k(\mathbf{x}_i) \sum_{\mathbf{y}_j \in C} \lambda_j g_k(\mathbf{y}_j). \tag{2.4}$$

The terms in (2.4) involving $g_k(\mathbf{y}_j)$ are generalized moments of cluster C and since they are independent of particle $\mathbf{x}_i$, they can be used for different $\mathbf{x}_i$, which leads to a savings in CPU time. Treecodes also use a third option involving recursive application of (2.4) to subclusters of C. The choice of functions f_k, g_k determines the accuracy and efficiency of this approach, and in general there may be several alternatives with different properties. The Laurent series in $\mathbf{x}$ is a natural choice for the multiquadric RBF [4, 5, 10], but as we shall see, the Taylor series in $\mathbf{y}$ yields a suitable far-field expansion in $\mathbf{x}$ with better convergence properties.

3. Far-field expansions for the multiquadric in one dimension. In this section we consider two far-field expansions for the multiquadric (1.1) in dimension $d = 1$. The scalar variables in this section are printed in regular font as opposed to bold font.

3.1. Laurent series for large x. The multiquadric function in one dimension defined by

$$\phi(x - y) = \sqrt{(x-y)^2 + c^2} \tag{3.1}$$

is viewed as having branch points in the complex x-plane at $x = y \pm ic$ and a branch cut as shown in Figure 1(a). Beatson and Greengard [4] proposed the Laurent series

$$\phi(x - y) = |x| - \frac{xy}{|x|} + \frac{c^2}{2|x|} + \frac{c^2 xy}{2|x|^3} + \frac{c^2(4y^2 - c^2)}{8|x|^3} + \cdots \tag{3.2}$$

for the far-field expansion in an FMM for the multiquadric in one dimension. The convergence criterion for the Laurent series (3.2) is

$$\rho_1 \equiv \frac{\sqrt{y^2 + c^2}}{|x|} < 1 \tag{3.3}$$

(the shaded region in Figure 1(a)), as noted in [4]. Then assuming $|x| > |y|$, the convergence criterion (3.3) restricts the values of c which can be used. For example, in [4] it was assumed that $0 \le c \le h$, where h is the radius of the leaves of the tree. Next we consider an alternative far-field expansion.

3.2. Taylor series for small y. Consider the Taylor series of (3.1) with respect to y about $y = 0$. In this case $\phi(x-y)$ is viewed as having branch points in the complex y-plane at $y = x \pm ic$ and branch cuts as shown in Figure 1(b). The expansion is

$$\phi(x - y) = x_c - \frac{xy}{x_c} + \frac{c^2 y^2}{2x_c^3} + \frac{c^2 xy^3}{2x_c^5} + \frac{c^2(4x^2 - c^2)y^4}{8x_c^7} + \cdots, \tag{3.4}$$

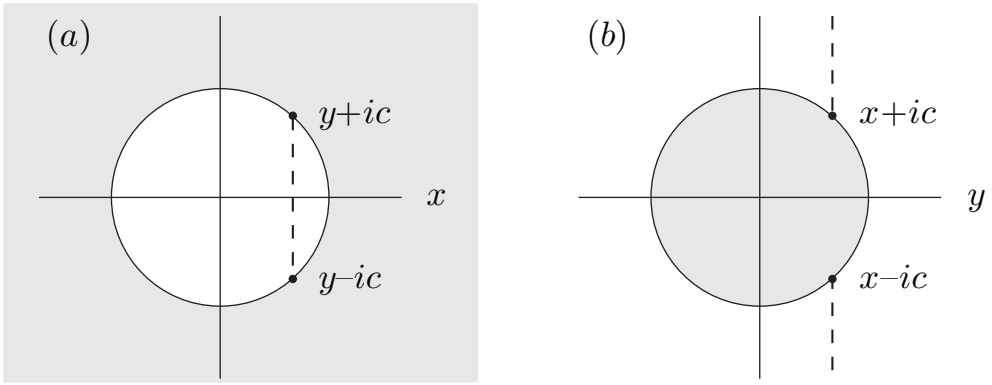

FIG. 1. *Schematic showing branch points, branch cuts, and domain of convergence (shaded) for far-field expansions of* $\phi(x-y)$ *given in* (3.1); (a) *the Laurent series* (3.2) *converges outside a disk in the x-plane;* (b) *the Taylor series* (3.4) *converges inside a disk in the y-plane. As c increases, the shaded region in* (a) *becomes smaller and the shaded region in* (b) *becomes larger.*

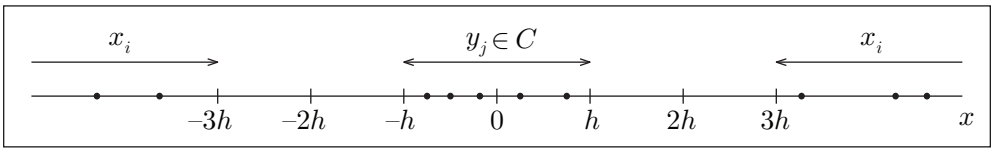

FIG. 2. *Example in one dimension from* [4] *showing a cluster C of nodes satisfying* $|y_j| \le h$ *and well-separated evaluation points satisfying* $|x_i| \ge 3h$.

where $x_c = \sqrt{x^2+c^2}$. The convergence criterion for the Taylor series (3.4) is

$$\rho_2 \equiv \frac{|y|}{\sqrt{x^2+c^2}} < 1 \tag{3.5}$$

(the shaded region in Figure 1(b)). Hence for $|x| > |y|$, the series (3.4) is uniformly convergent for $c \ge 0$, and increasing the value of c accelerates the rate of convergence. The Taylor series (3.4) has the required form (2.3), and hence it can be used as a far-field expansion in the treecode. Note that ρ_1, ρ_2 defined in (3.3), (3.5) control the rate of convergence of the two series; explicit error bounds are given in [4, 10].

3.3. Example. We consider an example from [4] in which the clusters are intervals on a line, as in Figure 2. The nodes y_j are contained in a cluster C of radius h, $|y_j| \le h$, and the evaluation points are separated from the cluster by at least one cluster diameter, $|x_i| \ge 3h$. Setting $c = h$ as in [4], the convergence factors are

$$\rho_1 = \max_{i,j} \frac{\sqrt{y_j^2+c^2}}{|x_i|} \le \frac{\sqrt{h^2+h^2}}{3h} = \frac{\sqrt{2}}{3} = 0.47, \tag{3.6}$$

$$\rho_2 = \max_{i,j} \frac{|y_j|}{\sqrt{x_i^2+c^2}} \le \frac{h}{\sqrt{9h^2+h^2}} = \frac{1}{\sqrt{10}} = 0.32. \tag{3.7}$$

Hence the Taylor series (3.4) converges faster than the Laurent series (3.2) for this example. In the next section we extend these considerations to the generalized multiquadric and higher space dimensions.

4. The generalized multiquadric in multiple dimensions. We start by quoting some results from [10] for the generalized multiquadric (1.5) on $\mathbb{R}^d$ for the case in which ν is an odd integer. The results are expressed using multivariate polynomials given by

$$P_l^{(\nu)}(b_1,b_2,b_3) = \sum_{j=\lfloor \frac{l+1}{2} \rfloor}^{l} \binom{\nu/2}{j}\binom{j}{l-j} b_2^{2j-l}(b_1 b_3)^{l-j} \tag{4.1}$$

for $l \geq 0$ and $P_l^{(\nu)}(b_1,b_2,b_3) = 0$ for $l < 0$.

4.1. Expansions. The first result is a Laurent series in $\mathbf{x}$,

$$\phi(\mathbf{x}-\mathbf{y}) = (|\mathbf{x}-\mathbf{y}|^2 + c^2)^{\nu/2} = \sum_{l=0}^{\infty} \frac{P_l^{(\nu)}(|\mathbf{y}|^2+c^2, -2\langle \mathbf{x},\mathbf{y}\rangle, |\mathbf{x}|^2)}{|\mathbf{x}|^{2l-\nu}}, \tag{4.2}$$

which converges for $|\mathbf{x}| > \sqrt{|\mathbf{y}|^2+c^2}$ (for an error bound see [10, p. 1557, Lemma 3.1]). The second result is a Taylor series in $\mathbf{x}$ in the form

$$\phi(\mathbf{x}-\mathbf{y}) = (|\mathbf{x}-\mathbf{y}|^2 + c^2)^{\nu/2} = \sum_{l=0}^{\infty} \frac{P_l^{(\nu)}(|\mathbf{x}|^2, -2\langle \mathbf{x},\mathbf{y}\rangle, |\mathbf{y}|^2+c^2)}{(|\mathbf{y}|^2+c^2)^{(2l-\nu)/2}}, \tag{4.3}$$

which converges for $|\mathbf{x}| < \sqrt{|\mathbf{y}|^2+c^2}$ (for an error bound see [10, p. 1569, Lemma 8.1]). Cherrie, Beatson, and Newsam [10] developed an FMM using the Laurent series (4.2) for the far-field expansion and the Taylor series (4.3) for the near-field expansion.

Here we interchange $\mathbf{x}$ and $\mathbf{y}$ in the Taylor series (4.3) and obtain

$$\phi(\mathbf{x}-\mathbf{y}) = (|\mathbf{x}-\mathbf{y}|^2 + c^2)^{\nu/2} = \sum_{l=0}^{\infty} \frac{P_l^{(\nu)}(|\mathbf{y}|^2, -2\langle \mathbf{x},\mathbf{y}\rangle, |\mathbf{x}|^2+c^2)}{(|\mathbf{x}|^2+c^2)^{(2l-\nu)/2}}, \tag{4.4}$$

which converges for $|\mathbf{y}| < \sqrt{|\mathbf{x}|^2+c^2}$. This is a Taylor series in $\mathbf{y}$, but as a function of $\mathbf{x}$ it converges in the exterior of a ball, and hence it can be viewed as a far-field expansion for the generalized multiquadric on $\mathbb{R}^d$. For $\nu = 1, d = 1$, the Laurent series (4.2) reduces to (3.2) and the Taylor series (4.4) reduces to (3.4).

In our application it is convenient to rewrite (4.4) as a standard Taylor expansion with respect to $\mathbf{y}$ about $\mathbf{y} = 0$ using Cartesian multi-index notation,

$$\phi(\mathbf{x}-\mathbf{y}) = (|\mathbf{x}-\mathbf{y}|^2 + c^2)^{\nu/2} = \sum_{||\mathbf{k}||=0}^{\infty} \frac{1}{\mathbf{k}!} D^{\mathbf{k}}\phi(\mathbf{x}) \cdot (-\mathbf{y})^{\mathbf{k}}, \tag{4.5}$$

where $\mathbf{k} = (k_1,\ldots,k_d)$, $k_i \geq 0$ are integers, $||\mathbf{k}|| = k_1 + \cdots + k_d$, $\mathbf{k}! = k_1!\cdots k_d!$, $\mathbf{y}^{\mathbf{k}} = y_1^{k_1}\cdots y_d^{k_d}$, and $D^{\mathbf{k}} = D_1^{k_1}\cdots D_d^{k_d}$. Equation (4.5) results from collecting like powers of $\mathbf{y}$ in (4.4), and it has the same convergence criterion, $|\mathbf{y}| < \sqrt{|\mathbf{x}|^2+c^2}$. This means that the Taylor series (4.5) has the same favorable convergence behavior as in the one-dimensional case. If p terms are retained in the approximation and we define convergence factors in the multidimensional case by

$$\rho_1 = \frac{\sqrt{|\mathbf{y}|^2+c^2}}{|\mathbf{x}|}, \quad \rho_2 = \frac{|\mathbf{y}|}{\sqrt{|\mathbf{x}|^2+c^2}}, \tag{4.6}$$

then the error bounds in [10] cited above imply that the error in the truncated Laurent series (4.2) is $O(\rho_1^{p+1})$ and the error in the truncated Taylor series (4.5) is $O(\rho_2^{p+1})$.

Hence assuming that $|\mathbf{x}| > |\mathbf{y}|$, as is relevant to the far-field expansion in a treecode or FMM, the Laurent series (4.2) converges for $0 \le c \le \bar{c}$ and diverges for $c > \bar{c}$, where $\bar{c} = \sqrt{|\mathbf{x}|^2 - |\mathbf{y}|^2}$, while the Taylor series (4.5) converges for all $c \ge 0$.

Finally we shift the center of the Taylor series so it can be applied to evaluate a general particle-cluster interaction. Let $\mathbf{y}_C$ denote the center of cluster C, and then replacing $\mathbf{x}$ by $\mathbf{x} - \mathbf{y}_C$ and $\mathbf{y}$ by $\mathbf{y} - \mathbf{y}_C$ in (4.5), we have

$$\phi(\mathbf{x}-\mathbf{y}) = (|\mathbf{x}-\mathbf{y}|^2 + c^2)^{\nu/2} = \sum_{||\mathbf{k}||=0}^{\infty} \frac{1}{\mathbf{k}!} D^{\mathbf{k}}\phi(\mathbf{x}-\mathbf{y}_C)\cdot(-(\mathbf{y}-\mathbf{y}_C))^{\mathbf{k}}, \tag{4.7}$$

which converges for $|\mathbf{y}-\mathbf{y}_C| < \sqrt{|\mathbf{x}-\mathbf{y}_C|^2 + c^2}$. The Taylor series (4.7) has the form required for a far-field expansion (2.3). Hence we can express the particle-cluster interaction (2.2) as

$$s(\mathbf{x}_i, C) = \sum_{\mathbf{y}_j \in C} \lambda_j \phi(\mathbf{x}_i - \mathbf{y}_j) \tag{4.8a}$$

$$= \sum_{\mathbf{y}_j \in C} \lambda_j \sum_{||\mathbf{k}||=0}^{\infty} \frac{1}{\mathbf{k}!} D^{\mathbf{k}}\phi(\mathbf{x}_i - \mathbf{y}_C)\cdot(-(\mathbf{y}_j - \mathbf{y}_C))^{\mathbf{k}} \tag{4.8b}$$

$$\approx \sum_{||\mathbf{k}||=0}^{p} \frac{1}{\mathbf{k}!} D^{\mathbf{k}}\phi(\mathbf{x}_i - \mathbf{y}_C) \sum_{\mathbf{y}_j \in C} \lambda_j (\mathbf{y}_C - \mathbf{y}_j)^{\mathbf{k}} \tag{4.8c}$$

$$= \sum_{||\mathbf{k}||=0}^{p} a_{\mathbf{k}}(\mathbf{x}_i - \mathbf{y}_C)\, m_{\mathbf{k}}(C), \tag{4.8d}$$

where p is a user-specified truncation parameter,

$$a_{\mathbf{k}}(\mathbf{x}_i - \mathbf{y}_C) = \frac{1}{\mathbf{k}!} D^{\mathbf{k}}\phi(\mathbf{x}_i - \mathbf{y}_C) \tag{4.9}$$

is the $\mathbf{k}$th Taylor coefficient of the multiquadric $\phi(\mathbf{x})$, evaluated at $\mathbf{x} = \mathbf{x}_i - \mathbf{y}_C$, and

$$m_{\mathbf{k}}(C) = \sum_{\mathbf{y}_j \in C} \lambda_j (\mathbf{y}_C - \mathbf{y}_j)^{\mathbf{k}} \tag{4.10}$$

is the $\mathbf{k}$th moment of the cluster taking into account the factor $(-1)^{\mathbf{k}}$. The expression (4.8d) defines a pth order far-field Taylor approximation for a particle-cluster interaction.

4.2. Recurrence relation. The coefficients in the Laurent series (4.2) can be computed using the recurrence relation derived in section 5 of [10]. Here we derive a recurrence relation for the Taylor coefficients (4.9) of the generalized multiquadric (1.5) in the case $d = 3$ (the result is easily extended to any dimension). Some special cases were previously considered (see, e.g., [27] for $\nu = -1$, [13] for $c = 0$). The first step is to show by explicit computation that

$$(|\mathbf{x}|^2 + c^2) D_1\phi(\mathbf{x}) - \nu x_1 \phi(\mathbf{x}) = 0. \tag{4.11}$$

In the remainder of the derivation we omit the argument of ϕ. Then applying $D_1^{k_1-1}$ and using Leibniz's rule for differentiating a product yields

$$\begin{aligned}(|\mathbf{x}|^2 + c^2) D_1^{k_1}\phi + (2(k_1-1) - \nu) x_1 D_1^{k_1-1}\phi \\ + (k_1-1)(k_1 - 2 - \nu) D_1^{k_1-2}\phi = 0.\end{aligned} \tag{4.12}$$

Then applying $D_2^{k_2} D_3^{k_3}$, dividing by $\mathbf{k}!$, and using the definition of $a_{\mathbf{k}}$ yields

$$(|\mathbf{x}|^2 + c^2) a_{\mathbf{k}} + 2 \sum_{i=1}^{3} x_i a_{\mathbf{k}-\mathbf{e}_i} + \sum_{i=1}^{3} a_{\mathbf{k}-2\mathbf{e}_i} - \frac{(2+\nu)}{k_1} (x_1 a_{\mathbf{k}-\mathbf{e}_1} + a_{\mathbf{k}-2\mathbf{e}_1}) = 0, \tag{4.13}$$

where $\mathbf{e}_i$ are the Cartesian basis vectors. Similar equations are obtained by replacing index 1 by 2 and 3. Multiplying these equations by k_1, k_2, k_3, respectively, and summing the results yields the symmetric expression,

$$a_{\mathbf{k}} + \frac{2(||\mathbf{k}|| - 1) - \nu}{||\mathbf{k}||(|\mathbf{x}|^2 + c^2)} \sum_{i=1}^{3} x_i a_{\mathbf{k}-\mathbf{e}_i} + \frac{||\mathbf{k}|| - 2 - \nu}{||\mathbf{k}||(|\mathbf{x}|^2 + c^2)} \sum_{i=1}^{3} a_{\mathbf{k}-2\mathbf{e}_i} = 0. \tag{4.14}$$

Setting $\mathbf{x} = \mathbf{x}_i - \mathbf{y}_C$ yields a recurrence relation for the Taylor coefficients (4.9). For $||\mathbf{k}|| = 0, 1$ the coefficients $a_{\mathbf{k}}$ are computed explicitly and then (4.14) yields the coefficients for $||\mathbf{k}| \geq 2$.

4.3. Comparison of errors. Next we compare the errors in the Laurent series and the Taylor series for a particle-cluster interaction in the case $\nu = 1, d = 3$. The particle is located at the origin in $\mathbb{R}^3$, $\mathbf{x} = 0$, and the cluster C consists of $N = 10^3$ nodes distributed randomly in the cube $[0.75, 1]^3$. The RBF coefficients are set to $\lambda_i = 1$ as in [10]. The error is defined by

$$E_1 = \frac{|s(\mathbf{x}, C) - \tilde{s}(\mathbf{x}, C)|}{|s(\mathbf{x}, C)|}, \tag{4.15}$$

where $s(\mathbf{x}, C)$ is the exact particle-cluster interaction (2.2) and $\tilde{s}(\mathbf{x}, C)$ is the approximation obtained by truncating the expansion for $\phi(\mathbf{x}_i - \mathbf{y}_j)$ using the Laurent series (4.2) or the Taylor series (4.7). For each approximation, the necessary series coefficients were computed using the recurrence relations discussed above.

Figure 3 shows the error as a function of the RBF parameter c and order p for $10^{-3} \leq c \leq 10^3$ and $p = 0:2:10$. As noted above, the Laurent series converges on an interval $0 \leq c \leq \bar{c}$ and diverges for $c > \bar{c}$, while the Taylor series converges for all $c \geq 0$. For some special values of p and c, the error drops to zero; this occurs near $c = 10^{-1}$ for the Laurent series and near $c = 1$ for the Taylor series. For small values of c, the two series have comparable errors. For a given order p, the rate of convergence of the Taylor series improves as $c \to \infty$. These results confirm the favorable convergence properties of the far-field Taylor series (4.7). Identical results, up to roundoff error, were obtained for the alternative multiquadric form (1.4) with $\epsilon = c^{-1}$.

5. Description of treecode. A treecode for evaluating the multiquadric approximation (1.2) in dimension $d = 3$ was implemented using the Taylor series (4.7) as the far-field expansion. We tested two versions, the original Barnes–Hut approach [1] which loops through the particles and determines the interaction list for each particle on the fly, and a modified approach suggested by Barnes [2] which determines the interaction list for each leaf before carrying out the computation. In our tests we found only minor differences in performance; the results below used the modified approach [2]. Table 1 presents pseudocode for key parts of the algorithm, including the main program and two subroutines, one of which is recursive.

The code starts by inputting data and constructing a hierarchical tree of particle clusters. The root cluster is a cube containing all the particles. The root is bisected along the Cartesian axes and the eight children become subclusters of the root. The

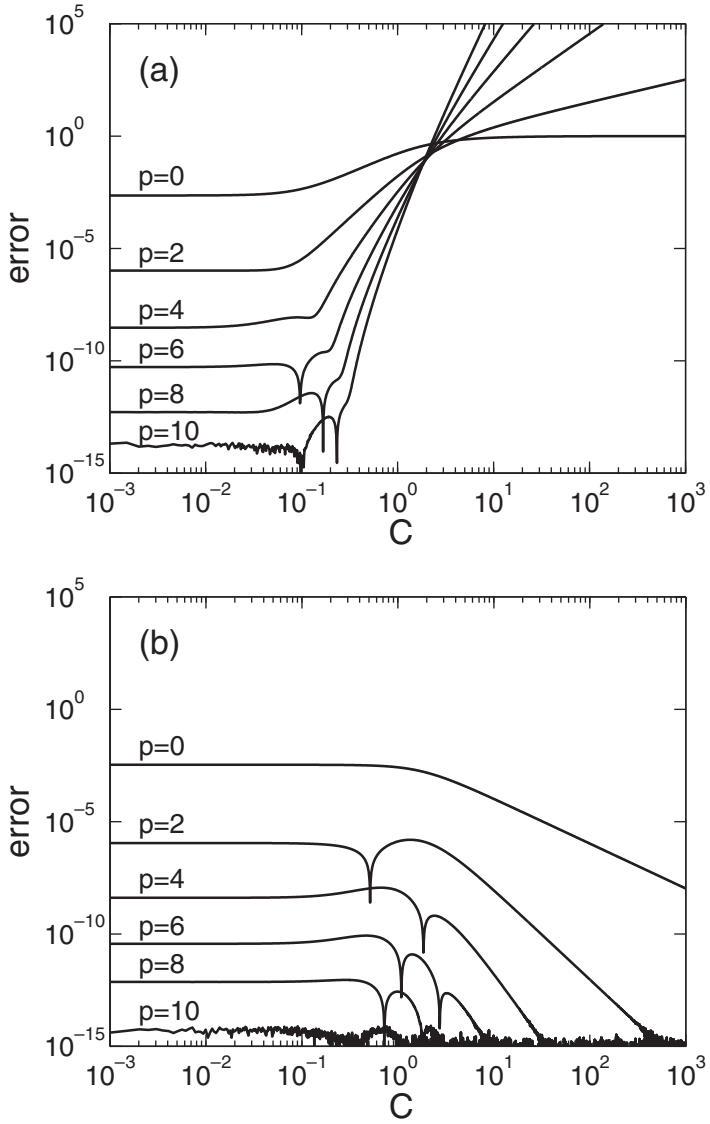

FIG. 3. *Error* (4.15) *in a particle-cluster approximation with one evaluation point at the origin in* $\mathbb{R}^3$ *and a cluster of* $N = 10^3$ *random nodes in the cube* $[0.75, 1]^3$*;* $\nu = 1, d = 3$*. The error is plotted as a function of the RBF parameter* c *and order* p *for* $10^{-3} \le c \le 10^3$ *and* $p = 0:2:10$*.* (a) *Laurent series* (4.2), (b) *Taylor series* (4.7).

child clusters are then bisected and the process continues until a cluster contains fewer than N_0 particles, where N_0 is a user-specified parameter. Each cluster has a data structure containing necessary information, e.g., pointers to the particles belonging to the cluster, pointers to the children of the cluster, coordinates of the cluster center, and so on. After the tree is constructed, the code cycles through the leaves, determining the interaction lists for each leaf and then computing the RBF approximation for each particle in the leaf.

Each leaf has two interaction lists, a far-list pointing to well-separated clusters which are processed by Taylor approximation, and a near-list pointing to non-well-

TABLE 1
Pseudocode for key parts of the treecode.

```
1   program main
2   input RBF parameters c, λ_i, x_i, i = 1 : N
3   input treecode parameters p, θ, N_0
4   construct tree
5   for each leaf
6      compute_interactions_list(leaf, root)
7      compute_RBF_approximation(leaf)
8   end program
9   subroutine compute_interaction_lists(leaf, C)
10  x=center(leaf), y=center(C), R = |x − y|, r=radius(C)
11  if r/√(R^2 + c^2) ≤ θ, add C to far-list of leaf
12  if r/√(R^2 + c^2) > θ
13     if C is not a leaf, compute_interaction_list(leaf, children of C)
14     if C is a leaf, add C to near-list of leaf
15  end subroutine
16  subroutine compute_RBF_approximation(leaf)
17  for each particle x in leaf
18     for each cluster C on far-list of leaf
19        compute interaction between x and C by Taylor approximation
20     for each cluster C on near-list of leaf
21        compute interaction between x and C by direct summation
22  end subroutine
```

separated clusters which are processed by direct summation. The code uses a multipole acceptance criterion (MAC) to determine whether a given leaf and cluster are well-separated [1, 2, 34]. The criterion for being well-separated is $r/\sqrt{R^2+c^2} \le \theta$, where r is the radius of the cluster, R is the distance between the leaf and the cluster (measured from the centers), and θ is a user-specified parameter which together with the order p controls the series truncation error.

The treecode was implemented in the C programming language and the computations were performed on an Intel Pentium M processor (1.5 GHz, 768MB RAM, 1024 KB level 2 cache). Memory usage was obtained using the performance counter tool in the Windows operating system.

6. Treecode performance. We present results using the treecode for two test cases in three dimensions. In the first case we consider random nodes in a unit cube, and in the second case the nodes were projected radially from the cube to the surface of a unit sphere. One motivation for considering the sphere is the growing interest in RBF methods for geophysical fluid flow [16]. In both test cases, the RBF coefficients were set to $\lambda_i = 1$, as in [10], and the RBF parameter was set to $c = 10^{-1}$. The relative error is defined by

$$E_2 = \left(\sum_{i=1}^{N} |s(\mathbf{x}_i) - \tilde{s}(\mathbf{x}_i)|^2 \Big/ \sum_{i=1}^{N} |s(\mathbf{x}_i)|^2 \right)^{1/2}, \tag{6.1}$$

where $s(\mathbf{x}_i)$ is the exact value (1.3) obtained by direct summation and $\tilde{s}(\mathbf{x}_i)$ is the treecode approximation.

6.1. Random nodes in a cube. First we present results using $N = 216$K nodes for order $p = 0:2:10$ and MAC parameter $\theta = 0.2, 0.5, 0.8$, with maximum leaf size $N_0 = 200$. Table 2 presents the data. As expected, for a given value of θ, increasing the Taylor expansion order p yields a smaller error and larger CPU time. Similarly,

TABLE 2

Random nodes in a cube, the treecode error (6.1)*, and CPU time (*sec*) are given for system size* $N = 216\text{K}$*, RBF parameter* $c = 10^{-1}$*, order* $p = 0 : 2 : 10$*, MAC parameter* $\theta = 0.2, 0.5, 0.8$*, and maximum leaf size* $N_0 = 200$*.*

	Treecode error (6.1)			CPU time (sec)		
p	$\theta = 0.8$	$\theta = 0.5$	$\theta = 0.2$	$\theta = 0.8$	$\theta = 0.5$	$\theta = 0.2$
0	3.379e-2	1.376e-2	2.213e-3	0.9	3.2	259.0
2	5.835e-5	1.292e-5	3.081e-7	7.8	30.2	470.1
4	2.014e-5	1.336e-6	2.711e-9	15.7	55.8	798.1
6	2.149e-6	8.761e-8	3.065e-11	33.5	119.3	1444.0
8	5.805e-7	5.461e-9	4.890e-13	63.0	308.9	2522.5
10	1.850e-7	6.107e-10	2.220e-14	115.7	684.4	4304.4

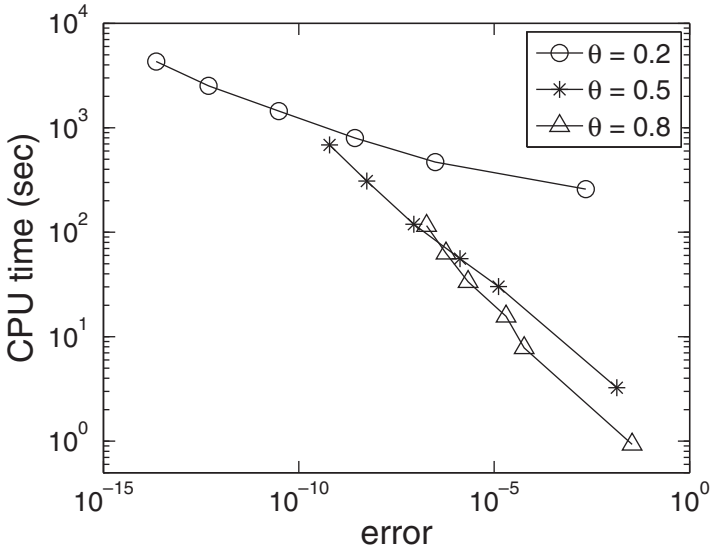

FIG. 4. *Random nodes in a cube, scatter plot of CPU time and error* (6.1)*, data from Table* 2*, system size* $N = 216\text{K}$*, order* $p = 0:2:10$ *(increasing from right to left), MAC parameter* $\theta = 0.2$ $(\circ)$*,* $\theta = 0.5$ $(*)$*,* $\theta = 0.8$ $(\triangle)$*, RBF parameter* $c = 10^{-1}$*, and maximum leaf size* $N_0 = 200$*. The lower envelope of the data gives the most efficient choice of parameters* (p, θ) *for a given error.*

for a given value of p, decreasing the MAC parameter θ also yields a smaller error and larger CPU time, since more particle-cluster interactions are performed by direct summation. Figure 4 displays the data as a scatter plot. The data points for each θ are connected by a line, and p increases from right to left on each line. The lower envelope of the data gives the most efficient choice of parameters (p, θ) for a given error. For example, errors in the interval $[10^{-6}, 10^{-1}]$ are computed with least CPU time using $\theta = 0.8$. We set $\theta = 0.8$ in the remainder of this work.

Next we vary the problem size. Figure 5(a) plots the error (6.1) as a function of order p for problems of size $N = 10^3, 20^3, \ldots, 100^3$. The maximum leaf size was $N_0 = 200$, except for the two largest systems ($N = 90^3, 100^3$) for which $N_0 = 400$. Figure 5(a) shows that the error is almost independent of system size N, and for a given value of N, the error decreases with increasing order p. Figure 5(b) plots the corresponding CPU time, showing that the treecode CPU time is consistent with $O(N \log N)$ scaling and that the treecode is faster than direct summation except for small system size N and large order p.

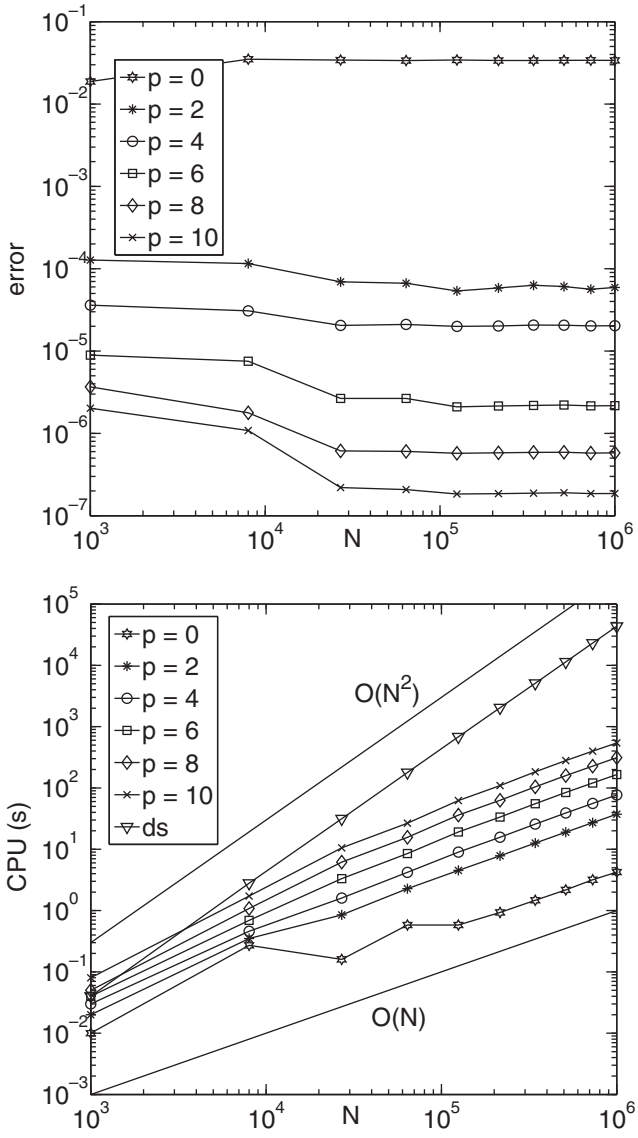

FIG. 5. *Random nodes in a cube,* $N = 10^3, 20^3, \ldots, 100^3$*, order* $p = 0:2:10$*, MAC parameter* $\theta = 0.8$*, RBF parameter* $c = 10^{-1}$*, and maximum leaf size* $N_0 = 200$ *except* $N_0 = 400$ *for* $N = 90^3, 100^3$*.* (a) *plots treecode error* (6.1)*, and* (b) *plots treecode CPU time (sec), where* ds *denotes direct sum.*

Figure 6 plots the memory used by the treecode as a function of system size N for order $p = 0:2:10$. The memory used by direct summation is also shown. The treecode uses arrays of size $O(p^3)$ to store cluster moments and Taylor coefficients, and so it requires more memory as the order p increases. For low order p, the treecode memory usage is a small multiple of the memory required by direct summation. For high order p, the treecode memory usage becomes irregular, and this is why the maximum leaf size was changed from $N_0 = 200$ to $N_0 = 400$ for the two largest systems. It is likely that the treecode memory usage can be reduced by further tuning, but even so, for

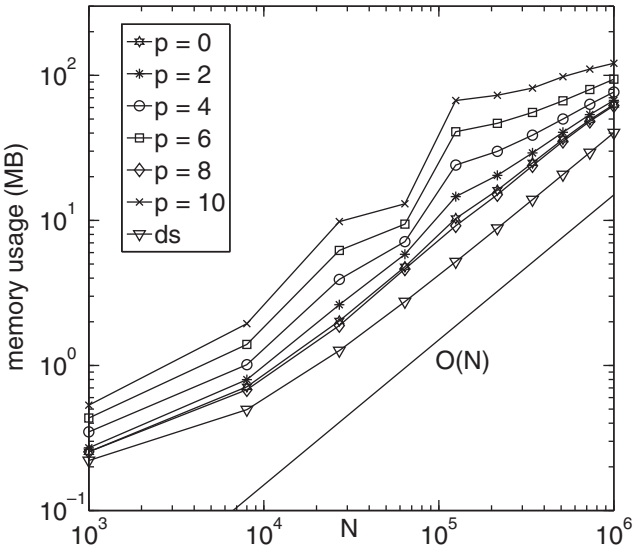

FIG. 6. *Random nodes in a cube, treecode memory usage, using the same parameters as in Figure* 5 *caption, with* ds *denoting direct sum.*

TABLE 3

Random nodes in a cube and on a sphere. tc *denotes treecode and* ds *denotes direct sum, CPU time in seconds and memory in MB, RBF parameter* $c = 10^{-1}$*, maximum leaf size* $N_0 = 200$ *except* $N_0 = 400$ *for* $N = 1000K$*, order* $p = 6$*, and MAC parameter* $\theta = 0.8$*.*

$N \rightarrow$	8K	64K	216K	512K	1000K
Cube					
error (tc)	7.5e-6	2.7e-6	2.1e-6	2.2e-6	2.2e-6
CPU (tc)	0.7	8.5	33.5	84.4	165.9
CPU (ds)	1.5	95.7	978.8	5479.0	21888.2
memory (tc)	1.0	7.1	29.9	50.0	76.7
memory (ds)	0.5	2.8	8.8	20.7	40.2
Sphere					
error (tc)	4.0e-6	2.6e-6	2.6e-6	2.4e-6	2.1e-6
CPU (tc)	0.7	8.7	34.1	88.5	180.7
CPU (ds)	1.5	95.7	978.8	5479.0	21888.2
memory (tc)	1.0	9.1	29.6	72.0	101.2
memory (ds)	0.5	2.8	8.8	20.7	40.2

the largest system considered here ($N = 10^6$) and the highest order ($p = 10$), the treecode used approximately only three times as much memory as direct summation.

6.2. Random nodes on a sphere. Table 3 compares two test cases, random nodes in a cube and on a sphere, for problems of varying size, using a representative set of parameter values, $\theta = 0.8, p = 6$. We record results for the treecode and direct summation. Note that the direct sum CPU time and memory usage depend on the system size N, but not on the node distribution, so these values are the same for the two test cases. Next note that the treecode error is essentially the same for the two node distributions, while the treecode CPU time and memory usage are slightly higher for the sphere than for the cube, with approximately a 10% penalty in CPU time and a 33% penalty in memory usage for the largest system $N = 1000$K. It is likely that greater efficiency can be gained in the sphere test case, for example, by shrinking

the clusters [26] or spherical panel clustering [20]. The favorable performance of the treecode in the sphere test case arises from the adaptive nature of the tree structure, which has been noted before [1].

7. Conclusion. We presented a treecode for fast evaluation of multiquadric RBF sums using a far-field Taylor series to compute well-separated particle-cluster interactions. The treecode was implemented in Cartesian coordinates and a recurrence relation was used to compute the necessary Taylor coefficients. Numerical results were presented for random nodes in a cube and on the surface of a sphere. For a given level of accuracy, the treecode CPU time scales like $O(N \log N)$, where N is the number of nodes, and a substantial speedup over direct summation is achieved for large systems. The code is available by contacting one of the authors. This approach can be applied to any RBF for which a suitable recurrence relation exists, including the generalized multiquadric in any dimension.

A novel aspect of this approach is that the Taylor series converges for all values of the RBF parameter $c \geq 0$. This is a consequence of the properties of analytic functions and the location of the complex branch points of the multiquadric RBF. Previous applications of the FMM to multiquadric RBFs have relied on the Laurent series [4, 10], but in that case the RBF parameter is restricted to values $0 \leq c \leq \bar{c}$, where $\bar{c}$ is proportional to the minimum node spacing. There is great interest in the regime $c \gg 1$, where the multiquadric RBF has spectral accuracy. A future goal therefore is to combine the treecode with a method to overcome the ill-conditioning in that limit [3, 12, 15, 17, 18]. If this can be accomplished, the resulting RBF method may lead to advances in various applications, including geophysical fluid flow [16].

Acknowledgments. We thank John Boyd, Natasha Flyer, and Bengt Fornberg for introducing us to RBF methods.

REFERENCES

[1] J. E. Barnes and P. Hut, *A hierarchical $O(N \log N)$ force-calculation algorithm*, Nature, 324 (1986), pp. 446–449.
[2] J. E. Barnes, *A modified tree code: Don't laugh; it runs*, J. Comput. Phys., 87 (1990), pp. 161–170.
[3] R. K. Beatson, J. B. Cherrie, and C. T. Mouat, *Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration*, Adv. Comput. Math., 11 (1999), pp. 253–270.
[4] R. K. Beatson and L. Greengard, *A short course on fast multipole methods*, in Wavelets, Multilevel Methods and Elliptic PDEs, M. Ainsworth, J. Levesley, W. Light, and M. Marletta, eds., Oxford University Press, Oxford, 1997, pp. 1–37.
[5] R. K. Beatson and G. N. Newsam, *Fast evaluation of radial basis functions:* I, Comput. Math. Appl., 24 (1992), pp. 7–19.
[6] R. K. Beatson and G. N. Newsam, *Fast evaluation of radial basis functions: Moment-based methods*, SIAM J. Sci. Comput., 19 (1998), pp. 1428–1449.
[7] M. D. Buhmann, *Radial Basis Functions*, Cambridge University Press, Cambridge, UK, 2003.
[8] M. D. Buhmann and N. Dyn, *Spectral convergence of multiquadric interpolation*, Proc. Edinb. Math. Soc. (2), 36 (1993), pp. 319–333.
[9] A. H.-D. Cheng, M. A. Golberg, E. J. Kansa, and G. Zammito, *Exponential convergence and h-c multiquadric collocation method for partial differential equations*, Numer. Methods Partial Differential Equations, 19 (2003), pp. 571–594.
[10] J. B. Cherrie, R. K. Beatson, and G. N. Newsam, *Fast evaluation of radial basis functions: Methods for generalized multiquadrics in $\mathbb{R}^n$*, SIAM J. Sci. Comput., 23 (2002), pp. 1549–1571.
[11] C. Draghicescu and M. Draghicescu, *A fast algorithm for vortex blob interactions*, J. Comput. Phys., 116 (1995), pp. 69–78.

[12] T. A. Driscoll and A. R. H. Heryudono, *Adaptive residual subsampling methods for radial basis function interpolation and collocation problems*, Comput. Math. Appl., 53 (2007), pp. 927–939.
[13] Z.-H. Duan and R. Krasny, *An adaptive treecode for computing nonbonded potential energy in classical molecular systems*, J. Comput. Chem., 22 (2001), pp. 184–195.
[14] G. E. Fasshauer, *Meshfree Approximation Methods with* Matlab, World Scientific Publishing, Singapore, 2007.
[15] A. C. Faul, G. Goodsell, and M. J. D. Powell, *A Krylov subspace algorithm for multiquadric interpolation in many dimensions*, IMA J. Numer. Anal., 25 (2005), pp. 1–24.
[16] N. Flyer and G. B. Wright, *Transport schemes on a sphere using radial basis functions*, J. Comput. Phys., 226 (2007), pp. 1059–1084.
[17] B. Fornberg and C. Piret, *A stable algorithm for flat radial basis functions on a sphere*, SIAM J. Sci. Comput., 30 (2007), pp. 60–80.
[18] B. Fornberg and G. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter*, Comput. Math. Appl., 48 (2004), pp. 853–867.
[19] J. Frank and S. Reich, *A particle-mesh method for the shallow water equations near geostrophic balance*, J. Comput. Phys., 180 (2002), pp. 407–426.
[20] W. Freeden, O. Glockner, and M. Schreiner, *Spherical panel clustering and its numerical aspects*, J. Geodesy., 72 (1998), pp. 586–599.
[21] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.
[22] L. Greengard and J. Strain, *The fast Gauss transform*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 79–94.
[23] L. Greengard, *Fast algorithms for classical physics*, Science, 265 (1994), pp. 909–914.
[24] N. A. Gumerov and R. Duraiswami, *Fast radial basis function interpolation via preconditioned Krylov iteration*, SIAM J. Sci. Comput., 29 (2007), pp. 1876–1899.
[25] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, Taylor & Francis, New York, 1988.
[26] P. Li, H. Johnston, and R. Krasny, *A Cartesian treecode for screened Coulomb interactions*, J. Comput. Phys., 228 (2009), pp. 3858–3868.
[27] K. Lindsay and R. Krasny, *A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow*, J. Comput. Phys., 172 (2001), pp. 879–907.
[28] O. E. Livne and G. B. Wright, *Fast multilevel evaluation of smooth radial basis function expansions*, Electron. Trans. Numer. Anal., 23 (2006), pp. 263–287.
[29] Y. M. Marzouk and A. F. Ghoniem, *K-means clustering for optimal partitioning and dynamic load balancing of parallel hierarchical N-body simulations*, J. Comput. Phys., 207 (2005), pp. 493–528.
[30] C. Piret and G. Wright, *Comparison of Three Fast Algorithms for Evaluating RBF Interpolation Functions: FMM, Modified FGT and a Multilevel Technique*, unpublished report, 2005.
[31] G. Roussos and B. J. C. Baxter, *Rapid evaluation of radial basis functions*, J. Comput. Appl. Math., 180 (2005), pp. 51–70.
[32] T. Sakajo, *An extension of Draghicescu's fast tree-code algorithm to the vortex method on a sphere*, J. Comput. Appl. Math., 225 (2009), pp. 158–171.
[33] T. Sakajo and H. Okamoto, *An application of Draghicescu's fast summation method to vortex sheet motion*, J. Phys. Soc. Japan, 67 (1998), pp. 462–470.
[34] J. K. Salmon and M. S. Warren, *Skeletons from the treecode closet*, J. Comput. Phys., 111 (1994), pp. 136–155.
[35] R. Schaback, *Error estimates and condition numbers for radial basis function interpolation*, Adv. Comput. Math., 3 (1995), pp. 251–264.
[36] B. Shanker and H. Huang, *Accelerated Cartesian expansions: A fast method for computing of potentials of the form $R^{-\nu}$ for all real ν*, J. Comput. Phys., 226 (2007), pp. 732–753.
[37] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, J. Comput. Phys., 196 (2004), pp. 591–626.
[38] L. Ying, *A kernel independent fast multipole algorithm for radial basis functions*, J. Comput. Phys., 213 (2006), pp. 451–457.
[39] F. Zhao, *An O(N) Algorithm for Three-Dimensional N-Body Simulations*, Technical report AI-TR-995, M.I.T. Artificial Intelligence Laboratory, 1987.