SIAM J. SCI. COMPUT.
Vol. 34, No. 2, pp. A1126–A1140

A FAST TREECODE FOR MULTIQUADRIC INTERPOLATION WITH VARYING SHAPE PARAMETERS[*]

QUAN DENG[†] AND TOBIN A. DRISCOLL[†]

Abstract. A treecode algorithm is presented for the fast evaluation of multiquadric radial basis function (RBF) approximations. The method is a dual approach to one presented by Krasny and Wang, which applies far-field expansions to clusters of RBF centers (source points). The new approach clusters evaluation points instead and is therefore easily able to cope with basis functions that have different multiquadric shape parameters. The new treecode is able to evaluate an approximation on N centers at M points in $O((N+M)\log M)$ time in the ideal case when evaluation points are uniformly distributed. When coupled with a two-level restricted additive Schwarz preconditioner for GMRES iterations, the treecode is well suited for use within an adaptive RBF iteration, previously described by Driscoll and Heryudono, as is demonstrated by experiments on test functions.

Key words. radial basis function interpolation, treecode, restricted additive Schwarz method

AMS subject classifications. Primary, 65D05; Secondary, 65F10, 41A58

DOI. 10.1137/110836225

1. Introduction.

Radial basis function (RBF) approximation methods provide a mesh-free, dimension-independent way to represent functions on domains discretized as arbitrary point sets. Their simplicity, flexibility, and approximation power have led to interest in their use in many scientific computing applications, including the solution of differential equations [3, 11, 12, 20]. A common form of an RBF approximation is

$$s(\mathbf{x}) = \sum_{j=1}^{N} w_j \phi\big(\epsilon_j \|\mathbf{x} - \mathbf{y}_j\|\big), \tag{1.1}$$

where the w_j are coefficients, $\phi(r)$ is a *basic function* such as e^{-r^2} or $\sqrt{1+r^2}$, the ϵ_j's are known as *shape parameters*, and the $\mathbf{y}_j$'s are points in $\mathbf{R}^d$ typically called *centers*. The norm in (1.1) is usually taken to be the Euclidean 2-norm.

The coefficients w_j are selected to satisfy approximation or collocation constraints, requiring the solution of N algebraic equations. In interpolation, for instance, we are given data values $f_j = f(\mathbf{x}_j)$ at N nodes, leading to

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}, \tag{1.2}$$

where the *interpolation matrix* A satisfies $a_{ij} = \phi(\mathbf{x}_i - \mathbf{y}_j)$. Technically, additional terms may be required in (1.1) in order to guarantee a solution of this system [27], but including them rarely makes any practical difference. Indeed, the main practical concern with solving (1.2) is that the matrix tends to be very ill-conditioned in regimes

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section June 2, 2011; accepted for publication (in revised form) January 30, 2012; published electronically April 10, 2012. This work was supported by the UNIDEL fellowship and by NSF grant DMS-1022704.
http://www.siam.org/journals/sisc/34-2/83622.html

[†]Department of Mathematics, University of Delaware, Newark, DE 19716 (dengquan@udel.edu, driscoll@udel.edu).

where the interpolant has good approximation properties, whether or not supplemental terms are included; hence, we omit them for brevity. The basic function, shape parameters, and center locations have major effects on the nature of the linear system and the approximation properties of the resulting interpolant [10, 16, 31].

Throughout this work we focus on the *multiquadric* basic function $\phi(r) = \sqrt{1 + r^2}$. Multiquadrics are popular because they give high-order or spectral convergence [16, 26] and are observed to work very efficiently, in terms of the number of centers required, for both interpolation and differential equations [12, 17]. The main disadvantage of multiquadrics, compared to other choices of ϕ that have compact support or rapid decay, is that they lead to full matrices to solve for the coefficients w_j in (1.2). For equivalent resolving power, the multiquadric matrices can often be far smaller than in the sparse cases, and the most efficient route to achieve a desired interpolation accuracy is far from clear, even when direct linear algebra is applied. When N is required to be large, however, the $O(N^3)$ requirements of a direct solution of (1.2) are unacceptable, and one must turn to iterative methods. For these to be successful, one requires two key elements: a method to evaluate the RBF sum (1.1) rapidly at the interpolation nodes, and a good preconditioner.

Fast evaluation strategies for RBFs have a substantial history [2, 4, 5, 15, 22, 23, 14], but to the best of our knowledge, all methods presented to date have been demonstrated only for the case of constant shape parameter, $\epsilon_j \equiv \epsilon$. However, allowing shape parameters to vary, often over orders of magnitude, is important to achieve optimal interpolation accuracy at a particular set of centers. In this work we modify the approach of Krasny and Wang [22], who applied a treecode summation based on particle-cluster interactions to multiquadric RBF sums. We show that a dual approach of cluster-particle interactions, grouping together evaluation points rather than basis functions (centers), makes the use of varying shape parameters simple.

We also show how to apply the treecode summation method within an adaptive strategy for selecting RBF centers and shape parameters described by Driscoll and Heryudono [10, 18]. The adaptive iteration pairs well with the fast summation since it constructs centers and nodes within a geometric quadtree/octree structure that can be used for the treecode as well. We also describe a two-level restricted additive Schwarz preconditioner based on domain decomposition, following Cai and Sarkis [8] and Beatson, Light, and Billings [6]. The effectiveness of the fast summation and preconditioned GMRES for adaptive interpolation are all demonstrated in numerical experiments.

2. A new treecode for RBF summation.

In this section we describe an asymptotically fast method for computing the RBF sum (1.1) at a set of nodes $\mathbf{x}_i$, $i = 1, \dots, M$. In the application to Krylov iterations such as GMRES for the linear system (1.2), we have $M = N$, and the node set is equal to the center set, but our description does not make this assumption.

2.1. Use of far-field expansions.

Krasny and Wang [22] proposed a treecode method for this task. The treecode builds a quadtree/octree structure $\mathcal{T}$ on the center set. Each node of $\mathcal{T}$ is associated with a box or cell in $\mathbf{R}^d$. The root of $\mathcal{T}$ contains all the center points $\mathbf{y}_j$. The 2^d children of a node are defined by bisection of the node's cell in each dimension, and each leaf of $\mathcal{T}$ contains no more than N_0 centers, where $N_0 = 200$ is a common choice. Assuming a uniform distribution of the N centers, the height of the tree is $O(\log N)$ on average.

Krasny and Wang's method evaluates (1.1) sequentially at each evaluation node $\mathbf{x}_i$ by applying far-field Taylor expansions to cells containing centers that are "well-separated" (in a sense to be made clear below) from $\mathbf{x}_i$ in cells of the tree structure. If we denote one such cell as C and its centroid as $\mathbf{y}_c$, the interaction between $\mathbf{x}_i$ and the centers in cell C can be expanded as

$$\sum_{\mathbf{y}_j \in C} w_j \phi(\epsilon_j \|\mathbf{x}_i - \mathbf{y}_j\|) = \sum_{\mathbf{y}_j \in C} w_j \sum_{\|\mathbf{k}\|=0}^{\infty} \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \phi(\epsilon_j \|\mathbf{x}_i - \mathbf{y}_c\|)(\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}}.$$

Here we have used Cartesian multi-index notation: $\mathbf{k} = (k_1, \dots, k_d)$, $\|\mathbf{k}\| = k_1 + \cdots + k_d$, $\mathbf{k}! = k_1! \cdots k_d!$, $\mathbf{y}^{\mathbf{k}} = y_1^{k_1} \cdots y_d^{k_d}$, and $D_{\mathbf{y}}^{\mathbf{k}} = D_{y_1}^{k_1} \cdots D_{y_d}^{k_d}$.

Assuming a constant shape parameter $\epsilon_j \equiv \epsilon$, the key step is to interchange the sums to get a Taylor series for the entire cell,

$$\begin{aligned}\sum_{\mathbf{y}_j \in C} w_j \phi(\epsilon_j \|\mathbf{x}_i - \mathbf{y}_j\|) &\overset{\epsilon_j \equiv \epsilon}{=} \sum_{\|\mathbf{k}\|=0}^{\infty} \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \phi(\epsilon \|\mathbf{x}_i - \mathbf{y}_c\|) \sum_{\mathbf{y}_j \in C} w_j (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} \\ &\approx \sum_{\|\mathbf{k}\|=0}^{p} a_{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_c) m_{\mathbf{k}}(C),\end{aligned}$$

where p is the order of the truncated series and

$$m_{\mathbf{k}}(C) = \sum_{\mathbf{y}_j \in C} w_j (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}}$$

is called a moment of the cell C. The moments are independent of evaluation points and can be saved with C for use at future evaluation nodes. Krasny and Wang established a recurrence relation for $a_{\mathbf{k}}(\mathbf{x}, \mathbf{y}) = \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \phi(\epsilon \|\mathbf{x} - \mathbf{y}\|)$ so that the computation of each Taylor expansion coefficient takes constant time. Finally, the sum $s(\mathbf{x}_i)$ can be computed efficiently in $O(\log N)$ time overall by adding the contributions of all well-separated cells and individual interactions:

$$s(\mathbf{x}_i) = \sum_j w_j \phi(\epsilon \|\mathbf{x}_i - \mathbf{y}_j\|) \approx \sum_{C \in T_i} \sum_{\|\mathbf{k}\|=0}^{p} a_{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_c) m_{\mathbf{k}}(C). \tag{2.1}$$

Here we have used T_i as shorthand for the sets of tree cells that interact with point $\mathbf{x}_i$. In practice these sets are determined for each i by depth-first search on $\mathcal{T}$ for satisfaction of the multipole acceptance criterion (MAC) described in section 2.2.

The assumption $\epsilon_j \equiv \epsilon$ is critical to Krasny and Wang's method. Intuitively, basis functions whose centers are close together but whose shape parameters are very different cannot easily be represented by a single expansion. Instead, we take a dual approach of clustering evaluation nodes rather than centers. This is akin to clustering by rows in a matrix-vector product rather than by columns. Accordingly, while the tree structure $\mathcal{T}$ is similar to the previous case, it is associated with nodes, not centers, and each leaf of $\mathcal{T}$ contains exactly one $\mathbf{x}_i$. Assuming a uniform distribution of the evaluation points, the height of the tree is $O(\log M)$ on average.

We now will express the value of the RBF sum $s(\mathbf{x}_i)$ as

$$s(\mathbf{x}_i) = \sum_{C \in H(\mathbf{x}_i)} s_C, \tag{2.2}$$

where $H(\mathbf{x}_i)$ is the hierarchy of cells in the tree that contain the node $\mathbf{x}_i$, and s_C represents the accumulated interaction between cell C and centers well-separated from it in the sense of the criterion to be explained in section 2.2. We denote the centroid of cell C as $\mathbf{x}_c$, and the set of centers satisfying the MAC with C as F_C.

If C is a leaf (containing just one evaluation point), we use direct interaction to compute s_C. Otherwise, we can employ a far-field expansion. Denoting $\phi_j(\mathbf{x}_i, \mathbf{y}_j) = \phi(\epsilon_j \|\mathbf{x}_i - \mathbf{y}_j\|)$,

$$\begin{aligned} s_C &= \sum_{\mathbf{y}_j \in F_C} w_j \phi_j(\mathbf{x}_i, \mathbf{y}_j) \\ &\approx \sum_{\mathbf{y}_j \in F_C} w_j \sum_{\|k\|=0}^{p} \frac{1}{\mathbf{k}!} (D_{\mathbf{x}}^{\mathbf{k}} \phi_j(\mathbf{x}_c, \mathbf{y}_j))(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \\ &= \sum_{\|\mathbf{k}\|=0}^{p} \left[\sum_{\mathbf{y}_j \in F_C} w_j \frac{1}{\mathbf{k}!} (D_{\mathbf{x}}^{\mathbf{k}} \phi_j(\mathbf{x}_c, \mathbf{y}_j)) \right] (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \\ &= \sum_{\|\mathbf{k}\|=0}^{p} \left[\sum_{\mathbf{y}_j \in F_C} w_j a^{\mathbf{k}}(\mathbf{x}_c, \mathbf{y}_j) \right] (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}}, \end{aligned} \tag{2.3}$$

where $a^{\mathbf{k}}(\mathbf{x}_c, \mathbf{y}_j) = \frac{1}{\mathbf{k}!} D_{\mathbf{x}}^{\mathbf{k}} \phi_j(\mathbf{x}_c, \mathbf{y}_j)$ are Taylor coefficients that can be derived from a recurrence relation to be given below. We call $\sum_{\mathbf{y}_j \in F_C} w_j a^{\mathbf{k}}(\mathbf{x}_c, \mathbf{y}_j)$ the *potential* at cell C, denoted as $P_{\mathbf{k}}(C)$. In other words, $P_{\mathbf{k}}(C)$ contains the total order-$\mathbf{k}$ contribution to cell C of all centers well-separated from C. None of these centers will interact otherwise with the descendants of C in the tree $\mathcal{T}$.

The recurrence relation used to compute the coefficients of the Taylor expansion in (2.3) is similar to that developed by Krasny and Wang for their treecode. In d dimensions, if we use $a^{\mathbf{k}}(\mathbf{x}, \mathbf{y}) = a^{(k_1, \dots, k_d)}(\mathbf{x}, \mathbf{y})$ to denote the coefficients, then $a^{\mathbf{k}}(\mathbf{x}, \mathbf{y})$ satisfies the following recurrence relation:

$$\|\mathbf{k}\|(\epsilon^2 \|\mathbf{x} - \mathbf{y}\|^2 + 1) a^{\mathbf{k}} = \epsilon^2 (2\|\mathbf{k}\| - 3) \sum_{i=1}^{d} (\mathbf{x}_i - \mathbf{y}_i) a^{\mathbf{k} - \mathbf{e}_i} + \epsilon^2 (\|\mathbf{k}\| - 3) \sum_{i=1}^{d} a^{\mathbf{k} - 2\mathbf{e}_i}, \tag{2.4}$$

where $a^{\mathbf{0}} = \phi_j(\mathbf{x}_c, \mathbf{y}_j)$, $a^{\mathbf{k}} = 0$ if any $k_i < 0$, $\|\mathbf{k}\| = k_1 + \cdots + k_d$, $\mathbf{e}_i$ are unit vectors, and ϵ is the shape parameter associated with $\mathbf{y}$. The proof of this relation is similar to that given by Krasny and Wang [22], so we omit it here.

2.2. Multipole acceptance criterion.

The MAC is used to check whether a source point and a cell of evaluation points are well enough separated to allow effective far-field expansion. The choice of the MAC has a significant effect on the efficiency of the algorithm. In the past,

$$\theta_0 \geq \theta = \frac{r}{R} \tag{2.5}$$

was used as a MAC for the computation of Newtonian potential [1], where r is the cell radius, R is the distance between the source and the centroid of the cell, and θ_0 is a user-selected parameter. Krasny and Wang [22] proposed the MAC

$$\theta_0 \geq \theta = \frac{\epsilon r}{\sqrt{\epsilon^2 R^2 + 1}}. \tag{2.6}$$

This MAC considers the impact of ϵ, and through experiments we find that it is relaxed to some extent relative to (2.5), such that more potentials tend to be computed at earlier stages of the computation—i.e., closer to the root of the tree; thus the treecode employing this MAC should save computation time. Because we anticipate that shape parameters may range over orders of magnitude, we will adopt this MAC in our experiments.

2.3. Summary of the algorithm. The computation of $s(\mathbf{x}_i)$, $i = 1, \ldots, M$, is divided into two phases: potential computation to get s_C for each cell of the evaluation point tree $\mathcal{T}$ by (2.3), and influence accumulation to compute $s(\mathbf{x}_i)$ by (2.2). The complete algorithm is presented in Algorithm 1.

Potential computation is illustrated in Figure 1. For each RBF center (source point) $\mathbf{y}_j$, we start from the root of $\mathcal{T}$. If the MAC is satisfied for the cell C, we add a contribution to the potentials of C due to interaction with $\mathbf{y}_j$. Otherwise, we recursively call the children of C, or add a direct contribution to C if it is a leaf containing just one evaluation point. The potential computation phase takes $O(N \log M)$ on average, where N is the result of looping over all centers, and $\log M$ is the average depth of $\mathcal{T}$ for a uniformly distributed set of evaluation points.

Influence accumulation is illustrated in Figure 2. We start at the root of the tree and execute a depth-first search. For each searching direction (or following each tree branch), we create a stack of cells encountered in this direction. When a leaf is reached containing the single evaluation point $\mathbf{x}_i$, we add all the contributions from potentials following the cells stored in the stack to the direct influences already stored for $s(\mathbf{x}_i)$. There are M stacks to be built in this process, where each stack keeps $O(\log M)$ cells on average. So the stack size is $O(M \log M)$ in the ideal case when evaluation points are uniformly distributed. Counting both phases of the algorithm, the work is $O((M + N) \log M)$ when the evaluation points are uniformly distributed.

3. Restricted Schwarz preconditioner for determining coefficients. In the previous section we introduced a treecode that provides efficient computation of Aw, which makes iterative solvers like GMRES attractive. However, multiquadric RBF interpolation generally leads to a highly ill-conditioned system [21], which makes GMRES iteration converge slowly. Instead, one must solve a preconditioned system $M^{-1}Aw = M^{-1}f$ for some M selected to give us a much better convergence performance. There are many reported successful preconditioners for RBFs [2, 24, 28].

Our approach is to combine two existing ideas. Cai and Sarkis [8] introduced the restricted additive Schwarz method to solve general sparse linear systems and showed it to have superior performance over the traditional additive Schwarz method. Yokota, Barba, and Knepley [33] applied it in RBF interpolation with a Gaussian basis function. Separately, Beatson, Cherrie, and Mouat designed a two-level preconditioner [2], where a coarse grid of uniformly distributed nodes is coupled with the fine grid to improve the conditioning.

We combine the two ideas and introduce an augmented restricted additive Schwarz method (ARASM). Because the preconditioner is applied only when the centers and interpolation nodes are the same, the point tree $\mathcal{T}$ is also a tree for the RBF centers. We descend through the tree until we reach a cell C that contains no more than a prescribed number K of centers, and we call this cell a *restricted domain*. Applying this search throughout the tree, we eventually divide the region into nonoverlapping restricted domains. The boundaries of these domains are extended by a fraction γ in each dimension to form overlapping *computational domains*. Finally, a small subset of centers is chosen as a coarse, global set for the region. The coarse set allows global influence and represents a compromise between trying to keep the number of GMRES iterations constant, which would require a fully multiscale approach, and trying to keep the cost of the preconditioner from growing too quickly.

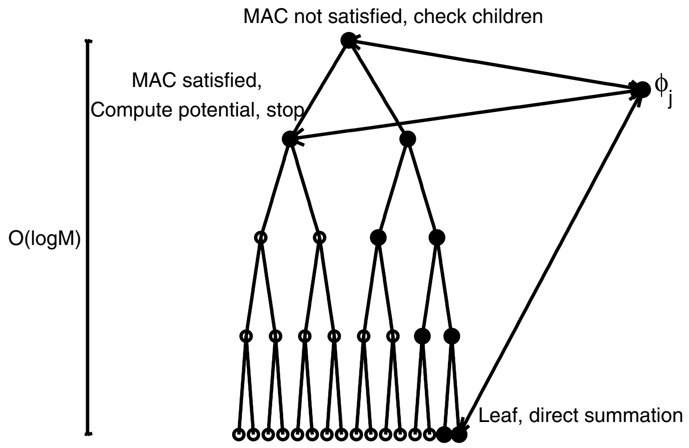


FIG. 1. *Potential computation phase of the treecode algorithm. Filled dots are the cells of tree $\mathcal{T}$ visited to interact with the single source term ϕ_j. Empty dots indicate cells not visited due to the clustering of interactions.*

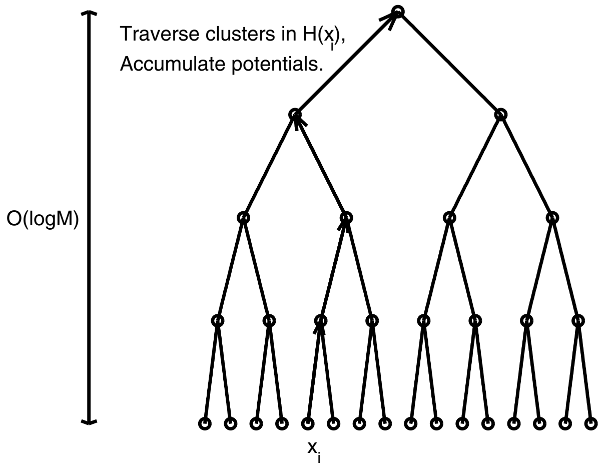


FIG. 2. *Influence accumulation phase of the treecode algorithm. The distribution of nodes determines the balance of the tree. Here we assume nodes are distributed uniformly in the domain to obtain the desired tree depth $O(\log M)$.*

On each computational domain, we formulate an interpolation subproblem combining the centers in the domain with the coarse set. This smaller system is solved by direct matrix factorization. The results of the solution are then applied only to the restricted domain; we hope to get basis function weights that are close to those for the global problem. The values in the overlap regions are discarded due to edge effects. Figure 3 depicts how the preconditioner works. Mathematically, the ARASM preconditioner can be written as

$$M^{-1} = \widetilde{R}_1^T A_1^{-1} R_1 + \cdots + \widetilde{R}_N^T A_N^{-1} R_N, \tag{3.1}$$

```
input  : Evaluation nodes $\mathbf{x}_i$, $i = 1, \ldots, M$, centers $\mathbf{y}_j$, weights $w_j$, shapes
         $\epsilon_j, j = 1, \ldots, N$, MAC parameter $\theta_0$, expansion order $p$
output: Interpolant values $s(\mathbf{x}_i), i = 1, \ldots, M$

main program;
Build quadtree/octree structure on interpolation nodes $\mathbf{x}_i$;
for $j = 1$ to $N$ do
    ComputePotential($\mathbf{y}_j$,root);
end
Accumulate(root,[]);

subroutine;
ComputePotential($\mathbf{y}_j$,$C$);
if $C$ is leaf then
    do direct summation $s(\mathbf{x}_i) = s(\mathbf{x}_i) + w_j \phi_j(\mathbf{x}_i, \mathbf{y}_j)$;
    (comment: $\mathbf{x}_i$ is the only node contained in leaf $C$);
else
    if $\mathbf{y}_j$,$\mathbf{x}_c$ satisfy MAC($\theta_0$) then
        (comment: $\mathbf{x}_c$ is the center of cell $C$);
        ComputeCoeff $a^{\mathbf{k}}(\mathbf{x}_c, \mathbf{y}_j)$;
        Update potential $P_{\mathbf{k}}(C) = P_{\mathbf{k}}(C) + w_j \cdot a^{\mathbf{k}}(\mathbf{x}_c, \mathbf{y}_j)$;
    else
        for $k = 1$ to number of children of $C$ do
            ComputePotential($\mathbf{y}_j$, child $k$ of $C$);
        end
    end
end

subroutine;
Accumulate ($C$,$H$);
if $C$ is leaf then
    (comment: $\mathbf{x}_i$ is the only node contained in leaf cell $C$);
    for each cell $\tilde{C}$ in hierarchy $H$ do
        $s(\mathbf{x}_i) = s(\mathbf{x}_i) + \sum_{\|\mathbf{k}\|=0}^{p} P_{\mathbf{k}}(\tilde{C})(\mathbf{x}_i - \mathbf{x}_{\tilde{c}})^{\mathbf{k}}$ ;
        (comment: $\mathbf{x}_{\tilde{c}}$ is the centroid of cell $\tilde{C}$);
    end
else
    for each child $\tilde{C}$ of $C$ do
        Accumulate($\tilde{C}$,$[H, C]$);
    end
end
```

ALGORITHM 1. *A fast treecode summation.*

where R_i is the restriction operator that maps to the augmented system of computational domain and coarse set, while $\widetilde{R}_i^T$ is the injection operator that maps elements in a restricted domain to the whole domain.

The performance of the ARASM preconditioner is strongly dependent on the parameters K, the number of centers in a restricted domain, and the domain overlap

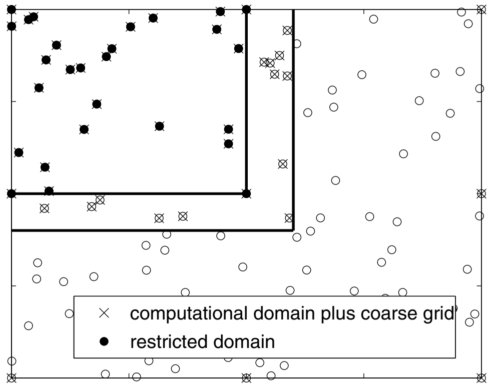


Fig. 3. *A depiction of the ARASM preconditioner. An RBF interpolation problem using the centers in one computational domain and the coarse set is solved to produce coefficients, but these are saved only for centers lying in the smaller restricted domain.*

fraction γ. Increasing γ means that there are more centers in overlap regions, thus increasing the complexity of the preconditioner, while decreasing γ requires more GMRES iterations because the subproblem solution is less accurate. Similarly, increasing K makes the preconditioner both more accurate and more expensive. In the case of a uniformly distributed node set, we can estimate the time complexity of applying the preconditioner. The domain is divided into $O(N/K)$ restricted domains. For each restricted domain, a computation is performed on its augmented system (computational domain plus the coarse set), so the cost to solve problems on subdomains is $O(((1+\gamma)^d K + N_0)^3) = O(K^3)$, where N_0 stands for the number of nodes in the coarse set. So the total cost of the preconditioner will be $O((N/K) \cdot K^3) = O(K^2 N)$. This bound will get more accurate as the ratio K/N_0 increases, but the GMRES performance will suffer and total computation time will increase accordingly. For fixed parameters, this factor is $O(N)$, but we cannot bound the number of GMRES iterations required. In practice, we take advantage of the fact that the preconditioner is used in many GMRES iterations and cache the LU factors belonging to each computational domain during the first GMRES iteration. The cache size for all of this information should be $O((N/K) \cdot (K^2)) = O(KN)$ in the ideal case of uniformly distributed nodes. We mention that the preconditioner is potentially parallelizable for the major effort of solving on different computational domains.

4. Adaptive RBF iteration.

Depending on the application, the center locations $\mathbf{y}_j$ and shape parameters ϵ_j in the RBF interpolant (1.1) may be selected to improve its approximation quality. In various precise and imprecise ways it is well established that centers should be clustered near boundaries in order to avoid interpolation instabilities (Runge-type oscillations) and that both equation conditioning and interpolation accuracy are sensitively dependent on the relationship between shape parameters and local center spacing [7, 9, 29, 30, 32]. However, while theory and experience give useful rules of thumb, and brute force optimization can provide parameter values in small problems, there is no general algorithm available for selecting centers and shape parameters effectively for a large class of functions on a fixed domain.

An alternative to the construction of a universal set of centers and shape parameter values is to select them through adaptation to the targeted function. A number

```
input  : Domain $\Omega \in \mathbf{R}^d$, function $f$, tolerance $\tau$, top-level $\epsilon$
output : RBF centers $\mathbf{y}_j$, shape parameters $\epsilon_j$, and weights $w_j$, $j = 1, \dots, N$;
         tree $\mathcal{T}$
Initialize tree $\mathcal{T}$ for coarse set of centers $\mathbf{y}_j$ with shapes $\epsilon_j$ on $\Omega$;
repeat
    Set flag = False;
    Find RBF interpolant weights for (1.1) on current center set;
    for each leaf $t$ of $\mathcal{T}$ do
        Evaluate residual $\|s(\mathbf{x}) - f(\mathbf{x})\|$ at the centroids of $2^d$ child cells of $t$;
        if maximum residual $> \tau$ then
            Subdivide $t$ into $2^d$ children and add centroids to the center set;
            Set children's shape value to twice that of $t$;
            Set flag = True;
        end
    end
until flag = False;
```

Algorithm 2. *Adaptive residual subsampling iteration for RBF interpolation.*

of greedy adaptive procedures has been proposed [13, 25]; they typically attempt to select nodes optimally from a predetermined candidate set. Driscoll and Heryudono [10, 18] proposed instead adding nodes in a multiscale fashion by subsampling the residual at nodes descending the levels of a quadtree structure. Because we can identify this quadtree with the tree $\mathcal{T}$ required for the fast summation algorithm of section 2 and the ARASM preconditioner of section 3, pairing the adaptive iteration with these methods is straightforward.

An outline of the adaptive residual subsampling algorithm is shown in Algorithm 2. Initially, a coarse set of centers is created by refining a cell that contains the domain Ω several times to get an initial tree $\mathcal{T}$. If Ω is not rectilinear, this step can include geometric refinement to adapt to boundaries [19]. Then the RBF interpolant weights w_j are found for the current center set. The error of this interpolant (or, for a collocated PDE, the residual) is evaluated at the points that lie one level below the leaves of $\mathcal{T}$. Any time a leaf has a descendant where the residual is too large, that subdivision is made part of the tree and the descendants are added to the center set. It is important to note that the shape parameters of the new centers are double that of their parent, in order to maintain scale invariance. (In a more sophisticated form of the algorithm, a center may also be removed if it and its siblings are all far below the error threshold.) The process repeats until no more centers are added.

The method has been observed to produce node sets that cluster near boundaries and near salient features of the interpolated function or PDE solution. The multiscale nature of the method makes it possible to represent a jump of relative width 10^{-9} with a few hundred nodes. Because of the doubling of shape parameters as one descends the tree, the condition number of the RBF interpolation matrix does not tend to grow as the number of nodes increases.

For the numerical experiments of section 5, we use ARASM preconditioning and fast multiquadric summation to perform the solution for RBF weights. Clearly, since the tree $\mathcal{T}$ is updated from one adaptive iteration to the next rather than created from scratch for each GMRES problem, we could find significant efficiencies in updating summation potential data and preconditioner subproblems. For the experiments we report, however, we treat each new tree $\mathcal{T}$ as a new problem.

5. Numerical results.

We now present some numerical experiments on two aspects of the treecode: the time and accuracy performance of the treecode for multiquadric RBFs with varying shape parameters, and the adaptive iterative interpolation using the treecode-based GMRES method with the ARASM preconditioner.

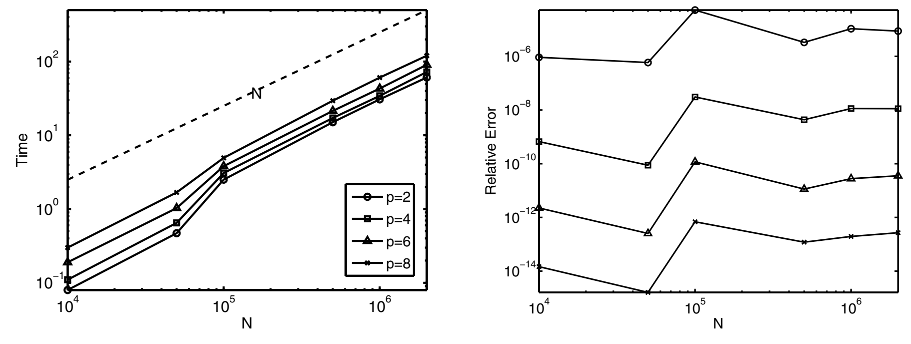
In our experiments we chose the following parameter values: threshold $\theta_0 = 0.5$ in (2.6) for the MAC, domain overlap fraction $\gamma = 0.2$ and restricted domain size $K = 800$ in the ARASM preconditioner, $\epsilon = 10$ for the shape parameter of the top-level multiquadric functions in the adaptive iteration, and expansion order $p = 12$ to maintain necessary accuracy of the treecode in the GMRES process. Stated errors are always in the relative discrete 2-norm (sum-of-squares), made relative to the given exact function.

5.1. Treecode performance.

In order to test the performance of the treecode, we used uniform random numbers to generate a node set $\{\mathbf{x}_j\}$, $j = 1, \dots, N$, in domain $[-1, 1]^d$, $d = 2, 3$, and chose the center set equal to the node set. Shape parameters $\{\epsilon_j\}$, $j = 1, \dots, N$, in $[0, 1]$ and weights $\{w_j\}$, $j = 1, \dots, N$, in $[-1, 1]$ were both generated by uniform random numbers. We have also tested node sets created from the normal distribution, with no observable difference in the asymptotic behavior.

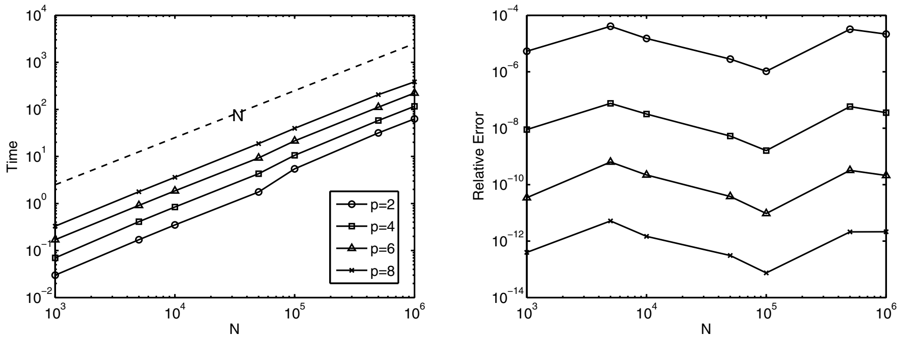
Figure 4(a) shows the elapsed time in seconds compared to the size of the node set, where different orders of truncated Taylor series are compared. The figure is consistent with the time complexity $O(N \log N)$. Figure 4(b) gives the relative error of interpolation compared to the size of node set. It shows that the error remains relatively constant with N and improves geometrically with truncation order p. Figure 5 shows similar results for random RBF sums in three dimensions.

The dual treecode is very similar to Krasny and Wang's treecode when shape parameters are all the same. Since the new treecode uses two phases instead of one, times increase by about 30% for the more general method.



(a) time performance (b) accuracy performance

FIG. 4. *Treecode performance for evaluating random RBF sums at N points in $d = 2$ dimensions. In* (a)*, time performance for different orders of truncation p is nearly $O(N)$. In* (b)*, the relative error remains constant with N and improves with p.*



(a) time performance (b) accuracy performance

FIG. 5. *Treecode performance for evaluating random RBF sums in $d = 3$ dimensions. Compare to Figure* 4.

5.2. Adaptive interpolation with treecode and preconditioned GMRES. As mentioned in section 4, adaptive RBF interpolation requires the shape parameter of the centers to vary in order to maintain scale invariance and bound the growth of the condition number of the linear system. In this section we apply treecode summation in the context of a GMRES iteration with the ARASM preconditioner from section 3. Since the treecode will perform in $O((M + N)\log M)$ time complexity for uniformly distributed nodes, where M is the number of nodes and N is the number of centers, we expect that each GMRES iteration will require $O(N \log N)$ time complexity. However, by the nature of adaptive RBF interpolation, nodes will cluster near boundaries and features. As a result, the complexity estimate for the ideal case may not be achieved.

All of the numerical experiments are done in two dimensions on the domain $[-1, 1] \times [-1, 1]$. We tested the Franke function $f(x, y) = \text{franke}(x, y)$ [12] and hyperbolic tangent function $f(x, y) = \tanh(10(x - 2y))$ at a range of requested interpolation error tolerances. We set the tolerance for each GMRES iteration to be the same as the tolerance τ used in the fast adaptive interpolation.

Tables 5.1 and 5.2 show the behavior of adaptive interpolation for our test functions. The first column specifies the tolerance parameter τ of Algorithm 2. The second column gives the number of interpolation nodes at the end of the adaptation process. Columns three through six present items about solving for coefficients in the final step of the adaptive interpolation: the time used for treecode summation, the time used for GMRES iterations, the number of GMRES iterations, and the time for each GMRES iteration, respectively. This data is consistent with $O(N \log N)$ time as required by the treecode. The last column reports the time for the whole adaptation. This spans all of the adaptive iterations and therefore grows more rapidly than $O(N \log N)$.

In each case, both the time spent in the treecode and the time in each GMRES iteration scale consistently with $O(N \log N)$ despite the very nonuniform distribution of centers. The number of GMRES iterations required does grow slowly with N. If computational domains are chosen to give a fixed overlap rather than a fixed fraction of the restricted domains, the number of iterations that can be made is $O(1)$, but at the cost of much more expensive preconditioner applications, resulting in a larger total time.

TABLE 5.1
Fast adaptive interpolation applied to Franke's function.

tol.	N	treecode (sec)	GMRES (final node set) time (sec)	iter.	time per iter. (sec)	total time (sec)
1e-5	1250	0.131	0.743	3	0.247	4.433
1e-6	3709	0.476	3.303	3	1.101	25.096
5e-7	5016	0.682	5.502	4	1.375	36.254
2e-7	7968	1.191	11.704	5	2.341	86.820
1e-7	11158	1.766	20.271	6	3.378	191.534
5e-8	14844	2.446	36.109	8	4.514	253.936
2e-8	22305	3.914	53.717	8	6.715	416.183
1e-8	33264	6.120	113.499	10	11.350	1064.3
5e-9	44986	8.611	239.729	15	15.982	1826.9

TABLE 5.2
Fast adaptive interpolation applied to $\tanh 10(x-2y)$.

tol.	N	treecode (sec)	GMRES (final node set) time (sec)	iter.	time per iter. (sec)	total time (sec)
1e-4	1274	0.160	0.734	2	0.367	4.147
1e-5	3330	0.503	3.451	3	1.150	23.120
5e-6	4434	0.696	6.01	4	1.502	35.087
3e-6	6248	1.059	8.967	4	2.242	63.352
1e-6	10288	1.876	21.501	6	3.583	176.822
3e-7	16310	3.141	46.683	7	6.669	444.216
2e-7	21319	4.340	80.962	9	8.996	494.304
1e-7	32250	6.807	197.074	15	13.136	2328.821
5e-8	41388	8.990	255.176	13	19.629	2471.300

As described in section 3, the ARASM preconditioner saves a great deal of time by caching the LU factors of subproblem matrices for use throughout a single GMRES solution. Figure 6 shows that the growth rate of the cache size is between $O(N)$, which is the ideal case for uniformly scattered nodes, and $O(N^2)$, which is the worst case. It shows that our estimation for the space complexity is not achieved, presumably as a result of the nonuniform distribution of nodes from the adaptive interpolation.

6. Concluding remarks.

Practical uses of multiquadric RBF approximations often employ varying shape parameters to control the conditioning of the coefficient solution and the accuracy of the resulting interpolant. We have demonstrated a treecode that clusters evaluation points to achieve fast summation for such RBF approximants, even when the shape parameters vary over orders of magnitude.

The treecode algorithm can share the tree data structure with a previously described adaptive interpolation procedure based on residual measurement on subgrids. A relatively simple two-level restricted additive Schwarz preconditioner has proven effective for using GMRES to find RBF coefficients in moderately large problems. It is also potentially parallelizable, but will possibly be inadequate for large problems, in which case multilevel or other preconditioners may become relevant.

A large part of the interest in RBF interpolation is its application in collocation or method-of-lines approximations to PDEs. The algorithm presented in this work applies only to evaluation of the interpolant itself, but it can be adapted straightforwardly to evaluate derivatives of the interpolant, at least in some cases. For example, evaluation of the Laplacian of the interpolant (1.1) requires evaluation of $\phi''(r)$ and

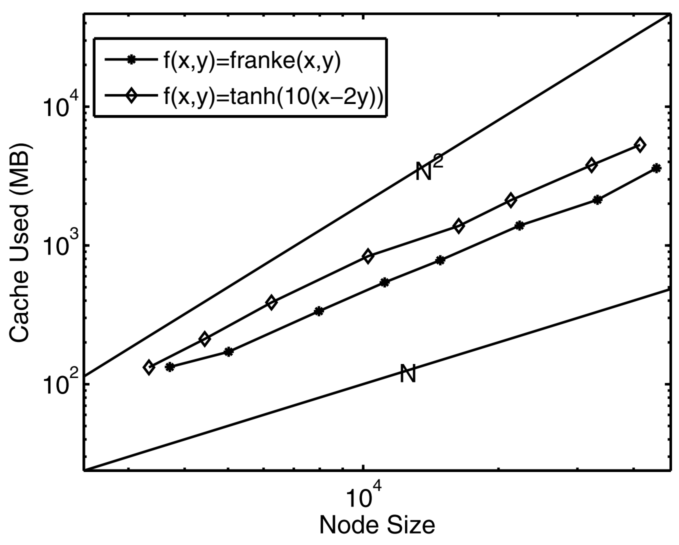


FIG. 6. *ARASM preconditioner cache size against N for our two adaptive interpolation test functions.*

$\phi'(r)/r$. Each of these terms has its own recursion relation similar to (2.4):

$$\|\mathbf{k}\|(\epsilon^2\|\mathbf{x}-\mathbf{y}\|^2+1)a^{\mathbf{k}} = \epsilon^2(2\|\mathbf{k}\|+1)\sum_{i=1}^{d}(\mathbf{x}_i-\mathbf{y}_i)a^{\mathbf{k}-\mathbf{e}_i} - \epsilon^2(\|\mathbf{k}\|+1)\sum_{i=1}^{d}a^{\mathbf{k}-2\mathbf{e}_i},$$

$$\|\mathbf{k}\|(\epsilon^2\|\mathbf{x}-\mathbf{y}\|^2+1)b^{\mathbf{k}} = \epsilon^2(2\|\mathbf{k}\|-1)\sum_{i=1}^{d}(\mathbf{x}_i-\mathbf{y}_i)b^{\mathbf{k}-\mathbf{e}_i} - \epsilon^2(\|\mathbf{k}\|-1)\sum_{i=1}^{d}b^{\mathbf{k}-2\mathbf{e}_i}.$$

These coefficients could be evaluated within the treecode similarly to those for ϕ itself, so that an $O(N \log N)$ method results for use in iterative linear algebra or time stepping. We have not yet attempted to apply this capability to a PDE.

Significant challenges to adaptive treecodes for multiquadrics remain. We have not investigated the possibility of updating the summation moments between adaptive node selection iterations, rather than starting over with each new adaptive node set. In addition, the number of moments required to achieve pth order in d dimensions is $O(p^d)$, which makes the global multiquadric approach less attractive in high dimensions, and the number of points in each quadtree/octree refinement grows as 2^d, which is problematic for the adaptive strategy in high dimensions. Alternative data structures such as the k-d tree [12] may be useful for controlling tree growth.

REFERENCES

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

[2] R. Beatson, J. Cherrie, and C. Mouat, *Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration*, Adv. Comput. Math., 11 (1999), pp. 253–270.

[3] R. K. Beatson, J. B. Cherrie, and D. L. Ragozin, *Fast evaluation of radial basis functions: Methods for four-dimensional polyharmonic splines*, SIAM J. Math. Anal., 32 (2001), pp. 1272–1310.

[4] R. Beatson and L. Greengard, *A short course on fast multipole methods*, in Wavelets, Multilevel Methods and Elliptic PDEs, Numer. Math. Sci. Comput., Oxford Univ. Press, New York, 1997, pp. 1–37.

[5] 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.

[6] R. K. Beatson, W. A. Light, and S. Billings, *Fast solution of the radial basis function interpolation equations: Domain decomposition methods*, SIAM J. Sci. Comput., 22 (2000), pp. 1717–1740.

[7] M. Buhmann and N. Dyn, *Spectral convergence of multiquadric interpolation*, Proc. Edinburgh Math. Soc. (2), 36 (1993), pp. 319–333.

[8] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.

[9] T. Driscoll and B. Fornberg, *Interpolation in the limit of increasingly flat radial basis functions*, Comput. Math. Appl., 43 (2002), pp. 413–422.

[10] 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.

[11] 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.

[12] G. Fasshauer, *Meshfree Approximation Methods with MATLAB*, World Scientific Publishing, Hackensack, NJ, 2007.

[13] R. Franke, H. Hagen, and G. M. Nielson, *Least squares surface approximation to scattered data using multiquadratic functions*, Adv. Comput. Math., 2 (1994), pp. 81–99.

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

[15] N. A. Gumerov and R. Duraiswami, *Fast radial basis function interpolation via preconditioned Krylov iteration*, SIAM J. Sci. Comput., 29 (2007), pp. 1876–1899.

[16] H. Wendland, *Scattered Data Approximation*, Cambridge University Press, Cambridge, UK, 2005.

[17] R. L. Hardy, *Multiquadric equations of topography and other irregular surfaces*, J. Geophys. Res., 76 (1971), pp. 1905–1915.

[18] A. R. H. Heryudono, *Adaptive Radial Basis Function Methods for the Numerical Solution of Partial Differential Equations, with Application to the Simulation of the Human Tear Film*, Ph.D. thesis, University of Delaware, Newark, DE, 2008.

[19] A. R. H. Heryudono and T. A. Driscoll, *Radial basis function interpolation on irregular domain through conformal transplantation*, J. Sci. Comput., 44 (2010), pp. 286–300.

[20] E. Kansa, *Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics* II. *Solutions to parabolic, hyperbolic and elliptic partial differential equations*, Comput. Math. Appl., 19 (1990), pp. 147–161.

[21] E. Kansa, *Circumventing the ill-conditioning problem with multiquadric radial basis functions: Applications to elliptic partial differential equations*, Comput. Math. Appl., 39 (2000), pp. 123–137.

[22] R. Krasny and L. Wang, *Fast evaluation of multiquadric RBF sums by a Cartesian treecode,* SIAM J. Sci. Comput., 33 (2011), pp. 2341–2355.

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

[24] L. Ling and E. J. Kansa, *A least-squares preconditioner for radial basis functions collocation methods*, Adv. Comput. Math., 23 (2005), pp. 31–54.

[25] T. Lyche and K. Mørken, *Knot removal for parametric B-spline curves and surfaces*, Comput. Aided Geom. Design, 4 (1987), pp. 217–230.

[26] W. Maydch and S. Nelson, *Multivariate Interpolation and Conditionally Positive Definite Functions*, Math. Comput., 54 (1990), pp. 211–230.

[27] C. A. Micchelli, *Interpolation of scattered data: Distance matrices and conditionally positive definite functions*, Constr. Approx., 2 (1986), pp. 11–22.

[28] C. T. Mouat, *Fast Algorithms and Preconditioning Techniques for Fitting Radial Basis Function*, Ph.D. thesis, University of Canterbury, 2001.

[29] R. B. Platte and T. A. Driscoll, *Polynomials and potential theory for Gaussian radial basis function interpolation*, SIAM J. Numer. Anal., 43 (2005), pp. 750–766.

[30] R. Schaback, *Error estimates and condition numbers for radial basis function interpolation*, Adv. Comput. Math., 3 (1995), pp. 251–264.

[31] R. Schaback, *Multivariate interpolation and approximation by translates of a basis function*, in Approximation Theory VIII. Vol. 1, C. K. Chui and L. L. Schumaker, eds., Ser. Approx. Decompos. 6, World Scientific Publishing, River Edge, NJ, 1995, pp. 491–514.

[32] R. Schaback, *On the efficiency of interpolation by radial basis functions*, in Surface Fitting and Multiresolution Methods, Vanderbilt University Press, Nashville, TN, 1997, pp. 309–318.

[33] R. Yokota, L. Barba, and M. G. Knepley, *PetRBF—a parallel O(N) algorithm for radial basis function interpolation with Gaussians*, Comput. Methods Appl. Mech. Engrg., 199 (2010), pp. 1793–1804.