Electronic Transactions on Numerical Analysis.
Volume 23, pp. 263-287, 2006.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

FAST MULTILEVEL EVALUATION OF SMOOTH RADIAL BASIS FUNCTION EXPANSIONS*

OREN E. LIVNE† AND GRADY B. WRIGHT‡

Abstract. Radial basis functions (RBFs) are a powerful tool for interpolating/approximating multidimensional scattered data. Notwithstanding, RBFs pose computational challenges, such as the efficient evaluation of an n-center RBF expansion at m points. A direct summation requires $O(nm)$ operations. We present a new multilevel method whose cost is only $O((n+m)\ln(1/\delta)^d)$, where δ is the desired accuracy and d is the dimension. The method applies to smooth radial kernels, e.g., Gaussian, multiquadric, or inverse multiquadric. We present numerical results, discuss generalizations, and compare our method to other fast RBF evaluation methods. This multilevel summation algorithm can be also applied beyond RBFs, to discrete integral transform evaluation, Gaussian filtering and deblurring of images, and particle force summation.

Key words. Radial basis functions, fast multilevel multi-summation, integral transforms, particle interaction

AMS subject classifications. 41A21, 41A30, 41A63, 65D25, 65N06, 65R10, 68Q25

1. Introduction. In many science and engineering disciplines we need to interpolate or approximate a function $f:\mathbb{R}^d \to \mathbb{R},\ d \geq 1$ from a discrete set of scattered samples. Radial basis functions (RBFs) are a simple and powerful technique for solving this problem, with applications to cartography [31], neural networks [27], geophysics [9, 10], pattern recognition [41], graphics and imaging [16, 17, 44], and the numerical solution of partial differential equations [33, 34].

The basic RBF approximation to $f(\mathbf{x})$ is given by the *expansion*

$$s(\mathbf{x}) = \sum_{j=0}^{n} \lambda(\mathbf{y}_j)\phi\left(\|\mathbf{x}-\mathbf{y}_j\|\right)\ , \tag{1.1}$$

where boldface symbols are in $\mathbb{R}^d$, $\|\cdot\|$ denotes the d-dimensional Euclidean norm, $\{\mathbf{y}_j\}_{j=0}^n$ are called *centers*, $\lambda(\mathbf{y}_j)$ are the *expansion coefficients*, and ϕ is a univariate, *radially symmetric kernel*. Given *data* $f_j = f(\mathbf{y}_j)$, $j=0,1,\ldots,n$, the expansion coefficients are chosen to satisfy the interpolation conditions $s(\mathbf{y}_j) = f_j$, $j=0,1,\ldots,n$, or in matrix notation, to solve

$$\begin{bmatrix} & & \\ & \Phi & \\ & & \end{bmatrix}\begin{bmatrix} \\ \boldsymbol{\lambda} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{f} \\ \\ \end{bmatrix}, \ \text{where } \Phi_{i,j} = \phi\left(\|\mathbf{y}_i-\mathbf{y}_j\|\right)\ . \tag{1.2}$$

More generally, one can choose more data than centers and solve the corresponding overdetermined system for $\boldsymbol{\lambda}$ [15, Ch. 8]. Some common choices of ϕ are listed in Table 1.1. The well-posedness of (1.2) is discussed in [18, Ch. 12–16].

Although RBFs have been effectively applied to many applications, their wider adoption has been hindered by a prohibitively high, non-scalable computational cost, mainly stemming from the infinite support of the commonly used radial kernels [25]. The two main computational challenges of RBFs are

*Received July 12, 2005. Accepted for publication February 16, 2006. Recommended by S. Ehrich.

†Scientific Computing and Imaging Institute, University of Utah, 50 South Central Campus Drive, Room 3490, Salt Lake City, UT 84112, USA. (livne@sci.utah.edu).

‡Department of Mathematics, University of Utah, 155 South 1400 East, Room 233, Salt Lake City, UT 84112-0090, USA. (wright@math.utah.edu).

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 1.1
Some commonly used radial basis functions. In all cases, $\varepsilon > 0$.

Type of Radial Kernel	$\phi(r)$
Smooth Kernels	
Gaussian (GA)	$e^{-(\varepsilon r)^2}$
Generalized multiquadric (GMQ)	$(1+(\varepsilon r)^2)^{\nu/2},\ \nu \neq 0$ and $\nu \notin 2\mathbb{N}$
• Multiquadric (MQ)	$(1+(\varepsilon r)^2)^{1/2}$
• Inverse multiquadric (IMQ)	$(1+(\varepsilon r)^2)^{-1/2}$
• Inverse quadratic (IQ)	$(1+(\varepsilon r)^2)^{-1}$
Piecewise Smooth Kernels	
Generalized Duchon spline (GDS)	$r^{2k}\ln r,\ \ k \in \mathbb{N}$
	$r^{2\nu},\ \ \nu > 0$ and $\nu \notin \mathbb{N}$
Matérn	$\dfrac{2^{1-\nu}}{\Gamma(\nu)} r^{\nu} K_{\nu}(r),\ \ \nu > 0$
Wendland [43]	$(1-r)_+^k p(r),\ \ p =$ polynomial, $k \in \mathbb{N}$
Oscillatory Kernels	
J-Bessel (JB) [24]	$\phi_d(r) = \dfrac{J_{\frac{d}{2}-1}(\varepsilon r)}{(\varepsilon r)^{\frac{d}{2}-1}},\ \ d = 1, 2, \ldots$

(A) *Fitting:* Given $\{\mathbf{y}_j\}_{j=0}^n$ and $\{f_j\}_{j=0}^n$, determine the expansion coefficients $\{\lambda(\mathbf{y}_j)\}_{j=0}^n$. Because Φ in (1.2) is a *dense* $(n+1)$-by-$(n+1)$ symmetric matrix, standard direct solvers require $O(n^3)$ operations.

(B) *Evaluation:* Given $\{\mathbf{y}_j\}_{j=0}^n$ and $\{\lambda(\mathbf{y}_j)\}_{j=0}^n$, evaluate the RBF interpolant (1.1) at multiple points $\mathbf{x} = \mathbf{x}_i,\ \ i = 0, 1, \ldots, m$. The cost of direct summation of (1.1) for all $\mathbf{x}_i$ is $O(mn)$.

In practice, it is sufficient to solve (A) and (B) up to some specified error tolerance. A few Krylov-subspace iterative methods have been developed for (A) [2, 3, 21, 22, 23]. These algorithms require the ability to efficiently multiply Φ by a vector. Thus, overcoming (B) is also important to overcoming (A).

In this paper we develop a fast multilevel evaluation algorithm for reducing the computational cost of (B) to $O((m+n)(\ln(1/\delta))^d)$ operations, where δ is the desired evaluation accuracy. The method is applicable to any smooth ϕ (e.g., Table 1.1, top section) in any dimension d, and to any centers $\{\mathbf{y}_j\}_{j=0}^n$ and evaluation points $\{\mathbf{x}_i\}_{i=0}^m$. The idea is that a smooth ϕ can be accurately represented on a coarser grid of fewer centers and evaluation points, at which a direct summation of (1.1) is less expensive. This approach builds on Brandt's multilevel evaluation of integral transforms [11]; because of the smoothness of ϕ, our method only requires two levels.

Alternative fast RBF expansion methods are:

- *Fast Multipole Method (FMM).* Beatson and Newsam [7] originally developed an FMM for evaluating RBF expansions with the $k = 1$ GDS kernel (see Table 1.1). More recently, Beatson and colleagues have extended the FMM to other radial kernels [4, 5, 6, 7, 19]. The evaluation complexity is $O((m+n)(\ln n)(\ln(1/\delta))^{d+1})$, where δ is the desired accuracy. While these methods have been successful in applications [9, 10, 17], they are rather complicated to program – especially in higher dimensions, because of the complex hierarchical structure and tree codes required to

ETNA
Kent State University
etna@mcs.kent.edu

decompose ϕ into "near field" and "far field" components [15, §7.3]. Additionally, a different set of Laurent and Taylor coefficients must be determined for each new radial kernel and dimension d.

- *Fast Gauss Transform (FGT).* Roussos and Baxter [38, 39] adapted the Fast Gauss Transform (FGT) of Greengard and Strain [30] to RBF evaluation. Exploiting the conditionally positive (negative) definite properties of certain radial kernels, (1.1) is replaced by the evaluation of (1.1) with the GA kernel using FGT, followed by an appropriate Gaussian quadrature rule for translating the GA result back to the original kernel. The algorithm's cost is $O(m+n)$, where the constant increases with the desired accuracy and dimension. FGT is non-hierarchical and parallelizable, but it is hard to precisely control the evaluation error due to the complicated quadrature rules.

The main advantages of our method include:

(i) Its implementation is simple and easily parallelizable for *all* smooth kernels ϕ in *any* dimension d.

(ii) The evaluation error of the method normally depends only simple bounds on the derivatives of $\phi(r)$ at $r = 0$ (such bounds for many useful kernels are given in Appendix B).

(iii) The accuracy and complexity of any fast evaluation method must depend on ϕ and thus on the "shape parameter" ε. In practice, ε is required to grow with n to avoid severe numerical ill-conditioning of the linear system (1.2) for computing the expansion coefficients [40]. Unlike the FMM and FGT methods, we explicitly include the shape parameter ε in our algorithm's analysis. Therefore, we are able to provide precise user control over the evaluation error and precise asymptotic results on the complexity.

Unlike the FMM, our method is presently limited to smooth radial kernels. Its generalization to piecewise-smooth kernels like GDS is still a work-in-progress; see §7.

Our algorithm has important applications beyond RBFs: filtering and de-blurring of images [29, pp. 165–184]; force summation among particles/atoms with smooth potential of interactions ϕ (e.g., [11, §1–3]); evaluation and solution of *continuous* integral equations [20]

$$s(\mathbf{x}) = \int_{\Omega} \phi(\|\mathbf{x}-\mathbf{y}\|)\lambda(\mathbf{y})d\mathbf{y} \,, \tag{1.3}$$

discretized on the centers $\{\mathbf{y}_j\}_{j=0}^{n}$ and evaluation points $\{\mathbf{x}_i\}_{i=0}^{m}$; and so on.

The paper is organized as follows: In §2 we derive our fast evaluation algorithm in 1-D for any smooth kernel. We apply this general algorithm to specific smooth kernels from Table 1.1 and show numerical results in §3. The generalization of the algorithm to $d > 1$ dimensions is described in §4, followed by application of the d-dimensional algorithm to specific smooth kernels in §5. In §6, we present several numerical results in two and three dimensions. We discuss future research in §7.

2. 1-D fast evaluation algorithm.

Let ϕ be a smooth radial kernel (see Table 1.1, top section), and $\{y_j\}_{j=0}^{n}, \{x_i\}_{i=0}^{m} \subset \mathbb{R}$. Without loss of generality, we assume that the centers $\{y_j\}_{j=0}^{n}$ and the evaluation points $\{x_i\}_{i=0}^{m}$ lie in $[0,1]$ and are sorted so that $y_j < y_{j+1}$, $j = 0,1,\ldots,n-1$, and $x_i < x_{i+1}$, $i = 0,1,\ldots,m-1$. We denote by "level h" the collection of $\{y_j\}_{j=0}^{n}$ and $\{x_i\}_{i=0}^{m}$ and make no assumption on the densities of $\{y_j\}_{j=0}^{n}$ and $\{x_i\}_{i=0}^{m}$. Quantities defined at these centers and points are denoted by lowercase symbols

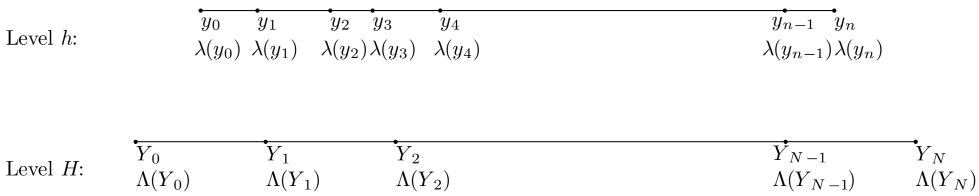

FIG. 2.1. *Illustration of the two-level approach for the case $p = 2$. $\{X_I\}_{I=0}^{M}$ are similarly defined over $\{x_i\}_{i=0}^{m}$. Level H contains $O(p)$ points outside the convex hull of level h to allow centered interpolation.*

(e.g., $\{\lambda(y_j)\}_{j=0}^{n}$, $\{s(x_i)\}_{i=0}^{m}$). The 1-D evaluation task is to compute

$$s(x_i) = \sum_{j=0}^{n} \lambda(y_j)\phi\left(|x_i - y_j|\right) \ , \quad i = 0, 1, \ldots, m \ . \tag{2.1}$$

We define an auxiliary "level H" consisting of two uniform grids $\{Y_J\}_{J=0}^{N}$ and $\{X_I\}_{I=0}^{M}$ with spacing H each. These grids cover $\{y_j\}_{j=0}^{n}$ and $\{x_i\}_{i=0}^{m}$, respectively, so that a discrete function defined at level H can be approximated at any y_j using *centered* pth-order interpolation, for some even $p \in \mathbb{N}$. Specifically, we choose

$$\begin{aligned} Y_J &= y_0 - \frac{(p-1)H}{2} + JH, \quad J = 0, 1, \ldots, N, \quad N = \left\lfloor \frac{y_n - y_0}{H} - 0.5 \right\rfloor + p, \\ X_I &= x_0 - \frac{(p-1)H}{2} + IH, \quad I = 0, 1, \ldots, M, \quad M = \left\lfloor \frac{x_m - x_0}{H} - 0.5 \right\rfloor + p. \end{aligned}$$

Quantities defined at level H are denoted by uppercase symbols (e.g., $\{\Lambda(Y_J)\}_{J=0}^{N}$, $\{S(X_I)\}_{I=0}^{M}$); see Fig. 2.1. Level H is coarser than, and at most comparable with level h; H, p are determined by the shape parameter ε of ϕ and the target accuracy δ in $\{s(x_i)\}_{i=0}^{m}$ (see §2.1, §3). The evaluation algorithm replaces the expensive summation (2.1) at level h by a less expensive summation at level H by utilizing the spatial smoothness of ϕ. First, $\phi(|x_i - y|)$ is a smooth function of y, for every fixed x_i. Therefore its value at $y = y_j$ can be approximated by a centered pth-order interpolation from its values at neighboring Y_J's. Namely,

$$\phi\left(|x_i - y_j|\right) = \sum_{J \in \sigma_j} \omega_{jJ}\phi\left(|x_i - Y_J|\right) \ + O(\delta_{\mathcal{I}}), \quad j = 0, 1, \ldots, n \ , \tag{2.2}$$

where $\sigma_j := \{J : |Y_J - y_j| < pH/2\}$, ω_{jJ} are the centered pth-order interpolation weights from the coarse centers Y_J to y_j, and $\delta_{\mathcal{I}}$ is the interpolation error, which we bound in §2.1 and §3. Substituting the approximation (2.2) into (2.1) and interchanging the order of summation, we obtain

$$\begin{aligned} s(x_i) &= \sum_{j=0}^{n} \left[\sum_{J \in \sigma_j} \omega_{jJ}\phi\left(|x_i - Y_J|\right) \ + O(\delta_{\mathcal{I}}) \right] \lambda(y_j) \\ &= \sum_{J=0}^{N} \phi\left(|x_i - Y_J|\right) \sum_{j : J \in \sigma_j} \omega_{jJ}\lambda(y_j) \ + O(n\|\boldsymbol{\lambda}\|_\infty \delta_{\mathcal{I}}) \\ &= \sum_{J=0}^{N} \Lambda(Y_J)\phi\left(|x_i - Y_J|\right) \ + O(n\|\boldsymbol{\lambda}\|_\infty \delta_{\mathcal{I}}), \quad i = 0, 1, \ldots, m \ , \end{aligned} \tag{2.3}$$

ETNA
Kent State University
etna@mcs.kent.edu

where

$$\Lambda(Y_J) := \sum_{j:J\in\sigma_j} \omega_{jJ}\lambda(y_j),\ J = 0,1,\ldots,N, \tag{2.4}$$

which is called *anterpolation* [11] or aggregation of $\{\lambda(y_j)\}_{j=0}^n$ to level H. The upper bound for the total interpolation error in $s(x_i)$ is $O(n\|\boldsymbol{\lambda}\|_\infty\delta_{\mathcal{I}})$, where $\|\boldsymbol{\lambda}\|_\infty$ is the maximum norm of $\{\lambda(y_j)\}_{j=0}^n$. In fact, if we assume that local errors accumulate randomly, the total error will only be $O(\sqrt{n}\|\boldsymbol{\lambda}\|_\infty\delta_{\mathcal{I}})$.

Second, we similarly use the smoothness of $\phi(|x - Y_J|)$ as a function of x for a fixed Y_J to approximate its value at $x = x_i$ by a centered pth-order interpolation from its values neighboring X_I's. Namely,

$$\phi\left(|x_i - Y_J|\right) = \sum_{I\in\overline{\sigma}_i} \overline{\omega}_{iI}\phi\left(|X_I - Y_J|\right)\ + O(\delta_{\mathcal{I}}),\ \ i = 0,1,\ldots,m\ . \tag{2.5}$$

where $\overline{\sigma}_i := \{I : |X_I - x_i| < pH/2\}$, and $\overline{\omega}_{iI}$ are the centered pth-order interpolation weights from the coarse evaluation points X_I to x_i. Substituting (2.5) into (2.3) gives

$$\begin{aligned} s(x_i) &= \sum_{J=0}^{N} \Lambda(Y_J)\left[\sum_{I\in\overline{\sigma}_i} \overline{\omega}_{iI}\phi\left(|X_I - Y_J|\right)\ + O(\delta_{\mathcal{I}})\right] + O(n\|\boldsymbol{\lambda}\|_\infty\delta_{\mathcal{I}}) \\ &= \sum_{I\in\overline{\sigma}_i} \overline{\omega}_{iI} \sum_{J=0}^{N} \Lambda(Y_J)\phi\left(|X_I - Y_J|\right)\ + O(n\|\boldsymbol{\lambda}\|_\infty\delta_{\mathcal{I}}) \\ &= \sum_{I\in\overline{\sigma}_i} \overline{\omega}_{iI} S(X_I)\ + O(n\|\boldsymbol{\lambda}\|_\infty\delta_{\mathcal{I}}),\ \ i = 0,1,\ldots,m\ , \end{aligned} \tag{2.6}$$

where

$$S(X_I) := \sum_{J=0}^{N} \Lambda(Y_J)\phi\left(|X_I - Y_J|\right),\ I = 0,1,\ldots,M\ . \tag{2.7}$$

For fast-decaying ϕ (e.g. GA),(2.7) can be truncated to a neighborhood of X_I and replaced by

$$S(X_I) = \sum_{J:|Y_J - X_I|<cH} \Lambda(Y_J)\phi\left(|X_I - Y_J|\right) + O(n\|\boldsymbol{\lambda}\|_\infty\delta_{\mathcal{T}}),\ \ I = 0,1,\ldots,M\ , \tag{2.8}$$

where $c \in \mathbb{N}$ and the truncation error $\delta_{\mathcal{T}}$ depends on ϕ and c. If ϕ does not decay fast (or at all) as $r \to \infty$ (e.g. IMQ and MQ), we resort to $c = N$ (i.e. no truncation).

Thus, the original evaluation task (2.1) is replaced by the less expensive, analogous evaluation task (2.8) at level H. Assuming (2.8) is directly summed, we can reconstruct $\{s(x_i)\}_{i=0}^m$ by *interpolating* $\{S(X_I)\}_{I=0}^M$ back to level h, namely, computing (2.6). Summarizing, our fast evaluation task consists of the following steps:

1. *Anterpolation:* for every $j = 0,1,\ldots,n$, compute the anterpolation weights $\{\omega_{jJ}\}_{J\in\sigma_j}$. Compute the coarse expansion coefficients $\{\Lambda(Y_J)\}_{J=0}^N$ using (2.4).
2. *Coarse Level Summation:* evaluate $S(X_I),\ I = 0,1,\ldots,M$ using (2.8).
3. *Interpolation:* for every $i = 0,1,\ldots,m$, compute the interpolation weights $\{\overline{\omega}_{iI}\}_{I\in\overline{\sigma}_i}$. Then interpolate $\{S(X_I)\}_{I=0}^M$ to $\{s(x_i)\}_{i=0}^m$ using (2.6).

2.1. Complexity and accuracy. Step 1 consists of two parts. Computing the weights $\{\omega_{jJ}\}_J$ requires $O(pn)$ operations (see Appendix A). Then (2.4) is executed in $O(pn)$ operations (see §2.2). The truncated coarse sum in step 2 requires $O(cM)$ operations. In some cases, this cost can be cut using the Fast Fourier Transform (FFT) as discussed below. Step 3 consists of computing $\{\overline{\omega}_{iI}\}_I$, which costs $O(pm)$, and interpolating the level H RBF expansion to level h for a cost of $O(pm)$. The algorithm's complexity is thus

$$W \sim (n+m)p + \frac{c}{H}. \tag{2.9}$$

Because the coarse grid interpolations are centered and p is even, the error $\delta_{\mathcal{I}}$ can be bounded by [42, p. 32]

$$\delta_{\mathcal{I}} = \left| \phi\left(|x_i - y_j|\right) - \sum_{J\in\sigma_j} \omega_{jJ}\phi\left(|x_i - Y_J|\right) \right| \le \frac{H^p\left[\Gamma\left(\frac{p+1}{2}\right)\right]^2}{p!\pi} \left|\phi^{(p)}(x_i-\xi)\right|,$$

where ξ is in the convex hull of $\{Y_J\}_{J\in\sigma_j}$, $j=0,1,\dots,n$. For infinitely smooth ϕ we obtain the uniform bound

$$\delta_{\mathcal{I}} \le \frac{H^p\left[\Gamma\left(\frac{p+1}{2}\right)\right]^2}{p!\pi} \left\|\phi^{(p)}\right\|_\infty .$$

The truncation error $\delta_{\mathcal{T}}$ in $S(X_I)$ due to (2.8) is bounded by the "tail" of ϕ. For every I,

$$\delta_{\mathcal{T}} \le \int_{-\infty}^{X_I - cH} |\phi(|Y-X_I|)|dY + \int_{X_I+cH}^{\infty} |\phi(|Y-X_I|)|dY = 2\int_{cH}^{\infty} |\phi(r)|dr.$$

Let $\mathbf{s}$ contain the values from directly summing (2.1) and $\hat{\mathbf{s}}$ the contain the values from our fast evaluation for $i=0,1,\dots,m$. We define the evaluation accuracy as the relative error norm

$$E := \frac{\|\mathbf{s}-\hat{\mathbf{s}}\|_\infty}{\|\mathbf{s}\|_\infty}. \tag{2.10}$$

Using the bounds on $\delta_{\mathcal{I}}$ and $\delta_{\mathcal{T}}$, we obtain

$$E \sim \kappa \left(\frac{H^p\left[\Gamma\left(\frac{p+1}{2}\right)\right]^2}{p!\pi} \left\|\phi^{(p)}\right\|_\infty + 2\int_{cH}^{\infty} \|\phi(r)\|dr \right), \tag{2.11}$$

where $\kappa := n\|\boldsymbol{\lambda}\|_\infty/\|\mathbf{s}\|_\infty$ is a measure of the condition number of the direct evaluation problem (2.1) (see the paragraph below). The algorithm's efficiency is determined by choosing the parameters H, p, c to minimize W for a bounded accuracy $E \le \delta$ (or minimize E subject to bounded W). An exact optimization is of course not required. The algorithm's efficiency depends only on ϕ, not on the specific locations $\{y_j\}_{j=0}^n, \{x_i\}_{i=0}^m$ or the values $\{\lambda(y_j)\}_{j=0}^n$. In §3 we show for a few specific RBFs that by correctly choosing the parameters (normally $p \sim c \sim \ln(1/\delta)$), the algorithm scales linearly with $n+m$ with constant $\sim \ln(1/\delta)$.

We conclude this section with a note on the condition number κ of (2.1). From [32], we know that if (2.1) is summed directly with precision u, then relative errors in $\mathbf{s}$ of size κu would be introduced. For example, if $u = 10^{-16}$, $\|\boldsymbol{\lambda}\|_\infty = O(10^8)$, and $\{\lambda(y_j)\}_{j=0}^n$ have alternating signs so that $\|\mathbf{s}\|_\infty = O(1)$ (as is often the case in RBF approximations), then $\kappa u = O(10^8 u) = O(10^{-8})$. This error analysis also holds true for *any* indirect summation method of (2.1) (e.g. FMM, FGT, or the current approach). Similar to these other fast evalution methods, we hereafter assume that the condition number κ is not too large, otherwise $\delta_{\mathcal{I}}$, $\delta_{\mathcal{T}}$ should be much smaller than δ to achieve $E \le \delta$.

2.2. Fast update; parallelization. *Interpolation.* Note that each $s(x_i)$, $i = 0, 1, \ldots, m$, in step 3 can be independently interpolated from $\{S(X_I)\}_{I=0}^M$. Hence, evaluating at a new point x costs $O(p) = O(\ln(1/\delta))$ operations, without repeating or updating steps 1 and 2. Most likely, new evaluation points lie in the convex hull of $\{Y_J\}_{J=0}^N$ and thus of $\{X_I\}_{I=0}^M$. However, if x cannot be centrally interpolated from existing $\{X_I\}_{I=0}^M$, we append the coarse grid with at most p new X_I's near x. The coarse summation (2.7) is then performed for the new X_I's and $s(x_i)$ is centrally interpolated from these $S(X_I)$. This update requires $O(pN)$ operations, which for some cases may be $O(\sqrt{n}\ln(1/\delta))$ (see §3); further evaluations inside the convex hull of the extended coarse grid cost only $O(\ln(1/\delta))$ per new evaluation point.

Anterpolation. Adjointly to interpolation, we implement anterpolation in our code as follows. First, all $\{\Lambda(Y_J)\}_{J=0}^N$ are set to zero. For each $j = 0, 1, \ldots, n$ we increment the coarse expansion coefficients which include $\lambda(y_j)$ in their sum (2.4); namely,

$$\Lambda(Y_J) \longleftarrow \Lambda(Y_J) + \omega_{jJ}\lambda(y_j), \quad \forall J \in \sigma_j. \tag{2.12}$$

This computation may be interpreted as distributing $\lambda(y_j)$ between several neighboring coarse centers, as illustrated in Fig. 2.2. A new center y can now be accommodated by incrementing

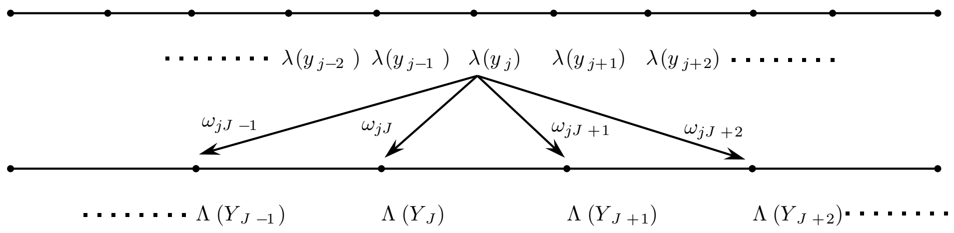

FIG. 2.2. *Anterpolation can be interpreted as distributing each $\lambda(y_j)$ between several neighboring coarse centers Y_J. The Figure depicts an example for $p = 4$.*

few $\Lambda(Y_J)$'s neighboring y by (2.12) ($O(p)$ operations), updating the coarse sums (2.7) for all X_I's ($O(pN)$ operations), and interpolating ($O(pm)$ operations), a total of $O(p(m+N))$ operations. Note that if we want to update the interpolant $s(x_i)$ only for some x_i near y, we need update only $O(p)$ relevant $S(X_I)$ with X_I near x_i, hence the cost of this update is only $O(p^2)$. For y outside the convex hull of $\{Y_J\}_{J=0}^N$, use a grid extension as in the previous paragraph. This complexity analysis also applies to *removing* a center y_j from the RBF interpolant.

Parallelization. The fast evaluation algorithm readily lends itself to distributed architectures. Anterpolation of each $\lambda(y_j)$ to the coarse lattice can be done independently of the others, however, multiple j's may require conflicting access to the same coarse data (all y_j's with $J \in \sigma_j$ would like to add their contributions to $\Lambda(Y_J)$). A domain decomposition approach in which each processor is assigned a contiguous sub-domain of the centers $\{y_j\}_{j=0}^n$ to work on, will resolve such conflicts. Similar ideas can be applied to the coarse level summation and interpolation steps.

2.3. Fast coarse level summation. In cases where the evaluation points $\{x_i\}_{i=0}^m$ are not far outside the convex hull of the centers $\{y_j\}_{j=0}^n$, and vice versa, the coarse evaluation points can be set equal to the coarse centers, viz. $M = N$, $X_J = Y_J$, $J = 0, 1, \ldots, N$. The coarse level summation (2.7) then amounts to a matrix vector product similar to (1.2), where Φ is now an $N \times N$ symmetric Toeplitz matrix. Thus, (2.7) can be computed in $O(N \ln N)$ operations using the FFT [28, pp. 201–202]. This approach is especially attractive for reducing the $O(N^2)$ complexity of a non-truncated coarse level summation (i.e $c = N$) for a large enough N. The complexity of the algorithm thus becomes

$$W \sim (n+m)p + \frac{1}{H} \min\left\{c, \ln\frac{1}{H}\right\},$$

which scales linearly with n and m.

3. Examples of 1-D evaluation.

3.1. GA kernel. Let $\phi(r) = e^{-(\varepsilon r)^2}$. Then

$$\left\|\phi^{(p)}\right\|_\infty = \varepsilon^p \frac{p!}{(p/2)!},$$

(see (B.2) of Appendix B with $d = 1$). In addition, ϕ exponentially decays at $r \to \infty$, and the "tail" is bounded by [1, p. 298]

$$\int_{cH}^{\infty} e^{-\varepsilon^2 r^2} dr \le \frac{e^{-(cH\varepsilon)^2}}{\varepsilon}. \tag{3.1}$$

Using (3.1) and Stirling's asymptotic formula [1, p. 257]

$$\Gamma(az+b) \sim \sqrt{2\pi} e^{-az} (az)^{az+b-1/2} \tag{3.2}$$

in (2.11), we obtain the accuracy estimate (keeping only the main terms)

$$E \sim \left(\frac{2}{\sqrt{\pi p}}\right)\left(\frac{H\varepsilon\sqrt{p}}{\sqrt{2e}}\right)^p + 2\frac{e^{-(cH\varepsilon)^2}}{\varepsilon} \lesssim \left(\frac{H\varepsilon\sqrt{p}}{\sqrt{2e}}\right)^p + \frac{e^{-(cH\varepsilon)^2}}{\varepsilon}.$$

The first term in E can be bounded by δ only if we require $H\varepsilon\sqrt{p} < b\sqrt{2e}$ for some $0 < b < 1$, and $p = O(\ln(1/\delta))$. Thus,

$$H \sim \frac{1}{\varepsilon\sqrt{p}}, \tag{3.3}$$

$$p \sim \ln\frac{1}{\delta}. \tag{3.4}$$

In practice, p is rounded to the next even integer. The second term is bounded by $O(\delta)$ if

$$\ln\frac{1}{\varepsilon} - (cH\varepsilon)^2 \lesssim \ln\delta \quad\Longleftrightarrow\quad \frac{1}{H\varepsilon}\left(\ln\frac{1}{\varepsilon\delta}\right)^{\frac{1}{2}} \lesssim c,$$

provided $\varepsilon\delta < 1$. W is minimized if and only if c is, hence we choose

$$c \sim \left(\ln\frac{1}{\varepsilon\delta}\ln\frac{1}{\delta}\right)^{\frac{1}{2}}. \tag{3.5}$$

By selecting (3.3)–(3.5), we can evaluate (2.1) for the GA RBF in

$$W \sim O\left(\left(\ln\frac{1}{\delta}\right)(n+m) + \left(\ln\frac{1}{\varepsilon\delta}\ln\frac{1}{\delta}\right)^{\frac{1}{2}}\varepsilon\right) \tag{3.6}$$

operations. The complexity scales linearly with n and m for all $\varepsilon \lesssim n$. If $\varepsilon \gg n$, the original summation (2.1) can be truncated similarly to (2.8) and directly evaluated in $O(n+m)$ operations. Hence, the evaluation task (2.1) with the GA kernel can be carried out in $O(n+m)$ operations, for all $\varepsilon > 0$.

ETNA
Kent State University
etna@mcs.kent.edu

3.1.1. Numerical experiments. We numerically verify the accuracy and work estimates for our multilevel method derived above. In all experiments we set $\varepsilon = \sqrt{n}/4$ and $m = 2n$. Similar results are obtained with other n, m, ε. We do not employ the FFT summation technique of §2.3.

To enforce $\delta_{\mathcal{I}} + \delta_{\mathcal{T}} \leq \delta$, we set the algorithm's parameters so that $\delta_{\mathcal{I}} = \delta_{\mathcal{T}} = \delta/2$. The work (3.6) is then minimized when H, p, c are chosen as follows:

$$\overline{p} = \frac{\ln \frac{2}{\delta}}{\ln \frac{1}{b}}$$

$$H = \frac{b}{\varepsilon}\left(\frac{2e}{\overline{p}}\right)^{\frac{1}{2}}, \tag{3.7}$$

$$p = \lceil\lceil \overline{p} \rceil\rceil, \tag{3.8}$$

$$c = \begin{cases} \left\lceil \frac{g}{H\varepsilon}\left(\ln \frac{4}{\varepsilon\delta}\right)^{\frac{1}{2}} \right\rceil & \varepsilon < \frac{4}{\delta}, \\ 0 & \varepsilon \geq \frac{4}{\delta}, \end{cases}$$

where $\lceil\cdot\rceil$ denotes rounding to the next integer and $\lceil\lceil\cdot\rceil\rceil$ indicates rounding the argument to the next even integer. We use $b = 1/4$ and $g = 1.1$; a more precise analysis (also alleviating the need for the deriviation (3.3)–(3.5)) could optimize b, g by preparing a numerical table of the relative error $E = E(b, g)$ (measured for several different n's versus a direct evaluation of (2.1), and averaged over several random $\{\lambda(y_j)\}_{j=0}^n$), and using b, g to minimize W under $E \leq \delta$. However, this hardly seems profitable, as good results are obtained for our rough estimates for b, g.

First, we verify that with this choice of parameters the relative error E (2.10) is indeed $O(\delta)$. Table 3.1 shows the E for various values of n and δ. Each entry in the table is the average of ten different experiments, where $\{y_j\}_{j=0}^n$ and $\{x_i\}_{i=0}^m$ were randomly selected in $[0, 1]$, and $\{\lambda(y_j)\}_{j=0}^n$ randomly chosen in $[-1, 1]$ in each experiment. Clearly, E is below δ in all cases.

TABLE 3.1
Relative error E of the multilevel evaluation method for the GA, versus n and δ ($m = 2n$).

n	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$
100	$5.34 \cdot 10^{-3}$	$1.34 \cdot 10^{-5}$	$7.12 \cdot 10^{-8}$	$1.84 \cdot 10^{-9}$	$5.90 \cdot 10^{-12}$
200	$5.55 \cdot 10^{-3}$	$2.07 \cdot 10^{-5}$	$5.66 \cdot 10^{-8}$	$1.39 \cdot 10^{-9}$	$7.43 \cdot 10^{-12}$
400	$3.50 \cdot 10^{-3}$	$1.12 \cdot 10^{-5}$	$5.71 \cdot 10^{-8}$	$1.23 \cdot 10^{-9}$	$6.49 \cdot 10^{-12}$
800	$4.28 \cdot 10^{-3}$	$1.99 \cdot 10^{-5}$	$4.99 \cdot 10^{-8}$	$1.59 \cdot 10^{-9}$	$4.81 \cdot 10^{-12}$
1600	$3.15 \cdot 10^{-3}$	$1.95 \cdot 10^{-5}$	$9.21 \cdot 10^{-8}$	$1.44 \cdot 10^{-9}$	$7.84 \cdot 10^{-12}$

Second, we verify that the work W (3.6) linearly scales with m, n, and $\ln(1/\delta)$. Table 3.2 compares the number of operations required for our multilevel method for various values of n and δ, with a direct evaluation. Each evaluation of e^{-t} is counted as one operation. As expected, the results follow the work estimate (3.6), giving $W \approx \alpha \ln(1/\delta)(n + m)$, where $5.7 < \alpha < 7.2$. Note that the hidden constant (which is of course only roughly estimated here) is very small.

3.2. MQ kernel. Let $\phi(r) = (1 + (\varepsilon r)^2)^{\frac{1}{2}}$. This kernel grows as $r \to \infty$, hence we do not truncate the coarse level sum by choosing $c = N$ in (2.8). From (B.6) of Appendix B

TABLE 3.2
Work (number of floating-point operations) W required of the multilevel evaluation method for the GA, for various n and desired accuracies δ. The right column indicates the number of operations required for a direct evaluation. We average over centers and expansion coefficients as in Table 3.1.

n	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$	direct
100	10402	19911	29940	35282	45686	100000
200	20492	38816	57820	68177	87431	400000
400	40227	76181	112710	132222	169441	1600000
800	79572	149716	221165	258857	331206	6400000
1600	158042	296596	436435	510302	651801	25600000

with $\nu = 1, d = 1$,

$$\left\| \phi^{(p)} \right\|_\infty = \varepsilon^p \frac{2^p \Gamma\left(\frac{p+1}{2}\right) \Gamma\left(\frac{p-1}{2}\right)}{2\pi}.$$

Applying this to (2.11) and expanding with (3.2), we obtain the accuracy estimate

$$E \sim \frac{2}{p} \left(\frac{2}{\pi p}\right)^{\frac{1}{2}} \left(\frac{H\varepsilon p}{2e}\right)^p \lesssim \left(\frac{H\varepsilon p}{2e}\right)^p. \tag{3.9}$$

This can be bounded by δ only if we require $H\varepsilon p < 2eb$ for some $0 < b < 1$, and $p = O(\ln(1/\delta))$. Thus,

$$H \sim \frac{1}{\varepsilon p}$$
$$p \sim \ln \frac{1}{\delta}.$$

Again, p is rounded to the next even integer. The fast evaluation complexity for the MQ kernel is

$$W \sim O\left(\left(\ln \frac{1}{\delta}\right)(n+m) + \left(\ln \frac{1}{\delta}\right)^2 \varepsilon^2 \right)$$

operations. Hence, the algorithm scales linearly with n and m for all $\varepsilon \sim \sqrt{n}$ or smaller. For larger ε, W is dominated by the coarse level summation (2.7). As discussed in §2.3, we can reduce this computation to $O((\ln(1/\delta)\varepsilon)\ln(\ln(1/\delta)\varepsilon))$ using the FFT.

3.2.1. Numerical experiments. Similarly to §3.1.1, we numerically verify the accuracy and work estimates derived above. The same assumptions on the centers, evaluation points and expansion coefficients are made; $\varepsilon = \sqrt{n}/4$ and $m = 2n$. The FFT summation technique of §2.3 is not used, as $\varepsilon \sim \sqrt{n}$.

Here, $\delta_{\mathcal{T}} = 0$, thus we choose p, H to have $\delta_{\mathcal{I}} \sim \delta$. The optimal H, p that minimize (2.9) are

$$\overline{p} = \frac{\ln \frac{1}{\delta}}{\ln \frac{1}{b}} \tag{3.10}$$

$$H = \frac{2eb}{\varepsilon \overline{p}}, \tag{3.11}$$

$$p = \lceil\lceil \overline{p} \rceil\rceil, \tag{3.12}$$

ETNA
Kent State University
etna@mcs.kent.edu

We use the non-optimized value of $b = 1/4$.

We first verify that the relative error $E \sim O(\delta)$. Table 3.3 shows the results for E for various values of n and δ. Each entry in the table is the average of ten different random experiments as in Table 3.1. Note that E is much smaller than δ, especially as δ becomes smaller (i.e. as p grows). This is most likely because the last bound in (3.9) does not account for the $p\sqrt{p}$ term in the denominator. More precise error bounds and parameter studies will be explored in future research.

TABLE 3.3
Relative error E of the multilevel evaluation method for MQ kernel.

n	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$
100	$2.15 \cdot 10^{-3}$	$2.37 \cdot 10^{-6}$	$2.41 \cdot 10^{-8}$	$1.61 \cdot 10^{-10}$	$4.61 \cdot 10^{-13}$
200	$1.31 \cdot 10^{-3}$	$3.16 \cdot 10^{-6}$	$2.15 \cdot 10^{-8}$	$9.80 \cdot 10^{-11}$	$2.84 \cdot 10^{-13}$
400	$9.55 \cdot 10^{-4}$	$1.13 \cdot 10^{-6}$	$1.21 \cdot 10^{-8}$	$3.37 \cdot 10^{-11}$	$1.34 \cdot 10^{-13}$
800	$4.36 \cdot 10^{-4}$	$1.07 \cdot 10^{-6}$	$6.78 \cdot 10^{-9}$	$3.06 \cdot 10^{-11}$	$5.47 \cdot 10^{-14}$
1600	$5.13 \cdot 10^{-4}$	$6.64 \cdot 10^{-7}$	$4.11 \cdot 10^{-9}$	$1.54 \cdot 10^{-11}$	$1.14 \cdot 10^{-13}$

Second, we verify that the work W (3.6) scales linearly with m, n, and $\ln(1/\delta)$. Table 3.4 compares the number of operations required for our multilevel method for various values of n and δ. Each evaluation of $\sqrt{1+t}$ is counted as one operation. The method scales linearly, similar to Table 3.2. For this case, $W \approx \alpha \ln(1/\delta)(n+m)$, where $5.8 < \alpha < 7.2$.

TABLE 3.4
Measure of work W required of the multilevel evaluation method for the MQ, in terms of operation count. The right column indicates the number of operations required for a direct evaluation. The centers and $m = 2n$ evaluation points are randomly distributed in the $[0, 1]$.

n	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$	direct
100	10222	20146	26278	38002	51211	100000
200	20267	39371	51138	73487	96911	400000
400	40102	77566	100303	142727	187711	1600000
800	79827	154091	197678	280127	366986	6400000
1600	159142	305891	391303	554002	723706	25600000

3.3. IMQ kernel. The infinitely smooth kernel $\phi(r) = (1+(\varepsilon r)^2)^{-\frac{1}{2}}$ decays as $r \to \infty$, but not rapidly enough for a coarse level truncation in (2.8). Thus, we again set $c = N$. The following bound on $\phi^{(p)}(r)$ follows from (B.6) with $\nu = -1$, $d = 1$:

$$\left\| \phi^{(p)} \right\|_\infty = \varepsilon^p \frac{2^p \left(\Gamma\left(\frac{p+1}{2}\right)\right)^2}{2\pi}.$$

Using this in (2.11) and expanding with (3.2), we obtain the same accuracy estimate as (3.9), thus we use the same parameters as for MQ, (3.11)–(3.12). The numerical results are similar to those of §3.2, hence we do not further elaborate on them.

4. Fast evaluation algorithm in two and higher dimensions. We now consider the multilevel algorithm for smooth kernels in higher dimensions. For simplicity, we only describe the two-dimensional algorithm as the generalization to higher dimensions should be straightforward. We do, however, describe the complexity of the algorithm in terms of $d \geq 2$ dimensions.

ETNA
Kent State University
etna@mcs.kent.edu

Let $\{\mathbf{y}_j\}_{j=0}^n, \{\mathbf{x}_i\}_{i=0}^m \subset \mathbb{R}^2$ and $\mathbf{y}_j = (y_j^{(1)}, y_j^{(2)})$, $\mathbf{x}_j = (x_i^{(1)}, x_i^{(2)})$. Without loss of generality, we assume that the centers $\{\mathbf{y}_j\}_{j=0}^n$ and evaluation points $\{\mathbf{x}_i\}_{i=0}^m$ lie in $[0,1]^2$. We again make no assumption on the densities of the centers and evaluation points, but still label them as "level h", where h is some measure of cell/point distribution. Quantities defined at level h are again denoted by lowercase symbols. The task is now to compute

$$s(\mathbf{x}_i) = \sum_{j=0}^{n} \lambda(\mathbf{y}_j)\phi\left(\|\mathbf{x}_i - \mathbf{y}_j\|\right)\ , \quad i = 0,1,\ldots,m\ . \tag{4.1}$$

We define an auxiliary "level $\mathbf{H}$", $\mathbf{H} = (H, H)$, where each component consists of two uniform grids $\left\{Y_{J_k}^{(k)}\right\}_{J_k=0}^{N_k}$, $\left\{X_{I_k}^{(k)}\right\}_{I_k=0}^{M_k}$, $k = 1, 2$, each with spacing H. Furthermore, we introduce the notation

$$\{\mathbf{Y}_\mathbf{J}\}_{\mathbf{J}=0}^{\mathbf{N}} = \left\{Y_{J_1}^{(1)}\right\}_{J_1=0}^{N_1} \times \left\{Y_{J_2}^{(2)}\right\}_{J_2=0}^{N_2},\ \mathbf{J} = (J_1, J_2),\ \mathbf{N} = (N_1, N_2),\ \text{and}$$
$$\{\mathbf{X}_\mathbf{I}\}_{\mathbf{I}=0}^{\mathbf{M}} = \left\{X_{I_1}^{(1)}\right\}_{I_1=0}^{M_1} \times \left\{X_{I_2}^{(2)}\right\}_{I_2=0}^{M_2},\ \mathbf{I} = (I_1, I_2),\ \mathbf{M} = (M_1, M_2),$$

where $\times$ denotes the Cartesian (or direct) product. As an example, the level $\mathbf{H}$ center $(Y_{J_1}^{(1)}, Y_{J_2}^{(2)})$ corresponds to $\mathbf{Y}_{(J_1,J_2)}$; similar notation holds for the level $\mathbf{H}$ evaluation points. Note that this notation differs from the level h notation where the centers and evaluation points are simply a list of the 2-D points. The grids $\{\mathbf{Y}_\mathbf{J}\}_{\mathbf{J}=0}^{\mathbf{N}}$ and $\{\mathbf{X}_\mathbf{I}\}_{\mathbf{I}=0}^{\mathbf{M}}$ are selected so that they cover $\{\mathbf{y}_j\}_{j=0}^n$ and $\{\mathbf{x}_i\}_{i=0}^m$, respectively, with the additional condition that a discrete function at level $\mathbf{H}$ can be approximated at any $\mathbf{y}_j$ using *centered*, pth-order, tensor product interpolation, for $p \in 2\mathbb{N}$. Specifically, for $k = 1, 2$, we choose

$$Y_J^{(k)} = y_{\text{min}}^{(k)} - \frac{(p-1)H}{2} + JH, \quad J = 0,1,\ldots,N_k,$$
$$X_I^{(k)} = x_{\text{min}}^{(k)} - \frac{(p-1)H}{2} + IH, \quad I = 0,1,\ldots,M_k,$$

where

$$N_k = \left\lfloor \frac{y_{\text{max}}^{(k)} - y_{\text{min}}^{(k)}}{H} - 0.5 \right\rfloor + p, \quad y_{\text{min}}^{(k)} = \min_{0\le j\le n}\ y_j^{(k)}, \quad y_{\text{max}}^{(k)} = \max_{0\le j\le n}\ y_j^{(k)},$$
$$M_k = \left\lfloor \frac{x_{\text{max}}^{(k)} - x_{\text{min}}^{(k)}}{H} - 0.5 \right\rfloor + p, \quad x_{\text{min}}^{(k)} = \min_{0\le i\le m}\ x_i^{(k)}, \quad x_{\text{max}}^{(k)} = \max_{0\le i\le m}\ x_i^{(k)}.$$

As for the 1-D algorithm, level $\mathbf{H}$ is coarser than, and at most comparable with level h. The values of $\mathbf{H}$ and p are determined by ε and the target accuracy δ in evaluating (4.1) as explained in the following sections. Utilizing ϕ's spatial smoothness, we again replace the expensive summation (4.1) by a less expensive summation at level $\mathbf{H}$.

Because $\phi(\|\mathbf{x}_i - \mathbf{y}\|)$ is a smooth function of $y^{(1)}, y^{(2)}$ for every fixed $\mathbf{x}_i$, its value at $\mathbf{y} = \mathbf{y}_j$ can be approximated by a centered pth-order, tensor product interpolation in $y^{(1)}$ and $y^{(2)}$ from its values at neighboring $\mathbf{Y}_\mathbf{J}$'s. Namely,

$$\phi\left(\|\mathbf{x}_i - \mathbf{y}_j\|\right) = \sum_{J_2 \in \sigma_j^{(2)}} \omega_{jJ_2} \sum_{J_1 \in \sigma_j^{(1)}} \omega_{jJ_1} \phi\left(\left\|\mathbf{x}_i - \mathbf{Y}_{(J_1,J_2)}\right\|\right) + O(\delta_{\mathcal{I}}), \quad j = 0,1,\ldots,n\ , \tag{4.2}$$

ETNA
Kent State University
etna@mcs.kent.edu

where for $k = 1, 2$, $\sigma_j^{(k)} := \left\{ J_k : |Y_{J_k}^{(k)} - y_j^{(k)}| < pH/2 \right\}$, ω_{jJ_k} are the centered pth-order interpolation weights from the coarse centers $Y_{J_k}^{(k)}$ to $y_j^{(k)}$, and $\delta_{\mathcal{I}}$ is the interpolation error, which we bound in §4.1 and §5. The anterpolation of $\{\lambda(\mathbf{y}_j)\}_{j=0}^n$ to level $\mathbf{H}$ is obtained by substituting the approximation (4.2) into (4.1) and interchanging the order of summation:

(4.3)
$$s(\mathbf{x}_i) = \sum_{J_2=0}^{N_2} \sum_{J_1=0}^{N_1} \Lambda\left(\mathbf{Y}_{(J_1,J_2)}\right) \phi\left(\left\|\mathbf{x}_i - \mathbf{Y}_{(J_1,J_2)}\right\|\right) + O(n\|\boldsymbol{\lambda}\|_\infty \delta_{\mathcal{I}}), \quad i = 0, 1, \ldots, m\,,$$

where

(4.4)
$$\Lambda\left(\mathbf{Y}_{(J_1,J_2)}\right) := \sum_{j:J_2 \in \sigma_j^{(2)}} \omega_{jJ_2} \sum_{j:J_1 \in \sigma_j^{(2)}} \omega_{jJ_1} \lambda(\mathbf{y}_j), \; J_2 = 0, 1, \ldots, N_2, \; J_1 = 0, 1, \ldots, N_1.$$

We implement (4.4) similarly to (2.12): all Λ's are initialized to zero and each $\lambda(\mathbf{y}_j)$ is distributed among p^2 neighboring $\mathbf{Y}_\mathbf{J}$'s as depicted in Fig. 4.1.

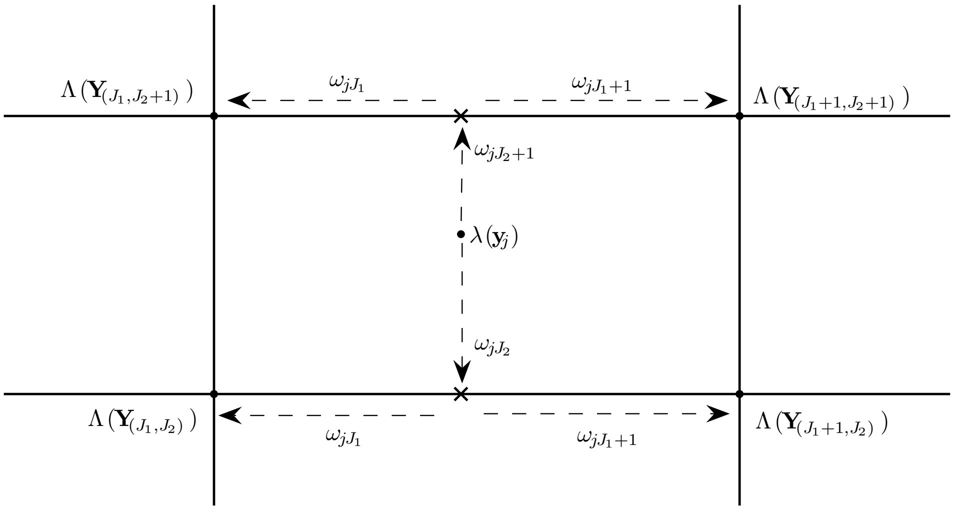

FIG. 4.1. *An example of anterpolation in 2D for $p = 2$.*

Similarly, using the smoothness of $\phi(\|\mathbf{x} - \mathbf{Y}_\mathbf{J}\|)$ as a function of $x^{(1)}, x^{(2)}$ for a fixed $\mathbf{Y}_\mathbf{J}$, we obtain

(4.5)
$$\phi\left(\|\mathbf{x}_i - \mathbf{Y}_\mathbf{J}\|\right) = \sum_{I_2 \in \overline{\sigma}_i^{(2)}} \varpi_{iI_2} \sum_{I_1 \in \overline{\sigma}_i^{(1)}} \varpi_{iI_1} \phi\left(\left\|\mathbf{X}_{(I_1,I_2)} - \mathbf{Y}_\mathbf{J}\right\|\right) + O(\delta_{\mathcal{I}}), \quad i = 0, 1, \ldots, m\,.$$

where for $k = 1, 2$, $\overline{\sigma}_i^{(k)} := \left\{ I_k : |X_{I_k}^{(k)} - x_i^{(k)}| < pH/2 \right\}$, ϖ_{iI_k} are the centered pth-order interpolation weights from the coarse evaluation point $X_{I_k}^{(k)}$ to $x_i^{(k)}$. Substituting (4.5) into (4.3) gives

$$s(\mathbf{x}_i) = \sum_{I_2 \in \overline{\sigma}_i^{(2)}} \varpi_{iI_2} \sum_{I_2 \in \overline{\sigma}_i^{(1)}} \varpi_{iI_1} S\left(\mathbf{X}_{(I_1,I_2)}\right) + O(n\|\boldsymbol{\lambda}\|_\infty \delta_{\mathcal{I}}), \quad i = 0, 1, \ldots, m\,, \tag{4.6}$$

ETNA
Kent State University
etna@mcs.kent.edu

where

$$S\left(\mathbf{X}_{(I_1,I_2)}\right) := \sum_{J_2=0}^{N_2}\sum_{J_1=0}^{N_1}\Lambda\left(\mathbf{Y}_{(J_1,J_2)}\right)\phi\left(\left\|\mathbf{X}_{(I_1,I_2)}-\mathbf{Y}_{(J_1,J_2)}\right\|\right),$$
$$I_1=0,1,\ldots,M_1,\ I_2=0,1,\ldots,M_2. \tag{4.7}$$

Note that summing all the terms as is indicated above may not be necessary because $\Lambda\left(\mathbf{Y}_{(J_1,J_2)}\right)$ could be zero at some coarse level centers. This would occur, for example, when the data fall on some smaller dimensional subset than $[0,1]^2$. We can also truncate (4.7) to a neighborhood of $\mathbf{X}_\mathbf{I}$ for fast-decaying ϕ (e.g. GA):

$$S\left(\mathbf{X}_{(I_1,I_2)}\right) := \sum_{J_2=0}^{N_2}\sum_{J_1:\|\mathbf{X}_\mathbf{I}-\mathbf{Y}_\mathbf{J}\|<cH}\Lambda\left(\mathbf{Y}_{(J_1,J_2)}\right)\phi\left(\left\|\mathbf{X}_{(I_1,I_2)}-\mathbf{Y}_{(J_1,J_2)}\right\|\right)$$
$$+O(n\|\boldsymbol{\lambda}\|_\infty\delta_\mathcal{T}),\ I_1=0,1,\ldots,M_1,\ I_2=0,1,\ldots,M_2, \tag{4.8}$$

where $c\in\mathbb{N}$, and the truncation error $\delta_\mathcal{T}$ depends on ϕ and c (and the dimension d). If ϕ does not decay fast (or at all) as $r\to\infty$ (e.g. MQ and IMQ), we resort to $c=\max\{N_1,N_2\}$ (i.e. no truncation).

Our 2-D fast multilevel evaluation task of (4.1) thus consists of the following steps:

1. *Anterpolation:* for every $j=0,1,\ldots,n$, compute the anterpolation weights $\{\omega_{jJ_k}\}_{J_k\in\sigma_j^{(k)}}$, $k=1,2$. Then compute the coarse expansion coefficients $\{\Lambda(\mathbf{Y}_\mathbf{J})\}_{\mathbf{J}=0}^{\mathbf{N}}$ using (4.4).
2. *Coarse Level Summation:* evaluate $S\left(\mathbf{X}_{(I_1,I_2)}\right)$, $I_1=0,1,\ldots,M_1$, $I_2=0,1,\ldots,M_2$ using (4.8).
3. *Interpolation:* for every $i=0,1,\ldots,m$, compute the interpolation weights $\{\varpi_{iI_k}\}_{I_k\in\bar\sigma_i^{(k)}}$, $k=1,2$. Then interpolate $\{S(\mathbf{X}_I)\}_{\mathbf{I}=0}^{\mathbf{M}}$ to $\{s(\mathbf{x}_i)\}_{i=0}^m$ using (4.6).

The generalization to $d>2$ dimensions follows by

- defining a "level $\mathbf{H}$", $\mathbf{H}=(H,H,\ldots,H)$ (i.e. d components) consisting of the d-dimensional Cartesian product of the uniform center-grid $\left\{Y_{J_k}^{(k)}\right\}_{J_k=0}^{N_k}$, and evaluation-grid $\left\{X_{I_k}^{(k)}\right\}_{I_k=0}^{M_k}$, $k=1,2,\ldots,d$;
- using centered, pth order, tensor product interpolation between the level h and level $\mathbf{H}$ grids;
- generalizing equations (4.2)–(4.8) to d dimensions.

4.1. Complexity and accuracy. We describe the complexity and accuracy of the above algorithm for a general dimension d and for the case when $\{\mathbf{x}_i\}_{i=0}^m$ and $\{\mathbf{y}_j\}_{j=0}^n$ are uniformly distributed in $[0,1]^d$. We may thus make the simplifying assumption that $N_k=N$ and $M_k=M$, $k=1,2,\ldots,d$, for appropriate values of M and N. The accuracy of the algorithm does not change for non-uniformly distributed points, but the work may be smaller for this case (e.g. if the centers are located on a lower dimensional space of $[0,1]^d$, most terms in (4.7) are zero and need not be summed). Uniform dense points provide the worst case estimate of the work.

Step 1 above consists of two parts. Computing the weights $\{\omega_{jJ_k}\}_{J_k}$, $k=1,2,\ldots,d$, requires $O(pdn)$ operations (see Appendix A). Then (4.4) is executed in $O(p^dn)$ operations. The truncated coarse sum in step 2 requires $O((cM)^d)$ operations. As discussed below, this cost can again be cut using the FFT. Step 3 consists of computing $\{\varpi_{iI_k}\}_{I_k}$, $k=1,2,\ldots,d$, which costs $O(pdm)$, and interpolating the level $\mathbf{H}$ RBF expansion to level h for a cost of

ETNA
Kent State University
etna@mcs.kent.edu

$O(p^d m)$. The algorithm's complexity is thus

$$W \sim (n+m)p^d + \left(\frac{c}{H}\right)^d . \tag{4.9}$$

Let $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_d) \in \mathbb{R}^d$. Then for *even-order*, p, tensor-product interpolation in d-dimensions, the error [42, §19] depends on bounds for the $2^d - 1$ terms

$$\begin{array}{ll}
\dfrac{\partial^p \phi(\|\boldsymbol{\xi}\|)}{\partial \xi_{j_1}^p} , & 1 \le j_1 \le d , \\
\dfrac{\partial^{2p} \phi(\|\boldsymbol{\xi}\|)}{\partial \xi_{j_1}^p \partial \xi_{j_2}^p} , & 1 \le j_1 < j_2 \le d , \\
\dfrac{\partial^{3p} \phi(\|\boldsymbol{\xi}\|)}{\partial \xi_{j_1}^p \partial \xi_{j_2}^p \partial \xi_{j_3}^p} , & 1 \le j_1 < j_2 < j_3 \le d , \\
\vdots & \vdots \\
\dfrac{\partial^{dp} \phi(\|\boldsymbol{\xi}\|)}{\partial \xi_{j_1}^p \partial \xi_{j_2}^p \cdots \partial \xi_{j_d}^p} , & 1 \le j_1 < j_2 < \cdots < j_d \le d .
\end{array}$$

Because ϕ is radially symmetric and infinitely smooth, it is sufficient to express the bounds in terms of

$$\frac{\partial^{kp} \phi(\|\boldsymbol{\xi}\|)}{\partial \xi_1^p \partial \xi_2^p \cdots \partial \xi_k^p}, \quad 1 \le k \le d . \tag{4.10}$$

The coarse grid interpolations are centered, thus the error $\delta_{\mathcal{I}}$ can be uniformly bounded by [42, p. 32,217]

$$\delta_{\mathcal{I}} \le \sum_{k=1}^{d} \binom{d}{k} \left[\frac{H^p \left[\Gamma\left(\frac{p+1}{2}\right)\right]^2}{p! \pi} \right]^k \left\| \frac{\partial^{kp} \phi(\|\boldsymbol{\xi}\|)}{\partial \xi_1^p \partial \xi_2^p \cdots \partial \xi_k^p} \right\|_\infty .$$

In §5 we provide more explicit bounds for the GA, MQ, and IMQ radial kernels.

The truncation error $\delta_{\mathcal{T}}$ in $S(\mathbf{X_I})$ due to (2.8) is again bounded by the "tail" of ϕ in d dimensions. For every $\mathbf{I} = (I_1, I_2, \ldots, I_d)$,

$$\delta_{\mathcal{T}} \le \frac{2\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)} \int_{cH}^{\infty} r^{d-1} \left|\phi(r)\right| dr, \tag{4.11}$$

where the constant in front of the integral is the surface area of the d-sphere.

We define the evaluation accuracy by the relative error norm (2.10). Using the bounds on $\delta_{\mathcal{I}}$ and $\delta_{\mathcal{T}}$, we obtain (assuming the condition number $\kappa \sim 1$)

$$E \sim \sum_{k=1}^{d} \binom{d}{k} \left[\frac{H^p \left[\Gamma\left(\frac{p+1}{2}\right)\right]^2}{p! \pi} \right]^k \left\| \frac{\partial^{kp} \phi(\|\boldsymbol{\xi}\|)}{\partial \xi_1^p \partial \xi_2^p \cdots \partial \xi_k^p} \right\|_\infty + \frac{2\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)} \int_{cH}^{\infty} r^{d-1} \left|\phi(r)\right| dr .$$

The algorithm's efficiency is again determined by choosing the parameters H, p, c to minimize W for a bounded accuracy $E \le \delta$ (or minimize E subject to a bounded W). The algorithm's efficiency depends on ϕ and d. In §5 we show for a few specific RBFs that by correctly choosing the parameters, the algorithm scales like $O((n+m)\ln(1/\delta)^d)$.

Fast updates and parallelization can be efficiently organized similarly to the 1-D case; see §2.2.

ETNA
Kent State University
etna@mcs.kent.edu

4.2. Fast coarse level summation. We again assume that $\{\mathbf{x}_i\}_{i=0}^m$ and $\{\mathbf{y}_j\}_{j=0}^n$ are uniformly distributed in $[0,1]^d$ (this is again not necessary for our algorithm, but makes the complexity analysis easier). Similarly to the 1-D algorithm, when the evaluation points $\{\mathbf{x}_i\}_{i=0}^m$ are not far outside the convex hull of the centers $\{\mathbf{y}_j\}_{j=0}^n$ and vice versa, the coarse evaluation points can be set equal to the coarse centers, viz. $\mathbf{M} = \mathbf{N}$, and $N_k = N$, $k = 1, 2, \ldots, d$. The coarse level summation (2.7) then again amounts to a matrix vector product similar to (1.2). In this case, however, Φ is a symmetric *d-level recursive block Toeplitz* matrix [35]. Using the algorithm of Lee [35, 36], we can multiply this matrix vector product in $O(N^d \ln N)$ operations. When no coarse level truncation is performed (i.e $c = N$) this greatly reduces the $O(N^{2d})$ complexity of (2.7). With this additional trick, the complexity of the algorithm thus becomes

$$W \sim (n+m)p^d + \frac{1}{H^d}\min\left\{c^d, \ln\frac{1}{H}\right\},$$

which scales linearly with n and m.

5. Applications to specific kernels in $d \geq 2$ dimensions. We discuss the accuracy and complexity of the d-dimensional algorithm applied to the GA, MQ, and IMQ kernels and postpone the numerical experiments until §6.

5.1. GA kernel. Let $\phi(r) = e^{-(\varepsilon r)^2}$. By changing variables to $t = (\varepsilon r)^2$, the following bound on the truncation error (4.11) is obtained:

$$\delta_{\mathcal{T}} \leq \frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)\varepsilon^d}\int_{(cH\varepsilon)^2}^{\infty} t^{\frac{d}{2}-1}e^{-t}dt \;=\; \left(\frac{\sqrt{\pi}}{\varepsilon}\right)^d \frac{\Gamma\left(\frac{d}{2},(cH\varepsilon)^2\right)}{\Gamma\left(\frac{d}{2}\right)}.$$

For $d \geq 2$, the incomplete gamma function can be bounded as follows [26]:

$$\frac{\Gamma\left(\frac{d}{2},(cH\varepsilon)^2\right)}{\Gamma\left(\frac{d}{2}\right)} < 1 - \left(1 - e^{-(cH\varepsilon)^2(\Gamma(d/2+1))^{-2/d}}\right)^{d/2}.$$

The bounds on (4.10) for the GA kernel are derived in Appendix B and are given by (B.2). Combining these bounds with the truncation error bounds and using Stirling's asymptotic formula (3.2), we obtain the accuracy estimate

$$E \sim \sum_{k=1}^{d}\binom{d}{k}\left(\frac{2}{\sqrt{\pi p}}\right)^k\left(\frac{H\varepsilon\sqrt{p}}{\sqrt{2e}}\right)^{kp} + \left(\frac{\sqrt{\pi}}{\varepsilon}\right)^d\left[1 - \left(1 - e^{-(cH\varepsilon)^2(\Gamma(d/2+1))^{-2/d}}\right)^{d/2}\right]. \tag{5.1}$$

Requiring $H\varepsilon\sqrt{p} < b\sqrt{2e}$ for some $0 < b < 1$, the first term becomes

$$\sum_{k=1}^{d}\binom{d}{k}\left(\frac{2}{\sqrt{\pi p}}\right)^k\left(\frac{H\varepsilon\sqrt{p}}{\sqrt{2e}}\right)^{kp} \sim b^p\,.$$

We can thus bound the first term in (5.1) by $O(\delta)$ by choosing $p = O(\ln(1/\delta))$. Therefore, H and p asymptotically behave like (3.3) and (3.5), respectively.

The second term in (5.1) is bounded by $O(\delta)$ if

$$\frac{1}{H\varepsilon}\left[\Gamma\left(\frac{d}{2}+1\right)\right]^{1/d}\left[\ln\left(\frac{1}{1-\left(1-\frac{\varepsilon^d\delta}{\pi^{d/2}}\right)^{2/d}}\right)\right]^{\frac{1}{2}} \lesssim c,$$

ETNA
Kent State University
etna@mcs.kent.edu

provided $\varepsilon < \sqrt{\pi}/\delta^{1/d}$. W is minimized if and only if c is, hence we choose (keeping only the main terms)

$$c \sim \left[\Gamma\left(\frac{d}{2}+1\right)\right]^{1/d} \left[\ln\left(\frac{1}{\varepsilon^d \delta}\right) \ln\frac{1}{\delta}\right]^{\frac{1}{2}}$$

Using the above results on H, p, and c in (4.9), we can evaluate (4.1) for the GA RBF in

$$W \sim O\left(\left(\ln\frac{1}{\delta}\right)^d (n+m) + \Gamma\left(\frac{d}{2}+1\right)\left[\ln\left(\frac{1}{\varepsilon^d\delta}\right)\ln\frac{1}{\delta}\right]^{\frac{d}{2}} \varepsilon^d\right)$$

operations (for $d = 1$, this is identical to (3.6)). For $\varepsilon \lesssim n^{1/d}$, W scales linearly with n and m. Like the 1-D algorithm, if $\varepsilon \gg n$ then the original sum can be truncated similarly to (4.8) and directly evaluated in $O(n+m)$ operations.

5.2. MQ kernel. Let $\phi(r) = (1+(\varepsilon r)^2)^{\frac{1}{2}}$. We set $c = N_{\max} = \max_{1\le k\le d} N_k$ because ϕ grows as $r \to \infty$. The bounds on (4.10) for MQ are given by (B.6) with $\nu = 1$. Using this result and Stirling's asymptotic formula (3.2) we obtain the accuracy estimate

$$E \sim \sum_{k=1}^{d} \binom{d}{k} \frac{\sqrt{2}}{kp} \left(\frac{2}{\sqrt{\pi p}}\right)^k \left(\frac{H\varepsilon p\sqrt{k}}{2e}\right)^{kp} . \tag{5.2}$$

Requiring $H\varepsilon p\sqrt{d} < 2eb$ for some $0 < b < 1$, the whole sum is asymptotic to b^p. We can thus bound E by $O(\delta)$ by choosing $p = O(\ln(1/\delta))$. Therefore,

$$H \sim \frac{1}{\varepsilon p\sqrt{d}} \tag{5.3}$$

$$p \sim \ln\frac{1}{\delta} . \tag{5.4}$$

In practice, p is rounded to the next even integer.

Using the above results on H, p, and noting that $c \sim 1/H$, then from (4.9), we can evaluate (4.1) for the MQ RBF in

$$W \sim O\left(\left(\ln\frac{1}{\delta}\right)^d (n+m) + \left(\varepsilon\sqrt{d}\ln\frac{1}{\delta}\right)^{2d}\right)$$

operations. Hence, the algorithm scales linearly with n and m for all $\varepsilon \lesssim n^{1/(2d)}$. For larger ε, W is dominated by the coarse level summation. As discussed in §4.2, we can reduce the complexity of this operation to $O\left(\left(\varepsilon\sqrt{d}\ln(1/\delta)\right)^d \ln\left(\varepsilon\sqrt{d}\ln(1/\delta)\right)^d\right)$ with the FFT.

5.3. IMQ kernel. Let $\phi(r) = (1+(\varepsilon r)^2)^{-\frac{1}{2}}$. This kernel does not decay fast enough, so we again set $c = N_{\max} = \max_{1\le k\le d} N_k$. The bounds on (4.10) applied for IMQ are given by (B.6) with $\nu = -1$. Using this result and Stirling's asymptotic formula (3.2), we obtain the accuracy estimate

$$E \sim \sum_{k=1}^{d} \binom{d}{k} \sqrt{2} \left(\frac{2}{\sqrt{\pi p}}\right)^k \left(\frac{H\varepsilon p\sqrt{k}}{2e}\right)^{kp} .$$

By choosing H and p according to (5.3) and (5.4), respectively, we obtain the same accuracy estimates as the MQ kernel. We thus do not elaborate further on the IMQ kernel.

ETNA
Kent State University
etna@mcs.kent.edu

6. Numerical results. We verify the complexity and accuracy results of our algorithm for a set of test examples similar to those presented in [39]. In all experiments, we set $\varepsilon = (n^{1/2d})/4$, select expansion coefficients $\{\lambda(y_j)\}_{j=0}^n$ randomly from $[-1,1]$, and measure the relative error E according to (2.10). In all cases, the reported results are averaged over at least five different random choices of $\{\lambda(y_j)\}_{j=0}^n$. Note that in all but the last test problem, the fast coarse summation technique of §4.2 is not employed.

To guarantee the accuracy of the algorithm is $O(\delta)$, the input parameters p, c, H for the different ϕ are chosen as follows:

- *GA kernel*. We choose $\delta_{\mathcal{I}} = \delta_{\mathcal{T}} = \delta/2$ to enforce $\delta_{\mathcal{I}} + \delta_{\mathcal{T}} \le \delta$. For H and p we use the same 1-D values given by (3.7) and (3.8), respectively. For c, we use the value

$$c = \begin{cases} \left\lceil \frac{1}{H\varepsilon}\left[\Gamma\left(\frac{d}{2}+1\right)\right]^{1/d}\left[\ln\left(\frac{1}{1-\left(1-\frac{\varepsilon^d\delta}{2\pi^{d/2}}\right)^{2/d}}\right)\right]^{\frac{1}{2}}\right\rceil & \varepsilon < \sqrt{\pi}\left(\frac{2}{\delta}\right)^{1/d}, \\ 0 & \varepsilon \ge \sqrt{\pi}\left(\frac{2}{\delta}\right)^{1/d}. \end{cases}$$

- *MQ and IMQ kernels*. No truncation is performed (i.e. $\delta_{\mathcal{T}} = 0$);

$$H = \frac{2eb}{\varepsilon\overline{p}\sqrt{d}}\,,$$

  where $\overline{p}$ is given by (3.10); and p is given by (3.12).

The paramters could be further optimized by optimizing b for each test case; this is not necessary because good results are obtained for a wide range of b values. In all the following results, b was chosen in $[0.25, 0.35]$.

EXAMPLE 6.1. We consider evaluation with the GA kernel with n centers $\{\mathbf{y}_j\}_{j=0}^n$ and $m = n$ evaluation points $\{\mathbf{x}_i\}_{i=0}^m$ uniformly distributed in $[0,1]^2$ (they do not necessarily coincide). Table 6.1 shows the relative error E of our fast evaluation method. As expected, $E < \delta$ in all cases. Table 6.2 similarly compares the number of operations required for our method versus a direct evaluation. The work scales as $W \approx \alpha\left(\ln(1/\delta)\right)^2(n+m)$, where $2.3 < \alpha < 6.2$.

TABLE 6.1
Relative error E of the multilevel evaluation method for Example 6.1 (GA kernel and uniform distribution in $[0,1]^2$).

$n=m$	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$
1000	$7.67 \cdot 10^{-3}$	$2.68 \cdot 10^{-5}$	$1.22 \cdot 10^{-7}$	$2.81 \cdot 10^{-9}$	$1.29 \cdot 10^{-11}$
2000	$5.37 \cdot 10^{-3}$	$2.60 \cdot 10^{-5}$	$6.86 \cdot 10^{-8}$	$3.88 \cdot 10^{-9}$	$1.67 \cdot 10^{-11}$
4000	$7.76 \cdot 10^{-3}$	$2.81 \cdot 10^{-5}$	$1.41 \cdot 10^{-7}$	$2.29 \cdot 10^{-9}$	$9.12 \cdot 10^{-12}$
8000	$7.37 \cdot 10^{-3}$	$2.40 \cdot 10^{-5}$	$1.25 \cdot 10^{-7}$	$2.61 \cdot 10^{-9}$	$1.19 \cdot 10^{-11}$
16000	$6.96 \cdot 10^{-3}$	$3.30 \cdot 10^{-5}$	$9.78 \cdot 10^{-8}$	$2.69 \cdot 10^{-9}$	$1.29 \cdot 10^{-11}$

EXAMPLE 6.2. We consider non-uniformly distributed points in $[0,1]^2$ with the MQ kernel. The n centers are randomly placed within one tenth of the diagonal of the unit square ("track data"), while the $m = n$ evaluation points are uniformly distributed in $[0,1]^2$. Table 6.3 shows the relative error E of our fast evaluation method. Similarly to the 1-D example of §3.2.1, we see that $E \ll \delta$, especially as $\delta \to 0$ (i.e. as p grows). This is most

TABLE 6.2
Comparison of the work (number of floating-point operations) W required of the multilevel evaluation method for Example 6.1. The right column indicates the number of operations required for a direct evaluation.

n	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$	direct
1000	222390	755430	2127278	3344730	6619962	$8.00 \cdot 10^6$
2000	409350	1400574	3252254	5329450	9885050	$3.20 \cdot 10^7$
4000	773678	2416630	5528558	8625914	15167242	$1.28 \cdot 10^8$
8000	1483126	4482486	9628334	14335274	25401362	$5.12 \cdot 10^8$
16000	2879694	8308262	17660518	25416082	42800330	$2.05 \cdot 10^9$

likely because our asymptotic error bound for (5.2) does not account for the $p^{k/2+1}$ quantity in the denominator of the terms in the summation. The complexity W for this example is similarly presented in Table 6.4. Again, the method scales linearly with n and m. We observe that for smaller δ the break-even point between our fast method and the direct method occurs for larger n (e.g., for $\delta = 10^{-10}$, at $n \approx 2500$). However, in these cases the actual error $E \ll \delta$ and is close to machine precision; still, for $E \geq 10^{-10}$ our method is faster than direct summation for all $n \geq 1000$.

TABLE 6.3
Relative error E of the multilevel evaluation method for Example 6.2 (MQ kernel and track data).

$n = m$	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$
1000	$2.61 \cdot 10^{-4}$	$1.91 \cdot 10^{-7}$	$2.51 \cdot 10^{-10}$	$2.25 \cdot 10^{-13}$	$6.26 \cdot 10^{-15}$
2000	$1.51 \cdot 10^{-4}$	$1.60 \cdot 10^{-7}$	$1.02 \cdot 10^{-10}$	$3.15 \cdot 10^{-13}$	$1.17 \cdot 10^{-14}$
4000	$2.14 \cdot 10^{-4}$	$1.53 \cdot 10^{-7}$	$1.91 \cdot 10^{-10}$	$1.07 \cdot 10^{-13}$	$7.47 \cdot 10^{-15}$
8000	$3.98 \cdot 10^{-4}$	$1.20 \cdot 10^{-7}$	$1.38 \cdot 10^{-10}$	$1.27 \cdot 10^{-13}$	$1.27 \cdot 10^{-14}$
16000	$1.19 \cdot 10^{-4}$	$6.64 \cdot 10^{-8}$	$9.12 \cdot 10^{-11}$	$1.33 \cdot 10^{-13}$	$3.65 \cdot 10^{-14}$

TABLE 6.4
Comparison of the work (number of floating-point operations) W required of the multilevel evaluation method for Example 6.2. The right column indicates the number of operations required for a direct evaluation.

$n = m$	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$	direct
1000	392802	1511234	5049898	12628610	33812486	$8.00 \cdot 10^6$
2000	687802	2516386	7326882	19076506	47297334	$3.20 \cdot 10^7$
4000	1311362	4236162	11561786	28960970	69457774	$1.28 \cdot 10^8$
8000	2599938	7427058	19664850	46111210	110157894	$5.12 \cdot 10^8$
16000	5024234	13636026	32958874	75800306	172780942	$2.05 \cdot 10^9$

EXAMPLE 6.3. We use the same "track data" setup as Example 6.2, but investigate the case where $n \ll m$, which typically occurs in applications. The exact relationship is $m = 10n$ evaluation points uniformly distributed in $[0,1]^2$. The relative error for this example is shown in Table 6.5. The observed $E \ll \delta$ can again be explained as in Example 6.2. Table 6.4 displays the operation count, which grows linearly with n and m. Here whenever the measured E is $O(\delta)$, the complexity of our fast evaluation is lower than a direct evaluation.

EXAMPLE 6.4. We consider evaluation of the IMQ kernel with n centers and $m = n$ evaluation points uniformly distributed in $[0,1]^3$. This simulates force calculation in an n-

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 6.5
Relative error E of the multilevel evaluation method for Example 6.3 (MQ kernel, track data, and $m = 10n$).

m	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$
2000	$3.12 \cdot 10^{-4}$	$2.72 \cdot 10^{-7}$	$3.04 \cdot 10^{-10}$	$3.53 \cdot 10^{-13}$	$4.63 \cdot 10^{-15}$
4000	$2.59 \cdot 10^{-4}$	$2.17 \cdot 10^{-7}$	$2.30 \cdot 10^{-10}$	$2.94 \cdot 10^{-13}$	$5.80 \cdot 10^{-15}$
8000	$2.98 \cdot 10^{-4}$	$2.35 \cdot 10^{-7}$	$2.67 \cdot 10^{-10}$	$3.20 \cdot 10^{-13}$	$7.22 \cdot 10^{-15}$
16000	$2.52 \cdot 10^{-4}$	$2.26 \cdot 10^{-7}$	$2.53 \cdot 10^{-10}$	$3.14 \cdot 10^{-13}$	$9.02 \cdot 10^{-15}$
32000	$3.18 \cdot 10^{-4}$	$2.60 \cdot 10^{-7}$	$2.80 \cdot 10^{-10}$	$3.38 \cdot 10^{-13}$	$1.72 \cdot 10^{-14}$
64000	$1.49 \cdot 10^{-4}$	$8.68 \cdot 10^{-8}$	$9.37 \cdot 10^{-11}$	$1.33 \cdot 10^{-13}$	$1.25 \cdot 10^{-14}$

TABLE 6.6
Comparison of the work (number of floating-point operations) W required of the multilevel evaluation method for Example 6.3. The right column indicates the number of operations required for a direct evaluation.

m	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$	direct
2000	372234	1164906	3172234	7533306	19002414	$3.20 \cdot 10^{6}$
4000	717634	1975122	5092482	11646866	26958734	$1.28 \cdot 10^{7}$
8000	1364834	3665794	8366586	17364106	41132334	$5.12 \cdot 10^{7}$
16000	2686602	6662458	14243802	29371442	63761742	$2.05 \cdot 10^{8}$
32000	5308962	12680186	25961810	50172050	104153542	$8.19 \cdot 10^{8}$
64000	10530994	24709058	48637778	87890530	173708942	$3.28 \cdot 10^{9}$

body simulation with the Plummer potential [39]. Table 6.7 displays the relative error, and again E is well below δ. Table 6.8 shows operation counts; here we use the FFT coarse summation (§4.2). The algorithm scales linearly with n, m and $(\ln(1/\delta))^q$ where $q \approx 2.5$ (i.e. slightly better than the expected $q = 3$, for $n \leq 10^5$). Our method is faster than direct summation in all cases except $n = 5000$ and $\delta = 10^{-10}$ (where again $E \approx 10^{-14} \ll \delta$).

TABLE 6.7
Relative error E of the multilevel evaluation method with fast coarse level summation for Example 6.4 (IMQ kernel and uniform distribution in $[0, 1]^3$).

$n = m$	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$
5000	$1.91 \cdot 10^{-3}$	$8.85 \cdot 10^{-7}$	$3.65 \cdot 10^{-9}$	$1.84 \cdot 10^{-12}$	$2.62 \cdot 10^{-14}$
10000	$3.63 \cdot 10^{-3}$	$1.90 \cdot 10^{-6}$	$8.39 \cdot 10^{-9}$	$4.79 \cdot 10^{-12}$	$4.17 \cdot 10^{-14}$
20000	$2.19 \cdot 10^{-3}$	$1.11 \cdot 10^{-6}$	$4.43 \cdot 10^{-9}$	$2.64 \cdot 10^{-12}$	$3.61 \cdot 10^{-14}$
40000	$2.72 \cdot 10^{-3}$	$1.25 \cdot 10^{-6}$	$5.16 \cdot 10^{-9}$	$3.13 \cdot 10^{-12}$	$6.07 \cdot 10^{-14}$
80000	$2.71 \cdot 10^{-3}$	$1.12 \cdot 10^{-6}$	$4.42 \cdot 10^{-9}$	$2.31 \cdot 10^{-12}$	$5.36 \cdot 10^{-14}$
100000	$1.69 \cdot 10^{-3}$	$6.68 \cdot 10^{-7}$	$2.56 \cdot 10^{-9}$	$1.40 \cdot 10^{-12}$	$6.05 \cdot 10^{-14}$

6.1. Comparison with other fast methods. Preliminary results comparing the performance of the multilevel approach with FMM and FGT for various smooth kernels and expansion coefficients suggest that the multilevel approach has a uniformly bounded evaluation error for all ε, whereas the other methods may have an uncontrolled error (or uncontrolled complexity) for some regions of the shape parameter. However, more extensive and systematic comparison of the methods is called for, which is left to a future paper.

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 6.8

Comparison of the work (number of floating-point operations) W required of the multilevel evaluation method for Example 6.4 with fast coarse level summation. The right column indicates the number of operations required for a direct evaluation.

$n = m$	$\delta = 10^{-2}$	$\delta = 10^{-4}$	$\delta = 10^{-6}$	$\delta = 10^{-8}$	$\delta = 10^{-10}$	direct
5000	$3.02 \cdot 10^6$	$2.91 \cdot 10^7$	$5.86 \cdot 10^7$	$1.35 \cdot 10^8$	$2.83 \cdot 10^8$	$2.75 \cdot 10^8$
10000	$6.16 \cdot 10^6$	$4.15 \cdot 10^7$	$8.14 \cdot 10^7$	$2.78 \cdot 10^8$	$5.83 \cdot 10^8$	$1.10 \cdot 10^9$
20000	$1.06 \cdot 10^7$	$6.61 \cdot 10^7$	$1.66 \cdot 10^8$	$3.97 \cdot 10^8$	$8.29 \cdot 10^8$	$4.40 \cdot 10^9$
40000	$1.95 \cdot 10^7$	$1.15 \cdot 10^8$	$2.57 \cdot 10^8$	$6.34 \cdot 10^8$	$1.32 \cdot 10^9$	$1.76 \cdot 10^{10}$
80000	$3.93 \cdot 10^7$	$2.32 \cdot 10^8$	$4.39 \cdot 10^8$	$1.28 \cdot 10^9$	$2.67 \cdot 10^9$	$7.04 \cdot 10^{10}$
100000	$4.82 \cdot 10^7$	$2.82 \cdot 10^8$	$5.31 \cdot 10^8$	$1.52 \cdot 10^9$	$3.16 \cdot 10^9$	$1.10 \cdot 10^{11}$

7. Concluding remarks. We presented a fast RBF evaluation algorithm for smooth radial kernels. It applies to other applications of many-body interactions with smooth kernels (e.g., image processing and atomistic simulations). The algorithm scales linearly with the number of centers and with the number of evaluation points. Each additional evaluation can be performed separately and costs $O((\ln(1/\delta)^d))$ where δ is the desired accuracy and d is the dimension. Numerical results with GA and GMQ RBFs fit the theoretical accuracy and work estimates. This fast evaluation will hopefully provide an important tool that will be easily implemented and integrated into existing RBF interpolation/approximation software, and will allow faster solutions of large-scale interpolation problems. We plan to expand the MATLAB code whose results were presented here, to a general-purpose library of fast RBF expansion evaluations. Directions for future research follow.

Piecewise Smooth RBFs. The evaluation algorithm can be generalized to piecewise smooth kernels as developed in [11, 14, 37]. Here $O(\ln n)$ levels must be employed rather than two. The kernel is decomposed into a *smooth* (interpolated from a coarse level) and a *local* (directly summed) parts via "kernel softening" near $r = 0$ [11]. To control the local part's evaluation complexity, local grid refinements should be employed at areas of high center/evaluation point densities [13].

Fast Fitting. The fast evaluation provides fast matrix-vector multiplication to be integrated to any of the existing iterative methods such as Krylov-subspace method [2, 3, 21, 22, 23]. Moreover, for some piecewise smooth kernels (e.g. GDS) the fitting problem can be solved by a multigrid cycle within a Full Multigrid (FMG) algorithm, along the lines of [12]. The fast evaluation is again employed to compute residuals. The cost of solving the fitting problem to a reasonable tolerance (analogous to the truncation errors in discretizing (1.3) at the centers) is equivalent to $2 - 3$ evaluations of s at all centers.

8. Acknowledgements. The first author was supported by the U.S. Department of Energy through the Center for Simulation of Accidental Fires and Explosions, under grant W-7405-ENG-48. The second author was supported by the National Science Foundation VIGRE grant DMS-0091675.

The authors would also like to thank Dr. Bengt Fornberg and Ms. Cecil Piret for providing prelimary results on the FMM and FGT algorithms.

Appendix A. Calculation of interpolation weights in $O(p)$ operations. The 1-D pth-order polynomial that interpolates $\{f_j\}_{j=0}^{p-1}$ at $\{x_j\}_{j=0}^{p-1}$ can be conveniently expressed using the *barycentric* formula [8]

$$q(x) = \frac{\sum_{j=0}^{p-1} \frac{c_j}{x-x_j} f_j}{\sum_{j=0}^{p-1} \frac{c_j}{x-x_j}} = \sum_{j=0}^{p-1} \left(\frac{\frac{c_j}{x-x_j}}{\sum_{i=0}^{p-1} \frac{c_i}{x-x_i}} \right) f_j =: \sum_{j=0}^{p-1} \omega_j f_j \,,$$

where $c_j = (\prod_{i \neq j}(x_j - x_i))^{-1}$, $j = 0, 1, \dots, p-1$. For equally spaced $\{x_j\}_{j=0}^{p-1}$, it is shown in [8, p.505] that, up to a multiplicative constant, $c_j = (-1)^j \binom{p-1}{j}$. Because the c_j appear symmetrically in the numerator and denominator, the multiplicative constant is inconsequential and $\{c_j\}_{j=0}^{p-1}$ can be precomputed once-for-all. Given x, the Lagrange interpolation weights $\{\omega_j\}_{j=0}^{p-1}$ are thus computed in $O(p)$ operations using the following algorithm.

$$\begin{aligned} \tilde{\omega}_j &\longleftarrow c_j/(x - x_j), \quad j = 0, \dots, p-1 \\ \tilde{\omega} &\longleftarrow \sum_{j=0}^{p-1} \tilde{\omega}_j \\ \omega_j &\longleftarrow \tilde{\omega}_j / \tilde{\omega}, \quad j = 0, \dots, p-1. \end{aligned}$$

One should be careful about the numerical stability of the computation. In the second step, for instance, $\{\tilde{\omega}_j\}_{j=0}^{p-1}$ should be sorted to minimize floating-point arithmetic round-off.

Appendix B. Bounds on the derivatives of $\phi(r)$. The GA kernel plays a central role in deriving the bounds for (4.10), thus we first consider it. We use the well-known result

$$e^{-(\varepsilon\|\boldsymbol{\xi}\|)^2} = \left(\frac{1}{\pi^{k/2}} \prod_{j=1}^{k} \int_0^\infty e^{-\frac{\omega^2}{4}} \cos(\omega\varepsilon\xi_j) d\omega \right) \prod_{j=k+1}^{d} e^{-(\varepsilon\xi_j)^2}, \quad 1 \le k \le d \,, \tag{B.1}$$

Thus, it sufficient to prove bounds for $d = 1$ and then extend them inductively to any $d > 1$. Recalling that p is even, we have for $d = 1$ that

$$\begin{aligned} \left\| \frac{d^p}{d\xi_1^p} e^{-(\varepsilon\xi_1)^2} \right\|_\infty &= \frac{1}{\sqrt{\pi}} \left\| \int_0^\infty e^{-\frac{\omega^2}{4}} (-1)^{p/2} (\varepsilon\omega)^p \cos(\omega\varepsilon\xi_1) d\omega \right\|_\infty \\ &\le \frac{\varepsilon^p}{\sqrt{\pi}} \int_0^\infty e^{-\frac{\omega^2}{4}} \omega^p d\omega = \varepsilon^p \left| \frac{d^p}{d\xi_1^p} e^{-\xi_1^2} \Big|_{\xi_1 = 0} \right| \\ &= \frac{p!\varepsilon^p}{(p/2)!} \,. \end{aligned}$$

The last equality follows from [1, p. 933]. By induction, we thus obtain

$$\left\| \frac{\partial^{kp} e^{-(\varepsilon\|\boldsymbol{\xi}\|)^2}}{\partial\xi_1^p \partial\xi_2^p \cdots \partial\xi_k^p} \right\|_\infty \le \left(\frac{p!\varepsilon^p}{(p/2)!} \right)^k \prod_{j=k+1}^{d} e^{-(\varepsilon\xi_j)^2} \le \left(\frac{p!\varepsilon^p}{(p/2)!} \right)^k, \quad 1 \le k \le d. \tag{B.2}$$

To derive the bounds for the MQ and IMQ kernels we need the following definition:

DEFINITION B.1. *A function Φ is said to be completely monotone on $[0, \infty)$ if*

1. $\Phi \in C\,[0, \infty)$
2. $\Phi \in C^\infty\,(0, \infty)$
3. $(-1)^\ell \dfrac{d^\ell}{dr^\ell} \Phi(r) \ge 0$ *for* $r > 0$ *and* $\ell = 0, 1, 2, \dots$

ETNA
Kent State University
etna@mcs.kent.edu

Many radial kernels $\phi(r)$ used in RBF interpolation have the property that $\Phi(r) = \phi(\sqrt{r})$ is completely monotone because then the existence and uniqueness of an interpolant is guaranteed [15, §2.1]. For example, the GMQ kernel with $\nu < 0$ (see Table 1.1) can be easily shown to have this property. Completely monotone functions are also classified by the Bernstein-Widder theorem [18, Ch.14]:

THEOREM B.2. *A function $\Phi(r)$, $r \geq 0$, is completely monotone if and only if it can expressed in the form*

$$\Phi(r) = \int_0^\infty e^{-sr} d\mu(s), \tag{B.3}$$

where μ is a nondecreasing bounded measure.

For radial kernels that satisfy $\Phi(r) = \phi(\sqrt{r})$ is completely monotone, this theorem allows them to be related to the GA kernel. We can thus use the bounds (B.2) to obtain bounds on (4.10) for these kernels (the validity for differentiating under the integral sign in (B.3) is justified in [18, p.97]). We illustrate the bounding procedure for the GMQ kernel with $\nu < 0$, which has the Bernstein-Widder form

$$(1 + (\varepsilon\|\boldsymbol{\xi}\|)^2)^{\nu/2} = \frac{1}{\Gamma(-\frac{\nu}{2})}\int_0^\infty \frac{e^{-s}}{s^{\nu/2+1}} e^{-s(\varepsilon\|\boldsymbol{\xi}\|)^2} ds \ , \quad \nu < 0, \tag{B.4}$$

as is easily verified by *Mathematica*. For $d = 1$, make the substitution

$$e^{-s(\varepsilon\xi_1)^2} = \frac{1}{\sqrt{\pi}}\int_0^\infty e^{-\frac{\omega^2}{4}}(-1)^{p/2}(\sqrt{s}\varepsilon\omega)^p \cos(\omega\sqrt{s}\varepsilon x) d\omega \ .$$

Differentiating under the integral of (B.4) then gives the bounds

$$\begin{aligned}\left\|\frac{d^p}{d\xi_1^p}(1+(\varepsilon\xi_1)^2)^{\nu/2}\right\|_\infty &\leq \frac{p!\varepsilon^p}{(p/2)!}\frac{1}{\Gamma(-\frac{\nu}{2})}\int_0^\infty e^{-s}s^{p/2-(\nu/2+1)}ds \\ &= \frac{p!\varepsilon^p}{(p/2)!}\frac{\Gamma(\frac{p-\nu}{2})}{\Gamma(-\frac{\nu}{2})}, \quad \nu < 0.\end{aligned}$$

The last equality follows from the definition of the Gamma function [1, p.255]. Using the GA kernel property (B.1), the bounds

$$\begin{aligned}\left\|\frac{\partial^{kp}(1+(\varepsilon\|\boldsymbol{\xi}\|)^2)^{\nu/2}}{\partial\xi_1^p\partial\xi_2^p\cdots\partial\xi_k^p}\right\|_\infty &\leq \left(\frac{p!\varepsilon^p}{(p/2)!}\right)^k \frac{1}{\Gamma(-\frac{\nu}{2})}\int_0^\infty e^{-s}s^{p/2-(\nu/2+1)}ds \\ &= \left(\frac{p!\varepsilon^p}{(p/2)!}\right)^k \frac{\Gamma(\frac{kq-\nu}{2})}{\Gamma(-\frac{\nu}{2})}, \quad 1 \leq k \leq d, \ \nu < 0,\end{aligned}$$

follow by induction.

To obtain the bounds for the MQ kernel (i.e. $\nu = 1$), we again need to relate it to the GA kernel. The following theorem of Sun (c.f. [18, p.110]) provides the relation:

THEOREM B.3. *Let $\Phi(r) \in C^\infty(0,\infty)$, continuous at zero, and $\frac{d}{dr}\Phi(r)$ be completely monotone but not constant on $(0,\infty)$. Then it is necessary and sufficient that $\Phi(r)$ have the form*

$$\Phi(r) = \Phi(0) + \int_0^\infty \frac{1-e^{-sr}}{s} d\mu(s) \ , \ r \geq 0, \tag{B.5}$$

ETNA
Kent State University
etna@mcs.kent.edu

for some nontrivial Borel measure $\mu : [0, \infty) \to \mathbb{R}$ *satisfying* $\int_{(1,\infty)} s^{-1} d\mu(s) < \infty$ *and* $\mu((0,t)) < \infty$ *for all* t.

For radial kernels $\phi(\sqrt{r}) = \Phi(r)$ satisfying this requirement (e.g. GMQ with $0 < \nu < 2$), we can thus use the GA kernel bounds (B.2) to obtain the desired bounds (4.10). The GMQ kernel with $0 < \nu < 2$, can be expressed in the form (B.5) as follows

$$(1 + (\varepsilon\|\boldsymbol{\xi}\|)^2)^{\nu/2} = 1 - \frac{1}{\Gamma(-\frac{\nu}{2})} \int_0^\infty \frac{e^{-s}}{s^{\nu/2}} \left(\frac{1 - e^{-s(\varepsilon\|\boldsymbol{\xi}\|)^2}}{s} \right) ds\,, \quad 0 < \nu < 2.$$

as is easily verified by *Mathematica*. Upon differentiating, the terms independent of $\boldsymbol{\xi}$ vanish and we are left with the exact same problem as the GMQ kernel with $\nu < 0$. Thus, we have the general result

$$\left\| \frac{\partial^{kp}(1 + (\varepsilon\|\boldsymbol{\xi}\|)^2)^{\nu/2}}{\partial\xi_1^p \partial\xi_2^p \cdots \partial\xi_k^p} \right\|_\infty \le \left(\frac{p!\varepsilon^p}{(p/2)!} \right)^k \left| \frac{\Gamma(\frac{kq-\nu}{2})}{\Gamma(-\frac{\nu}{2})} \right|, \quad 1 \le k \le d,\ \nu < 2. \tag{B.6}$$

The MQ and IMQ bounds are given by $\nu = 1$ and $\nu = -1$, respectively. We expect similar result for $\nu > 2$ by integrating (B.5) the necessary number of times.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, eds., *Handbook of mathematical functions–with formulas, graphs, and mathematical tables*, Dover, New York, 1972.

[2] B. J. C. Baxter, *The interpolation theory of radial basis functions*, PhD thesis, Trinity College, University of Cambridge, 1992.

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

[5] R. K. Beatson and L. Greengard, *A short course on fast multipole methods*, in Wavelets, Multilevel Methods, and Elliptic PDEs, J. Levesley, W. Light, and M. Marletta, eds., Oxford University Press, Oxford, 1997, pp. 1–37.

[6] R. K. Beatson and W. A. Light, *Fast evaluation of radial basis functions: methods for 2-dimensional polyharmonic splines*, IMA J. Numer. Anal., 17 (1997), pp. 343–372.

[7] R. K. Beatson and G. N. Newsam, *Fast evaluation of radial basis functions, part I*, Comput. Math. Appl., 24 (1992), pp. 7–19.

[8] J. P. Berrut and L. N. Trefethen, *Barycentric lagrange interpolation*, SIAM Rev., 46 (2004), pp. 501–517.

[9] S. D. Billings, R. K. Beatson, and G. N. Newsam, *Interpolation of geophysical data with continuous global surfaces*, Geophysics, 67 (2002), pp. 1810–1822.

[10] S. D. Billings, G. N. Newsam, and R. K. Beatson, *Smooth fitting of geophysical data with continuous global surfaces*, Geophysics, 67 (2002), pp. 1823–1834.

[11] A. Brandt, *Multilevel computations of integral transforms and particle interaction with oscillatory kernels*, Comput. Phys. Comm., 65 (1991), pp. 24–38.

[12] A. Brandt and A. A. Lubrecht, *Multilevel matrix multiplication and fast solution of integral equations*, J. Comput. Phys., 90 (1990), pp. 348–370.

[13] A. Brandt and C. H. Venner, *Multilevel evaluation of integral transforms on adaptive grids*, in Multigrid Methods V, W. Hackbusch and G. Wittum, eds., vol. 3 of Lecture Notes in Computational Science and Engineering, Springer, Berlin and Heidelberg, 1998, pp. 21–44.

[14] ———, *Multilevel evaluation of integral transforms with asymptotically smooth kernels*, SIAM J. Sci. Comput., 19 (1998), pp. 468–492.

[15] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations*, Cambridge Monographs on Applied and Computational Mathematics, No. 12, Cambridge University Press, Cambridge, 2003.

[16] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, *Reconstruction and representation of 3d objects with radial basis functions*, in SIGGRAPH '01: Proceedings of the 28th annual conference on computer graphics and interactive techniques, ACM Press, New York, NY, 2001, pp. 67–76.

ETNA
Kent State University
etna@mcs.kent.edu

[17] J. C. CARR, W. R. FRIGHT, AND R. K. BEATSON, *Surface interpolation with radial basis functions for medial imaging*, IEEE Trans. Med. Imaging, 16 (1997), pp. 96–107.
[18] E. W. CHENEY AND W. A. LIGHT, *A Course in Approximation Theory*, Brooks/Cole, New York, 2000.
[19] J. B. CHERRIE, R. K. BEATSON, AND G. N. NEWSAM, *Fast evaluation of radial basis functions: methods for generalized multiquadrics* $\mathbb{R}^n$, SIAM J. Sci. Comput., 23 (2002), pp. 1549–1571.
[20] L. M. DELVES AND J. L. MOHAMED, *Computational Methods for Integral Equations*, Cambridge University Press, Cambridge, 1985.
[21] N. DYN, D. LEVIN, AND S. RIPPA, *Numerical procedures for global surface fitting of scattered data by radial functions*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 639–659.
[22] A. C. FAUL, G. GOODSELL, AND M. J. D. POWELL, *A Krylov subspace alogrithm for multiquadric interpolation in many dimensions*, IMA J. Numer. Anal., 25 (2005), pp. 1–24.
[23] A. C. FAUL AND M. J. D. POWELL, *Krylov subspace methods for radial basis function interpolation*, in Numerical Analysis 1999, D. F. Griffiths, ed., Chapman and Hall, London, 1999, pp. 115–141.
[24] B. FORNBERG, E. LARSSON, AND G. WRIGHT, *A new class of oscillatory radial basis functions*, Comput. Math. Appl., (2006). In press.
[25] R. FRANKE, *Scattered data interpolation: tests of some methods*, Math. Comput., 38 (1982), pp. 181–200.
[26] W. GAUTSCHI, *The incomplete gamma functions since Tricomi*, in Tricomi's ideas and contemporary applied mathematics (Rome/Turin, 1997), vol. 147 of Atti Convegni Lincei, Accad. Naz. Lincei, Rome, 1998, pp. 203–237.
[27] F. GIROSI, *Some extensions of radial basis functions and their applications in artificial intelligence*, Comput. Math. Appl., 24 (1992), pp. 61–80.
[28] G. H. GOLUB AND C. F. V. LOAN, *Matrix Computations*, Third ed., Johns Hopkins Studies in the Mathematical Sciences, The John Hopkins University Press, Baltimore, MD, 1996.
[29] R. C. GONZALEZ AND R. E. WOODS, *Digital Image Processing*, Second ed., Prentice Hall, New Jersey, 2002.
[30] L. GREENGARD AND J. STRAIN, *The fast Gauss transform*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 79–94.
[31] R. L. HARDY, *Multiquadric equations of topograpy and other irregular surfaces*, J. Geophy. Res., 76 (1971), pp. 1905–1915.
[32] N. J. HIGHAM, *The accuracy of floating point summation*, SIAM J. Sci. Comput., 14 (1993), pp. 783–799.
[33] E. J. KANSA, *Multiquadrics – a scattered data approximation scheme with applications to computational fluid-dynamics – I: Surface approximations and parital derivative estimates*, Comput. Math. Appl., 19 (1990), pp. 127–145.
[34] ———, *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.
[35] D. LEE, *Fast multiplication of a recursive block Toeplitz matrix by a vector and its applications*, J. Complexity, 10 (1986), pp. 295–305.
[36] D. LEE AND J. H. SHIAU, *Thin plate splines with discontinuities and fast algorithms for their computation*, SIAM J. Sci. Comput., 14 (1994), pp. 1311–1330.
[37] O. E. LIVNE AND A. BRANDT, *N roots of the secular equation in $O(N)$ operations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 439–453.
[38] G. ROUSSOS AND B. J. C. BAXTER, *A scalable method for many body computations*, in Recent Advances in Parallel Virtual Machine and Message Passing Interface, Eighth European PVM/MPI Users' Group Meeting, Santorini/Thera, Greece, September 23–26, Y. Cotronis and J. Dongarra, eds., vol. 2131 of Lecture Notes in Computer Science, Springer, Berlin and Heidelberg, 2001, pp. 480–488.
[39] ———, *Rapid evaluation of radial basis functions*, J. Comput. Appl. Math., 180 (2005), pp. 51–70.
[40] R. SCHABACK, *Error estimates and condition numbers for radial basis function interpolants*, Adv. Comput. Math., 3 (1995), pp. 251–264.
[41] M. SHIN AND C. PARK, *A radial basis function approach to pattern recognition and its applications*, ETRI Journal, 22 (2000), pp. 1–10.
[42] J. H. STEFFENSEN, *Interpolation*, Chelsea, New York, 1950.
[43] H. WENDLAND, *Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree*, Adv. Comput. Math., 4 (1995), pp. 389–396.
[44] C. A. ZALA AND I. BARRODALE, *Warping aerial photographs to orthomaps using thin plate splines*, Adv. Comput. Math, 11 (1999), pp. 211–227.