

## FAST EVALUATION OF RADIAL BASIS FUNCTIONS: MOMENT-BASED METHODS\*

R. K. BEATSON<sup>†</sup> AND G. N. NEWSAM<sup>‡</sup>

**Abstract.** This paper presents a new method for the fast evaluation of univariate radial basis functions of the form  $s(x) = \sum_{n=1}^N d_n \phi(|x - x_n|)$  to within accuracy  $\epsilon$ . The method can be viewed as a generalization of the fast multipole method in which calculations with far field expansions are replaced by calculations involving moments of the data. The method has the advantage of being adaptive to changes in  $\phi$ . That is, with this method changing to a new  $\phi$  requires only coding a one- or two-line function for the (slow) evaluation of  $\phi$ . In contrast, adapting the usual fast multipole method to a new  $\phi$  involves much mathematical analysis of appropriate series expansions and corresponding translation operators, followed by a substantial amount of work expressing this mathematics in code.

**Key words.** fast multiple method, fast evaluation, moment-based method, radial basis functions

**AMS subject classifications.** 65D07, 65D20, 41A15

**PII.** S1064827595293569

**1. Introduction.** In this paper we introduce a new algorithm for fast evaluation of univariate radial basis functions of the form  $s(x) = \sum_{n=1}^N d_n \phi(|x - x_n|)$  to within accuracy  $\epsilon$ . The algorithm has a setup cost of  $\mathcal{O}(N |\log \epsilon| \log |\log \epsilon|)$  operations and an incremental cost per evaluation of  $s(x)$  of  $\mathcal{O}(|\log \epsilon|)$  operations. It is based on a hierarchical subdivision of the unit interval, the adaptive construction of a corresponding hierarchy of polynomial approximations, and the fast accumulation of moments. It can be applied in any case where the basic function  $\phi$  is smooth on  $(0, 1]$  and on any grid of centers  $\{x_n\}$ . The algorithm does not require that  $\phi$  be analytic at infinity, nor that the user specify new polynomial approximations or modify the data structures for each new  $\phi$ , nor that the points  $x_n$  form any sort of regular array. Furthermore, the algorithm can be extended to problems in higher dimensions.

The algorithm has its roots in a study of the very interesting and important new families of fast evaluation algorithms developed over the last decade. These enable rapid calculation of approximations to potential fields and matrix–vector products of the form  $A\mathbf{d}$  where  $a_{ij} = \phi(|x_i - x_j|)$ . Algorithms such as tree codes [4, 5], fast panel methods [14], and fast multipole methods [7, 9, 10, 11] have essentially reduced the cost of evaluating  $A\mathbf{d}$  to within accuracy  $\epsilon \|\mathbf{d}\|_1$  to  $\mathcal{O}(N \log N |\log \epsilon|^2)$  or even  $\mathcal{O}(N |\log \epsilon| \log |\log \epsilon|)$  operations. Thus these algorithms can dramatically reduce the operation count in problems where  $N$  is large, although at the cost of a relatively complex algorithm structure. The main purpose of this paper is to present a related algorithm that has a simpler and more widely applicable structure; the secondary purpose is to clarify some of the more subtle factors affecting the performance of fast algorithms and to present some associated improvements.

---

\*Received by the editors October 20, 1995; accepted for publication (in revised form) October 1, 1996; published electronically April 22, 1998.

<http://www.siam.org/journals/sisc/19-5/29356.html>

<sup>†</sup>Department of Mathematics and Statistics, University of Canterbury, Private Bag, Christchurch 1, New Zealand (r.beatson@math.canterbury.ac.nz). This research was partially supported by the Cooperative Research Centre for Sensor Signal and Information Processing, The Levels, SA 5109, Australia.

<sup>‡</sup>Wide Area Surveillance Division, Defence Science and Technology Organisation, P.O. Box 1500, Salisbury, SA 5108, Australia (garry.newsam@dsto.defence.gov.au). This research was assisted by an Erskine Fellowship from the University of Canterbury, Christchurch, New Zealand.

At present the most firmly established of these algorithms are the fast multipole methods (perhaps best described by Greengard in [11]). Fast multipole algorithms are based on finite expansions that approximate the far field influence of a given cluster of centers as a sum of truncated Laurent series expansions of  $\phi$ . These expansions are merged into a succession of ever larger clusters in an upward sweep through a tree subdivision of space. This is followed by a downward sweep, in which for each cluster the far field contributions from nearby (but not neighboring) clusters of a similar size are approximated by polynomials, usually through a Taylor series. These polynomials are then added to a polynomial inherited from the parent of the present cluster that summarized the far field contributions at higher levels. The result at the finest level is a polynomial approximation to that part of the radial basis function due to centers far away from the panel of interest; the remaining contribution from nearby centers is then calculated directly. The degree  $k$  of the polynomial approximations is determined by the specified accuracy  $\epsilon < 0.5$ , with the geometric convergence of polynomial approximations to analytic functions usually ensuring that  $k = \mathcal{O}(|\log \epsilon|)$ .

The major innovations in the algorithm presented here are the following. First, the Laurent series coefficients calculated in the upward sweep of the fast multipole algorithms are replaced by discrete moments of the weights  $d_n$  over the locations  $x_n$ . Second, in the downward sweep the polynomial approximations are not formed from Taylor series approximations to Laurent series. Instead they are formed from direct polynomial approximation of the interactions at a given level and the already computed moment sums. Third, the algorithm developed here is fully adaptive and does not require the derivation of expansions, or the modification of software, for each new  $\phi$ . The algorithm should work well for any function  $\phi$  that is smooth away from the origin. Fourth, we identify a general class of functions  $\phi$  to which the algorithm is particularly well suited.

In addition to the above innovations, the paper also looks at some other more subtle issues that influence the overall performance. First, the currently popular strategy of subdividing a cluster only if it contains more than a fixed number of points can lead to worse than linear operation counts on nonuniform distributions of centers: we introduce a new subdivision strategy that removes this problem. Second, we show that a closer analysis of error bounds in the problem indicates that for functions  $\phi$  with singularities at the origin or at infinity (e.g., potential functions), the degree  $k$  of the polynomial approximations will depend explicitly (although usually quite weakly) on the length scales in the problem. As these in turn depend on the number of points,  $N$ ,  $k$  will depend on  $N$  as well as  $\epsilon$ . Finally, we briefly consider the improvements in the operation count possible by separately approximating each of the interactions that must be evaluated at every stage.

The paper is organized as follows. The next section sets out the problem and gives a simple example which motivates the method. It also contains some numerical results illustrating the superior performance of the algorithm compared with direct evaluations. The third section presents some lemmas showing how polynomial approximations to  $s$  may be obtained from discrete moments. The fourth section sets out the data structures and notation used in the remainder of the paper. The fifth section presents a basic outline of the algorithm. The sixth section briefly reviews some methods of adaptive polynomial approximation. The seventh section presents an analysis that establishes for a large class of basic functions a geometric convergence rate for the approximation error similar to that in the standard fast multipole methods. For functions in this class the number of operations required to calculate

the matrix–vector product  $A\mathbf{d}$  is  $\mathcal{O}(N|\log \epsilon| \log |\log \epsilon|)$  and the incremental operation count per additional evaluation of  $s$  is  $\mathcal{O}(|\log \epsilon|)$ , where  $0 < \epsilon < 0.5$  is the required accuracy. The eighth section gives a detailed description of the algorithm. Section 9 presents an analysis of various subdivision strategies. Finally, the appendices briefly discuss three other technical issues that affect the algorithms performance: improved estimates of convergence rates, the use of more than one approximating polynomial at each level, and making efficient use of scale invariance.

After completing the second draft of this paper, the authors became aware of the recent papers of Alpert and Rokhlin [1]; Beylkin, Coifman, and Rokhlin [8]; Alpert [3]; and Alpert, Beylkin, Coifman, and Rokhlin [2]. [1] has already explored some of the ideas developed here in that it introduced the idea of purely polynomial approximation of interactions at a distance. However, it dealt primarily with a particular  $\phi$ , accumulated contributions through sums evaluated at Chebyshev nodes rather than through moments, and gave only a high level description of a general algorithm. [8] presented an important and wide ranging extension of the ideas underlying fast algorithms through the use of wavelet expansions to generate efficient representations of Calderon–Zygmund and related integral operators. [2, 3] extended these results to form efficient representations of systems derived from quadratures of integral equations through the use of waveletlike bases built up of piecewise polynomials and so derived fast algorithms for evaluating and solving these systems. The algorithms in [2] could be applied to the present problem. Nevertheless, we feel that the approach presented here will be competitive with that of [2] and may well be easier to implement and analyze for the particular problem of evaluating radial basis function expansions.

**2. Problem statement and numerical results.** Given an arbitrary collection of centers  $\{x_n\}_{n=1}^N$ , weights  $\{d_n\}_{n=1}^N$ , and a function  $\phi$  that is smooth away from zero, we wish to evaluate

$$(2.1) \quad s(x) = \sum_{n=1}^N d_n \phi(|x - x_n|)$$

to within an accuracy  $\epsilon \|\mathbf{d}\|_1 \equiv \epsilon \sum_{n=1}^N |d_n|$ . We shall refer to  $s$  as a *radial basis function* and to  $\phi$  as the *basic function*. Typical choices for  $\phi$  are the multiquadric  $\phi(x) = \sqrt{x^2 + \lambda^2}$ , the Gaussian  $\phi(x) = e^{-\lambda^2 x^2}$ , and the thin-plate spline  $\phi(x) = x^2 \log(x)$ . As illustrated by these examples,  $\phi$  is typically singular, or nearly singular, at the origin, analytic in the right half plane, and may be analytic at infinity. We shall assume the availability of a routine for calculating  $\phi$  in which the cost in flops  $c_\phi$  of one evaluation is approximately constant.

We shall also assume that the centers  $x_n$  have been translated and scaled so that  $\{x_n\}_{n=1}^N \subset [0, 1]$ . For any subinterval  $P$  of  $[0, 1]$ ,

$$s_P(x) = \sum_{n : x_n \in P} d_n \phi(|x - x_n|)$$

will denote the contribution from centers in this interval. Note that rescaling will affect any scale parameter  $\lambda$  in the basic function and may also change any multiplicative constant associated with the basic function. This constant may be absorbed into the definition of  $\phi$  or into the weights  $d_n$ . In either case rescaling will affect the desired accuracy  $\epsilon \|\mathbf{d}\|_1$  and so will affect thresholds and associated operation counts in the algorithm.

In this setting there are two canonical tasks calling for somewhat different versions of the algorithm. The first is the *matrix-vector product* task in which evaluation of  $s$  is only required at the centers  $\{x_n\}$ . The second is the *general evaluation* task in which  $s$  must be evaluated on an arbitrary set of points that is usually denser than the set of centers. Versions of the algorithm will be presented for both tasks.

We now present a simple example which illustrates some of the ideas underlying the algorithm.

*Example 1.* Suppose that we wish to evaluate  $s$  on the interval  $[\frac{15}{16}, 1]$ . Consider first the intervals  $[0, \frac{1}{4}]$  and  $[\frac{3}{4}, 1]$ . Because these intervals are well separated, the smoothness of  $\phi$  ensures that we can find a good polynomial approximation  $q_1$  to  $s_{[0, \frac{1}{4}]}$  on  $[\frac{3}{4}, 1]$ . Similarly we can find a good approximation  $q_2$  to  $s_{[\frac{1}{4}, \frac{3}{4}]}$  on  $[\frac{3}{4}, 1]$ . Continuing in this manner we find polynomial approximations:  $q_3$  approximating  $s_{[\frac{1}{8}, \frac{5}{8}]}$  on  $[\frac{7}{8}, 1]$ ,  $q_4$  approximating  $s_{[\frac{5}{8}, \frac{6}{8}]}$  on  $[\frac{7}{8}, 1]$ ,  $q_5$  approximating  $s_{[\frac{12}{16}, \frac{13}{16}]}$  on  $[\frac{15}{16}, 1]$ , and  $q_6$  approximating  $s_{[\frac{13}{16}, \frac{14}{16}]}$  on  $[\frac{15}{16}, 1]$ .

The sum  $q(x) = \sum_{j=1}^6 q_j(x)$  is now a polynomial approximation to  $s_{[0, \frac{14}{16}]}$  on the interval  $[\frac{15}{16}, 1]$ . Hence we can approximate  $s$  on  $[\frac{15}{16}, 1]$  by  $q + s_{[\frac{14}{16}, 1]}$ . Here, the polynomial  $q$  approximates the contribution to  $s(x)$  over  $[\frac{15}{16}, 1]$  of centers far from  $[\frac{15}{16}, 1]$ , while  $s_{[\frac{14}{16}, 1]}$  represents the contribution from nearby centers and is calculated directly. If the number of centers in  $[0, \frac{14}{16}]$  is large compared with the degree of  $q$ , then evaluation of  $q + s_{[\frac{14}{16}, 1]}$  will be much quicker than direct evaluation of  $s$ .

The remainder of this paper organizes and extends the ideas underlying Example 1 into efficient procedures for both the matrix–vector product and the general evaluation tasks.

An initial implementation of the evaluation version of the algorithm has been applied to a number of test problems. The results are summarized in Table 1 below. The numbers in the body of the table represent the time in seconds to perform the specified task on a Sun SPARC 2. Numbers in braces denote a power of 10 (e.g.,  $2.43(-1)$  represents  $2.43 \times 10^{-1}$ ). These timings were made by making several runs on an otherwise lightly loaded machine and taking the minimum elapsed time over all runs. In every case the direct time is the minimum over a greater number of runs than is the time for the algorithm. In these test problems both the centers and the evaluation points were uniformly distributed on  $[0, 1]$ . Also, the coefficients of each translate of  $\phi$  were chosen to be 1 as this corresponds to a worst-case error bound. The accuracy referred to in the table is an upper bound on the observed infinity norm error between the approximation used in fast evaluation and the true radial basis function  $s$ .

The table shows that even this preliminary implementation of the algorithm has several attractive features. First, crossover to the regime where the algorithm is faster than direct evaluation occurs for a relatively small value of  $N$ , certainly small enough for the algorithm to be of practical interest. Second, the algorithm can be orders of magnitude faster than direct evaluation. Third, the timings display linear (or very slightly faster) growth as a function of the number of centers  $N$ . This behavior is consistent with the bounds on operation counts given in section 7.

**3. Moment expansions.** Because the algorithm we propose here is based on polynomial approximations, we introduce some notation and results to speed their manipulation. First, it will be most convenient to express polynomials not as sums of monomials but rather as sums of the normalized monomials

$$(3.1) \quad V_j(x) := \frac{x^j}{j!}.$$

TABLE 1  
Time in seconds to complete an evaluation task.

$\phi$	# Centers	# Evaluation points	Direct evaluation	Fast method with accuracy		
				$10^{-4}$	$10^{-7}$	$10^{-10}$
Gaussian $e^{-10x^2}$	200	200	2.43(-1)	4.08(-2)	4.49(-2)	5.71(-2)
	400	400	1.01	7.63(-2)	8.81(-2)	1.14(-1)
	800	800	3.93	1.43(-1)	1.83(-1)	2.06(-1)
	1600	1600	1.58(1)	3.20(-1)	3.54(-1)	4.15(-1)
	3200	3200	6.36(1)	5.68(-1)	6.97(-1)	8.15(-1)
	6400	6400	2.63(2)	1.18	1.48	1.65
Gaussian $e^{-10x^2}$	200	2000	2.43	1.10(-1)	1.28(-1)	1.56(-1)
	400	4000	9.86	2.22(-1)	2.59(-1)	3.16(-1)
	800	8000	3.98(1)	4.53(-1)	5.30(-1)	5.76(-1)
	1600	16000	1.60(2)	8.90(-1)	1.02	1.24
	3200	32000	6.38(2)	1.70	2.11	2.43
	6400	64000	2.63(3)	3.86	4.23	4.60
Multiquadric $\sqrt{x^2 + 0.001}$	200	200	1.20(-1)	2.70(-2)	3.45(-2)	4.70(-2)
	400	400	4.91(-1)	5.25(-2)	6.90(-2)	8.77(-2)
	800	800	1.95	1.04(-1)	1.42(-1)	1.84(-1)
	1600	1600	7.93	2.30(-1)	2.76(-1)	3.54(-1)
	3200	3200	3.22(1)	4.45(-1)	5.61(-1)	7.10(-1)
	6400	6400	1.42(2)	8.96(-1)	1.14	1.38
Multiquadric $\sqrt{x^2 + 0.001}$	200	2000	1.20	8.05(-2)	1.10(-1)	1.36(-1)
	400	4000	4.87	1.62(-1)	2.24(-1)	2.67(-1)
	800	8000	1.99(1)	3.38(-1)	4.27(-1)	5.20(-1)
	1600	16000	7.99(1)	6.88(-1)	8.36(-1)	1.04
	3200	32000	3.23(2)	1.40	1.72	2.06
	6400	64000	1.42(3)	3.11	3.50	4.13
Thin-plate spline $x^2 \ln x $	200	200	2.42(-1)	4.40(-2)	4.86(-2)	6.05(-2)
	400	400	9.69(-1)	8.51(-2)	9.74(-2)	1.22(-1)
	800	800	3.89	1.61(-1)	2.05(-1)	2.46(-1)
	1600	1600	1.57(1)	3.71(-1)	3.90(-1)	4.87(-1)
	3200	3200	6.36(1)	6.57(-1)	7.98(-1)	9.69(-1)
	6400	6400	2.62(2)	1.39	1.71	2.03
Thin-plate spline $x^2 \ln x $	200	2000	2.43	1.14(-1)	1.40(-1)	1.78(-1)
	400	4000	9.73	2.29(-1)	2.80(-1)	3.56(-1)
	800	8000	3.94(1)	4.70(-1)	5.65(-1)	6.80(-1)
	1600	16000	1.58(2)	9.39(-1)	1.13	1.44
	3200	32000	6.35(2)	2.02	2.48	3.08
	6400	64000	2.62(3)	5.02	5.47	6.40

Next, given the points  $x_1, \dots, x_M \in \mathbf{R}$  and the corresponding weights  $d_1, \dots, d_M$ ,  $\sigma_{t,j}$  will denote the  $j$ th normalized moment of the data about the point  $t$ ,

$$(3.2) \quad \sigma_{t,j} := \sum_{m=1}^M d_m V_j(t - x_m) .$$

This simple notation will make apparent convolution structures which would otherwise be obscured. As a result several tasks which appear to require  $\mathcal{O}(k^2)$  flops will be identified as requiring only  $\mathcal{O}(k \log k)$  flops, where  $k$  is the degree of polynomial involved.<sup>1</sup> Unfortunately this attractive theoretical property probably has little prac-

<sup>1</sup>Greengard and Rokhlin [12] identified such convolution structures in the fast multipole algorithms for potential problems.

tical impact as  $k$  is generally so small that direct calculation is likely to be competitive with convolution by means of the FFT. The next two results show how the normalized moments can be used to efficiently manipulate polynomial approximations to radial basis functions. The first lemma gives an expression for the polynomial approximation to the far field influence of a sum of basic functions clustered about the origin in terms of these moments.

LEMMA 1. *Let  $b, c,$  and  $\epsilon > 0.$  Let  $|t| > b + c$  and  $\phi(\cdot)$  be a function in  $C[t - (b + c), t + (b + c)].$  Let*

$$q(x) = \sum_{j=0}^k a_j V_j(x - t)$$

*be a polynomial of degree  $k$  such that  $\|\phi(\cdot) - q\|_{L^\infty[t-(b+c), t+(b+c)]} \leq \epsilon.$  Given centers  $x_1, \dots, x_M$  with  $|x_m| \leq b$  for  $1 \leq m \leq M$  and weights  $d_1, \dots, d_M,$  let the corresponding radial basis function*

$$s(x) = \sum_{m=1}^M d_m \phi(|x - x_m|)$$

*be approximated by*

$$s_1(x) = \sum_{m=1}^M d_m q(x - x_m) .$$

*Then  $\|s - s_1\|_{L^\infty[t-c, t+c]} \leq \epsilon \|\mathbf{d}\|_1.$  Moreover,*

$$s_1(x) = \sum_{\ell=0}^k b_\ell V_\ell(x - t) ,$$

*where*

$$b_\ell = \sum_{j=\ell}^k a_j \sigma_{0, j-\ell} .$$

*Proof.* If  $|x - t| \leq c$  then for  $1 \leq m \leq M$  we have that  $|(x - x_m) - t| \leq |x - t| + |x_m| \leq (b + c).$  This shows all but the expression for  $s_1$  in terms of the normalized moments of the data. To see the latter write

$$\begin{aligned} s_1(x) &= \sum_{m=1}^M d_m q(x - x_m) \\ &= \sum_{m=1}^M d_m \sum_{j=0}^k a_j V_j(x - x_m - t) \\ &= \sum_{m=1}^M d_m \left\{ \sum_{j=0}^k \frac{a_j}{j!} ((x - t) - x_m)^j \right\} \\ &= \sum_{m=1}^M d_m \left\{ \sum_{j=0}^k \frac{a_j}{j!} \sum_{\ell=0}^j \binom{j}{\ell} (x - t)^\ell (-x_m)^{j-\ell} \right\} \end{aligned}$$

$$\begin{aligned}
 &= \sum_{m=1}^M d_m \left\{ \sum_{j=0}^k a_j \sum_{\ell=0}^j \frac{(x-t)^\ell}{\ell!} \frac{(-x_m)^{j-\ell}}{(j-\ell)!} \right\} \\
 &= \sum_{j=0}^k a_j \left\{ \sum_{\ell=0}^j V_\ell(x-t) \sigma_{0,j-\ell} \right\} \\
 &= \sum_{\ell=0}^k \left\{ \sum_{j=\ell}^k a_j \sigma_{0,j-\ell} \right\} V_\ell(x-t). \quad \square
 \end{aligned}$$

The next lemma shows that shifted moments may be expressed as a convolution of moments about a given point. This result will be used in generating the moments corresponding to a larger panel of centers from the moments corresponding to subpanels. Note in particular that the moments can be shifted with a flop count depending only on the degree of the highest moment involved and not on the number of centers in the radial function.

LEMMA 2. Let  $x_1, x_2, \dots, x_M \in \mathbf{R}$  be given points and  $d_1, \dots, d_M$  be corresponding weights. Let  $\sigma_{u,j}$  be the  $j$ th normalized moment of the data about  $u$  defined in (3.2). Then for all  $u, v \in \mathbf{R}$  and  $j \in \mathbf{N}_0$ ,

$$\sigma_{u,j} = \sum_{\ell=0}^j V_\ell(u-v) \sigma_{v,j-\ell}.$$

*Proof.*

$$\begin{aligned}
 \sigma_{u,j} &= \frac{1}{j!} \sum_{m=1}^M d_m (u-x_m)^j \\
 &= \frac{1}{j!} \sum_{m=1}^M d_m (v-x_m+u-v)^j \\
 &= \frac{1}{j!} \sum_{m=1}^M d_m \sum_{\ell=0}^j \binom{j}{\ell} (u-v)^\ell (v-x_m)^{j-\ell} \\
 &= \sum_{m=1}^M d_m \sum_{\ell=0}^j V_\ell(u-v) V_{j-\ell}(v-x_m) \\
 &= \sum_{\ell=0}^j V_\ell(u-v) \sigma_{v,j-\ell}. \quad \square
 \end{aligned}$$

**4. Data structures.** All the fast algorithms make use of a hierarchical subdivision of the problem domain: this section details the data structures used to support these subdivisions. The algorithms adaptively divide the unit interval  $[0, 1]$  into a binary tree of subintervals or panels. One possible such subdivision is illustrated in Figure 1 below.

A subdivision, or tree,  $\mathcal{P}$  of panels  $P$  can be expressed concisely in terms of the following set-valued maps. Let  $P$  be a panel; then

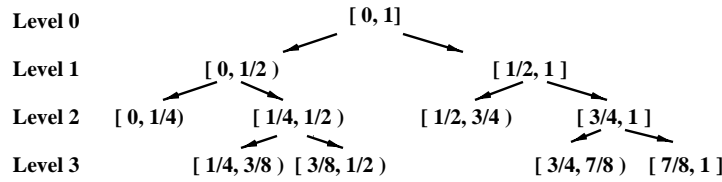


FIG. 1. A possible subdivision of  $[0, 1]$ .

- $pa(P) :=$  parent of  $P$ ,
- $\ell(P) :=$  level of  $P$   
 = number of steps down the tree from the root panel,  $[0, 1]$ , to  $P$ ,
- $c(P) :=$  children of  $P$  (if any),
- $p(P) := P$  and its peers (if any)  
 =  $P$  and panels immediately adjacent to and at the same level as  $P$ .

Some further notation will be needed to describe the algorithm. Namely,

- $\#P :=$  number of centers  $x_n$  in the panel  $P$ ,
- $s_P(\cdot) :=$  influence function of panel  $P$  = part of  $s(\cdot)$  due to centers in  $P$   
 =  $\sum_{n: x_n \in P} d_n \phi(|\cdot - x_n|)$ ,
- $t(P) :=$  center of panel  $P$ ,
- $m(P) :=$  vector of normalized moments  $\{\sigma_{t(P),j}\}_{j=0}^k$  of the data in  $P$   
 =  $\{\sum_{n: x_n \in P} d_n V_j(t(P) - x_n) : j = 0, 1, \dots, k\}$ ,
- $i\ell(P) :=$  interaction list of  $P$  =  $c(p(pa(P))) - p(P)$   
 = list of panels  $Q$  whose influence functions  $s_Q$  will be approximated on  $P$  by polynomials at level  $\ell(P)$  in the algorithm,
- $d(P) :=$  direct list of  $P$   
 = list of panels  $Q$  for which it is already known that the influence functions  $s_Q(\cdot)$  will be calculated directly at points  $x \in P$ ,
- $r_P :=$  polynomial approximation to the far field influencing panel  $P$   
 = polynomial approximation to  $(s - \sum_{Q \in (d(P) \cup p(P))} s_Q)$  on  $P$ ,
- $q_\ell :=$  polynomial approximation to  $\phi$  used at level  $\ell$ .

Finally  $L$  will denote the maximum depth in  $\mathcal{P}$ , where the root of the tree, the panel  $[0, 1]$ , is regarded as being at level 0.  $\ell$  will denote an arbitrary level in  $\mathcal{P}$ .

**5. Simplified overview of the algorithm.** Given a specified accuracy  $\epsilon > 0$ , a vector of centers  $\{x_n\}_{n=1}^N \subset [0, 1]$ , and a corresponding vector  $\{d_n\}_{n=1}^N$  of weights, we now present an algorithm to calculate  $s(x) = \sum_n d_n \phi(|x - x_n|)$  to within accuracy  $\epsilon \|d\|_1$ . The algorithm is adaptive, both with respect to different choices of  $\phi$  and also with respect to nonuniform distributions of centers. We will prove that the algorithm is efficient when  $\phi$  is a function of a particular type (to be described in the next section), and the distribution of centers is approximately uniform. We anticipate that it is also efficient for functions of much less smoothness and for fairly nonuniform distributions of centers.

We first present an overview of a simplified version of the algorithm. For the sake of readability this version presents only the simplest possible strategies for many steps such as deciding whether or not to subdivide a panel. More elaborate strategies, which should result in a more efficient algorithm, will be discussed later.

**Construction of the binary tree  $\mathcal{P}$ .**

**Step 1** Initialize the binary tree subdivision of  $[0, 1]$  by dividing panels in half down to level 2. Form a polynomial approximation to  $\phi(| \cdot |)$  to use at level 2.

**Step 2** For level  $\ell = 2, 3, \dots$

Adaptively determine a polynomial approximation  $q_{\ell+1}$ , to  $\phi(| \cdot |)$  for use at level  $\ell + 1$ , and from this determine the maximum desirable number of centers  $n_\ell$  per childless panel at level  $\ell$ .

For each panel  $P$  in level  $\ell$  form its interaction list.

If  $P$  contains  $n_\ell$  or more centers then form its children  $c(P)$  and add them to the tree. The children inherit the direct list of  $P$ .<sup>2</sup> Otherwise add  $P$  to the direct list of panels in  $p(P)$  and add  $p(P)$  to the direct list of  $P$ .

**Upward sweep generating moments summarizing interior data.**

**Step 3** For each childless panel  $P$ , calculate the normalized moments  $m(P)$  of the weights and centers in  $P$  about the panel center  $t(P)$ .

**Step 4** Sweep up the tree using the shifting rule of Lemma 2 to generate the normalized moments for each panel  $P$  from the normalized moments of its children.

**Downward sweep generating polynomial approximations summarizing far field contributions.**

**Step 5** For each panel  $P$  at level 1 initialize the polynomial  $r_P$  to zero.

**Step 6** Starting at level 2 sweep down the tree. For each panel  $P$  form the associated polynomial approximation  $r_P$  by adding to  $r_{pa(P)}$  polynomial approximations to the influence functions of panels on the interaction list of  $P$  when evaluated within  $P$ .

**Evaluation sweep.**

**Calculation of an approximate value for  $s(x)$**

**Step 1** Identify the unique childless panel  $P$  containing  $x$ .

**Step 2** Evaluate the polynomial approximation  $r_P(x)$ . Add to this the direct evaluation of the contributions  $s_Q(x)$  from panels  $Q$  on the direct list of  $P$ .

**6. Adaptive polynomial approximation.** In this section we outline an adaptive method for constructing the polynomial approximations to  $\phi$  required at various stages in the algorithm. The method is based on interpolating  $\phi$  at the zeros of shifted Chebyshev polynomials. The results of this and the next section also establish bounds on the error of these approximations, which in turn give asymptotic bounds on the algorithm's operation count.

In practice the user is unlikely to know or wish to compute the constants appearing in these bounds (or indeed in the bounds associated with any scheme for constructing polynomial approximations). Therefore, he or she cannot use them explicitly to identify the optimal degree for an approximation and so optimize operation counts. Indeed one reason for introducing an adaptive algorithm here is to remove the need for such a priori calculations. What the bounds do establish is that the adaptive algorithm produces nearly optimal approximations that automatically make full use of any smoothness in  $\phi$ . Moreover they show that for most basic functions of interest the approximations converge sufficiently rapidly that only  $\mathcal{O}(|\log \epsilon|)$  terms are needed to achieve the desired accuracy  $\epsilon$ .

<sup>2</sup>In matrix–vector product versions of the algorithm empty children will not be added to the tree.

We first briefly review the well-known approximation properties of interpolation at the zeros (or the extrema) of Chebyshev polynomials. It is well known that if  $F_k$  is a finite-dimensional subspace of a normed linear space  $F$  and  $L_k$  is a bounded linear projection onto  $F_k$ , then

$$\|\phi - L_k(\phi)\| \leq (1 + \|L_k\|)E_k(\phi),$$

where  $E_k(\phi) = \min_{f \in F_k} \|\phi - f\|$  is the error in the best approximation to  $\phi$  from  $F_k$ . We apply this result in the case where  $F_k = \pi_k$  (the space of algebraic polynomials of degree at most  $k$ ),  $F$  is the space  $C[a, b]$  equipped with the uniform norm  $\|g\|_{L^\infty[a,b]} = \max_{x \in [a,b]} |g(x)|$ , and  $L_k$  is interpolation at the zeros of the shifted Chebyshev polynomial  $W_{k+1}$  of degree  $k + 1$ . Using the estimate (see Rivlin [17, p. 18])

$$\frac{2}{\pi} \log k + \frac{2}{\pi} \left( \log \frac{8}{\pi} + \gamma \right) < \|L_k\| \leq 1 + \frac{2}{\pi} \log k,$$

where  $\gamma$  is Euler's constant, we see that

$$(6.1) \quad \|\phi - L_k(\phi)\|_{L^\infty[a,b]} \leq \left( 2 + \frac{2}{\pi} \log k \right) E_k(\phi).$$

Thus the uniform error in approximation by  $L_k(\phi)$  differs from the error  $E_k(\phi)$  in best uniform polynomial approximation by a multiplicative factor which is less than 4.93... for  $k \leq 100$  and grows only as  $\log k$ . In the next section we show that for functions of interest  $E_k(\phi)$  converges geometrically to zero with increasing  $k$ . These decay rates easily dominate the  $\log k$  growth in (6.1), so, as the analysis leading up to (7.5) shows,  $L_k(\phi)$  shares the same asymptotic error bounds and operation counts as those for the best uniform polynomial approximation. Moreover, as the degree of the polynomial approximation is likely to be less than 20 rather than close to 100,  $L_{k+1}$  will yield approximations that are, for practical purposes, as good as the best approximations from  $\pi_k$ .

Given these results, suitable approximations to  $\phi$  can be generated by the following adaptive procedure. First choose an initial value of  $k$  and approximate  $\phi$  with the operator  $L_k$ . Estimate the error by evaluating  $|\phi(x) - L_k\phi(x)|$  at the extrema of  $W_{k+1}$ . If these values are all smaller than  $\epsilon/2$ , then reduce  $k$ ; if they are larger, increase  $k$ . Repeat the procedure until  $L_k$  gives estimated accuracy less than  $\epsilon/2$  but  $L_{k-1}$  does not. The operations counts derived below are based on the assumption that such a procedure is used to find approximations to  $\phi$ .

In implementing the algorithm, however, we have generated the required approximations to  $\phi$  by Chebyshev economization rather than by repeated interpolation and error estimation. In this procedure an initial excessively accurate approximation  $q_{k+\ell}(x) = \sum_{j=0}^{k+\ell} a_j W_j(x)$  is first computed by interpolating at the zeros of  $W_{k+\ell+1}$ . An associated error estimate  $\epsilon_{k+\ell}$  is obtained by evaluating the residual at the extrema of  $W_{k+\ell+1}$ . Next the error in the lower-degree approximation  $q_k(x) = \sum_{j=0}^k a_j W_j(x)$  is estimated by  $\epsilon_k = \epsilon_{k+\ell} + \sum_{j=k+1}^{k+\ell} |a_j|$ . In our experience constructing  $q_{k+\ell}(x)$  followed by  $\ell > 0$  steps of economization, where  $\ell$  is chosen so that  $\epsilon_k \approx \epsilon$  almost always yields a better approximation than direct interpolation at the zeros of  $W_{k+1}$ . This experience, together with the savings made by using economization rather than repeated interpolations to determine the minimal degree, motivated our implementation decision.

**7. Functions to which the algorithm is well suited.** In the first part of this section we introduce classes  $Z(M, \alpha)$  of smooth functions to which the algorithm of section 5 is well suited. In particular, we prove that if the basic function  $\phi$  lies in some class  $Z(M, \alpha)$  with  $0 < \alpha < \pi/2$  and the centers are reasonably uniformly distributed, then the algorithm described above will evaluate  $s(x) = \sum_{n=1}^N d_n \phi(|x - x_n|)$  to within accuracy  $\epsilon \|\mathbf{d}\|_1$ ,  $0 < \epsilon < 0.5$  at an incremental cost of  $\mathcal{O}(|\log \epsilon|)$  flops after a setup cost of  $\mathcal{O}(N \log N + N |\log \epsilon| \log |\log \epsilon|)$ . While the class definitions are tailored to allow easy derivation of this bound, the classes are surprisingly rich: they include, for instance, Gaussians  $\phi(x) = e^{-\lambda^2 x^2/2}$ , generalized multiquadrics  $\phi(x) = (x^2 + \lambda^2)^{(2k-1)/2}$ , and thin-plate splines  $\phi(x) = x^2 \log x$ . Moreover, the last part of the section presents simple extensions of the definition to include functions which are unbounded near zero such as the logarithm.

The classes  $Z(M, \alpha)$  essentially consist of functions that are analytic and bounded in some wedge of the complex plane that contains the interval  $(0, 1]$ . More precisely, for any  $0 < \alpha < \pi/2$ , let  $D(\alpha)$  be the closed diamond in the complex plane bounded by the line segments cutting the positive real axis at angles  $\pm\alpha$  at 0 and  $\pi \pm \alpha$  at 2. Then for a given  $M > 0$  and  $0 < \alpha < \pi/2$ ,  $Z(M, \alpha)$  is the set of all functions  $\phi$  that are real valued for  $z \in [0, 2]$ , analytic on the interior of  $D(\alpha)$ , and continuous and bounded by  $M$  on  $D(\alpha)$ .

The key to the suitability of these classes is that the error in approximating a function  $\phi \in Z(M, \alpha)$  from  $\pi_k$  on a set of subintervals of  $(0, 1]$ , each of which is separated from 0 by an amount proportionate to its length, can be bounded uniformly. Thus there is a uniform upper bound on the degree of the polynomials needed to approximate the interactions at all scales in the algorithm outlined in the previous section. Moreover, this upper bound establishes at least geometric convergence of the approximations to  $\phi$ . More precisely we have the following lemma.

LEMMA 3. *Let  $0 < \alpha < \pi/2$ ,  $\phi \in Z(M, \alpha)$ , and  $0 < \beta < 1$ . Let  $K_{\alpha, \beta}$  be the line segment joining the points  $(-1/\beta, 0)$  and  $(0, (\tan \alpha)/\beta)$  in the complex plane. Let  $\rho > 1$  be that solution of*

$$(7.1) \quad \rho + \rho^{-1} = \min_{w \in K_{\alpha, \beta}} (|w - 1| + |w + 1|)$$

*that exceeds 1. Then for all  $k \in \mathbf{N}$  and  $0 < t \leq 1$  the best uniform approximation  $p_k$  to  $\phi$  on  $[(1 - \beta)t, (1 + \beta)t]$  from  $\pi_k$  satisfies*

$$(7.2) \quad \|\phi - p_k\|_{L^\infty[(1-\beta)t, (1+\beta)t]} \leq \frac{2M}{\rho - 1} \rho^{-k} .$$

*Proof.* Let  $D(\alpha)$  be defined in the  $x$ - $y$  plane associated with the complex variable  $z$ . Furthermore, for  $0 < t \leq 1$  let  $D(\alpha, t)$  be the closed diamond in the  $z$  plane bounded by the line segments cutting the positive real axis at angles  $\pm\alpha$  at 0 and  $\pi \pm \alpha$  at  $2t$ , and let  $M_t$  be the maximum of  $\phi$  on  $D(\alpha, t)$ . Note that as  $t \leq 1$ ,  $D(\alpha, t) \subset D(\alpha)$ , and  $M_t \leq M$ .

Next for a given  $0 < t \leq 1$  and  $0 < \beta < 1$  we define a transformation to a new plane with coordinates  $\tilde{x}$ - $\tilde{y}$  associated with the complex variable  $w$  by

$$\tilde{x} = (x - t)/\beta t, \quad \tilde{y} = y/\beta t .$$

This transformation takes the diamond  $D(\alpha, t)$  into the shifted and scaled diamond  $\tilde{D}(\alpha, \beta)$  in the  $\tilde{x}$ - $\tilde{y}$  plane that is bounded by the line segments cutting the real axis

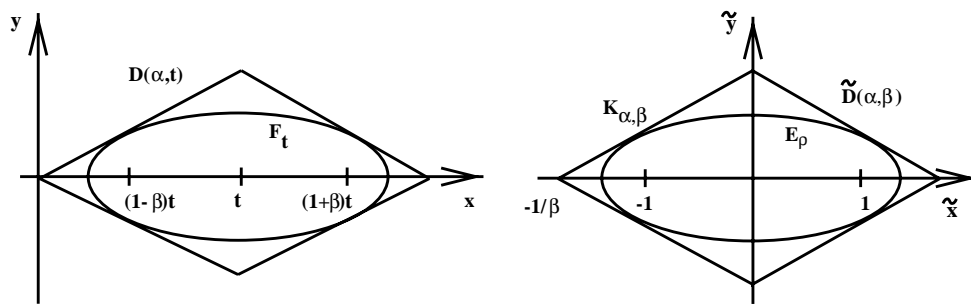


FIG. 2. Original and transformed intervals and regions of analyticity.

at angles  $\pm\alpha$  at  $(-1/\beta, 0)$  and  $\pi \pm \alpha$  at  $(1/\beta, 0)$ . Furthermore, it takes the interval  $[(1 - \beta)t, (1 + \beta)t]$  on the  $x$  axis into the interval  $[-1, 1]$  on the  $\tilde{x}$  axis. Thus the image  $\tilde{D}(\alpha, \beta)$  of  $D(\alpha, t)$  is now independent of  $t$  and has the line segment  $K_{\alpha, \beta}$  as one of its four sides. The situation is illustrated in Figure 2 above.

We now use some results of Bernstein on the approximation of analytic functions by polynomials. Following Lorentz [15, pp. 42 and 76], for each  $\rho > 1$  let  $E_\rho$  be the ellipse

$$(7.3) \quad E_\rho = \left\{ (\tilde{x}(\theta), \tilde{y}(\theta)) = \left( \frac{1}{2}(\rho + \rho^{-1}) \cos \theta, \frac{1}{2}(\rho - \rho^{-1}) \sin \theta \right) : 0 \leq \theta \leq 2\pi \right\}.$$

Some important properties of this family of ellipses are that the foci of  $E_\rho$  are at  $(\pm 1, 0)$ , that if  $\rho_1 > \rho_2 > 1$  then  $E_{\rho_2}$  lies strictly within  $E_{\rho_1}$ , and that as  $\rho \downarrow 1$   $E_\rho$  collapses to the segment  $[-1, 1]$  on the real axis. Moreover, there is a unique maximal  $\rho > 1$  such that  $E_\rho$  is still contained in  $\tilde{D}(\alpha, \beta)$ , and for this  $\rho$  the corresponding  $E_\rho$  touches  $\partial\tilde{D}(\alpha, \beta)$  at four points, one of which lies on  $K_{\alpha, \beta}$ . Hence this  $\rho$  is given by (7.1).

We now apply the preceding geometry to the problem of approximating  $\phi$  on the interval  $[(1 - \beta)t, (1 + \beta)t]$ . Let the function  $\psi(w)$  be defined by

$$\psi(w) = \phi(z) \quad \text{for} \quad w = (z - t)/(\beta t).$$

Then  $\psi$  is analytic on the interior of  $\tilde{D}(\alpha, \beta)$  and is continuous and bounded by  $M_t$  on  $\tilde{D}(\alpha, \beta)$ . Thus if  $q_k$  is the polynomial of best uniform approximation to  $\psi$  from  $\pi_k$  on  $[-1, 1]$  then estimate (6) of Lorentz [15, page 78] establishes that

$$(7.4) \quad \|\psi - q_k\|_{L^\infty[-1, 1]} \leq \frac{2M_t}{\rho - 1} \rho^{-k}.$$

Hence if we define  $p_k(z) = q_k(w)$  where  $w = (z - \beta t)/(\beta t)$ , then  $p_k$  is also a polynomial of degree  $k$  and

$$\|\phi - p_k\|_{L^\infty[(1-\beta)t, (1+\beta)t]} = \|\psi - q_k\|_{L^\infty[-1, 1]} \leq \frac{2M_t}{\rho - 1} \rho^{-k} \leq \frac{2M}{\rho - 1} \rho^{-k}.$$

This concludes the proof.  $\square$

The lemma establishes a uniform upper bound on the convergence rate of the polynomial approximations to all basic functions  $\phi \in Z(M, \alpha)$  at all scales. We now show that this in turn bounds the number of coefficients that need to be manipulated

at each level in the algorithm sketched in section 5 and so bounds the incremental cost per evaluation and the total work.

We begin by establishing that the incremental cost of evaluating  $s(x)$  to within accuracy  $\epsilon \|\mathbf{d}\|_1$  is  $\mathcal{O}(|\log \epsilon|)$ . To see this, first recall that the algorithm does not use polynomials of best approximation but rather interpolants at the zeros of Chebyshev polynomials. As pointed out in section 6, these interpolants have a uniform error which is at worst larger than the error of the best approximation by a multiplicative factor of order  $\mathcal{O}(1 + \log k)$ . Now suppose that for some  $0 < \alpha < \pi/2$  and  $M > 0$ ,  $\phi \in Z(M, \alpha)$ . Also suppose  $0 < \beta < 1$  is fixed. Then by the lemma there exists a  $\gamma$ ,  $0 < \gamma < 1$  such that the error in best uniform approximation to  $\phi$  from  $\pi_k$  on  $[(1 - \beta)t, (1 + \beta)t]$  is guaranteed to be  $\mathcal{O}(\gamma^k)$  for all  $0 < t \leq 1$  and  $k \in \mathbf{N}$ . Hence for all  $\delta > 0$  the uniform error in the interpolant used in the algorithm is

$$\mathcal{O}(1 + \log k)\mathcal{O}(\gamma^k) = \mathcal{O}((\gamma + \delta)^k)$$

for all  $0 < t < 1$  and  $k \in \mathbf{N}$ . It follows that the desired precision  $\epsilon$  in approximating  $\phi$  is achieved with polynomials of degree

$$(7.5) \quad k = \mathcal{O}\left(1 + \left\lceil \frac{\log \epsilon}{\log(\gamma + \delta)} \right\rceil\right).$$

Now, ignoring all sorting and setup costs, a single extra evaluation of  $s(x)$  involves evaluating the near field directly plus one evaluation of the polynomial representing the far field. Hence, as long as the maximum number of near-field centers is also  $\mathcal{O}(k)$ , then the extra approximate evaluation requires only  $\mathcal{O}(k) = \mathcal{O}(|\log \epsilon|)$  flops.

We now turn to an analysis of the number of operations required in the setup phase. Again we assume that  $\phi$  is in  $Z(M, \alpha)$  for some  $0 < \alpha < \pi/2$  and that the centers are approximately uniformly distributed. Then, as above, accuracy  $\epsilon$  in approximating  $\phi$  can be achieved with polynomials of degree  $k$  given by (7.5). In particular this bound on the degree is uniform across all levels in the tree. The panel splitting threshold value will therefore be set to some multiple of  $k$  and will be at least 2. Then from the uniform distribution assumption the lowest level in the tree will be  $\mathcal{O}(\log_2(N))$  and the number of panels in the tree will be  $\mathcal{O}(N)$ . Thus the work in each step in the setup phase can be bounded as follows.

- Sorting the centers and then splitting them into panels can be performed by one sweep through the data at each level and therefore takes  $\mathcal{O}(N \log N)$  comparisons.
- Forming the moments for childless nodes takes  $\mathcal{O}(k)$  flops per center and obtaining the moments of a parent panel from those of its children takes  $\mathcal{O}(k \log k)$  flops.<sup>3</sup> Thus forming all the moments takes  $\mathcal{O}(Nk \log k)$  flops.
- Calculating an  $\epsilon$  error polynomial approximation  $q_\ell$  to  $\phi$  for use at level  $\ell$  takes  $\mathcal{O}(k^3)$  operations. Since  $q_\ell$  need only be formed once per level, and not once per panel, forming all such polynomial approximations takes only  $\mathcal{O}(k^3 \log N)$  flops.
- Forming the polynomial approximations to the influence of panels at the same level and on the interaction list of a given panel takes  $\mathcal{O}(k \log k)$  flops. Also, recentering the polynomial  $r_{pa(P)}$  associated with a parent to initialize the polynomial  $r_P$  of a child takes  $\mathcal{O}(k \log k)$  flops. Hence generating all the polynomial approximations from the moments takes  $\mathcal{O}(Nk \log k)$  flops.

<sup>3</sup>The implementation that produced the timings of Table 1 used  $\mathcal{O}(k^2)$  rather than  $\mathcal{O}(k \log k)$  code fragments to shift moments and to form and shift polynomials.

Hence the dominant terms in the operation count for setup costs are

$$(7.6) \quad \mathcal{O}(N \log N + Nk \log k) = \mathcal{O}(N \log N + N |\log \epsilon| \log |\log \epsilon|).$$

In practice the costs associated with sorting and panel splitting are negligible compared with those of the other manipulations, so that the dominant cost is  $\mathcal{O}(N |\log \epsilon| \log |\log \epsilon|)$ .

Finally, for a matrix–vector evaluation, evaluating  $s$  at the  $N$  centers takes  $\mathcal{O}(Nk)$  further flops. Hence evaluating the single matrix–vector product  $A\mathbf{d}$  to within accuracy  $\epsilon \|\mathbf{d}\|_1$  has the same asymptotic costs as in (7.6). When performing many matrix–vector products corresponding to fixed centers but different weight vectors, only one full sort of the centers will be required. Hence the factor  $N \log N$  in estimate (7.6) will not contribute to the second and subsequent matrix–vector products. Hence the operation count per product reduces to  $\mathcal{O}(N |\log \epsilon| \log |\log \epsilon|)$ .

We now examine the tightness of the bounds derived above in a little more detail and consider some associated extensions of the definition of  $Z(M, \alpha)$  to cover a broader range of functions and get better estimates. It is clear that for each of the Gaussian, generalized multiquadric, and thin-plate splines there exist constants  $M^{(\alpha)}$  such that these functions are in  $Z(M^{(\alpha)}, \alpha)$  for each  $0 < \alpha < \pi/2$ . Lemma 3, therefore, shows that the polynomial approximations to  $\phi$  on the intervals  $[(1 - \beta)t, (1 + \beta)t]$  converge geometrically with convergence factor  $1/\rho$  decreasing in  $\alpha$  and increasing in  $\beta$ . However,  $M^{(\alpha)}$  also increases with  $\alpha$ ; moreover, the appearance of  $M_t$  in equation (7.4) of the proof of Lemma 3 indicates that for small  $t$ ,  $M^{(\alpha)}$  may be overly conservative. These observations raise questions such as what is the best choice of  $\alpha$  and whether significant reduction in degree is possible for small  $t$ .

For the Gaussian especially attractive bounds hold. To see this note that  $\phi(x) = e^{-\lambda^2 x^2/2}$  is in the class  $Z(1, \pi/4)$  for all  $\lambda \in \mathbf{R}$ . This follows since for any  $z = re^{i\theta}$  with  $|\theta| \leq \pi/4$ ,  $\Re(-z^2) \leq 0$ , and so  $|e^{-\lambda^2 z^2/2}| \leq 1$ . Moreover, we have that  $M_t = M = 1$  for all  $0 < t \leq 1$ . Thus for Gaussian radial basis functions the degree  $k$  of the approximating polynomials required in the algorithm can be bounded independently of the level in the tree and of the scale parameter  $\lambda$ .<sup>4</sup>

We now briefly consider two useful extensions of the class definitions. The first extension gives a tighter analysis of the case of functions  $\phi(z)$  real on  $[0, 2] \subset \mathbf{R}$ , analytic in the strip  $0 < \Re(z) < 2$ , and continuous at  $(0, 0)$  and  $(2, 0)$ . This function class is in a sense the limit of  $Z(M, \alpha)$  as  $\alpha \rightarrow (\pi/2)^-$  and includes the Gaussian, generalized multiquadric and thin-plate spline. Appendix A shows that the best approximations to functions in this class on the interval  $[(1 - \beta)t, (1 + \beta)t]$  satisfy estimates of the form (7.2) but with  $\rho$  now given by

$$(7.7) \quad \rho = \frac{1}{\beta} \left( 1 + \sqrt{1 - \beta^2} \right),$$

which is the limit as  $\alpha$  goes to  $(\pi/2)^-$  of (7.1). The corresponding value of  $M$  is the maximum of  $|\phi(z)|$  on the ellipse similar to  $E_\rho$  with major axis from  $(0, 0)$  to  $(2, 0)$ .

For the second extension, we consider functions  $\phi(z)$  that are analytic in some open diamond  $D(\alpha)$ ,  $0 < \alpha < \pi/2$  and are continuous at all parts of the closed diamond, except possibly the origin. We will say  $\phi \in Z(M(r), \alpha)$  if  $M$  is a function continuous on the interval  $(0, \infty)$  such that  $|\phi(z)| \leq M(|z|)$  everywhere on the open

---

<sup>4</sup>Greengard and Strain [13] and Strain [18] have already presented a special purpose fast evaluation algorithm for sums of Gaussians that exploits the exponential decay of the basic function.

diamond. Our primary interest is in the behavior of  $\phi(z)$  near the origin; therefore, we shall assume that  $M(r) = cr^\nu$  as  $r \rightarrow 0^+$  for some real  $\nu$ . Examples of functions in this extended family are  $\phi(x) = 1/x$ , which is a member of  $Z(1/r, \alpha)$  for all  $0 < \alpha < \pi/2$ , and  $\phi(x) = \log x$  which is a member of some class  $Z(c_\nu r^\nu, \alpha)$  for every  $\nu < 0$  and  $0 < \alpha < \pi/2$ .

Since  $M(r)$  is continuous for  $r > 0$ , the proof of Lemma 3 can still be applied to establish geometric convergence rates for polynomial approximations to functions in this extended family. The only change is that the constant  $M_t$  in equation (7.2) must be replaced by

$$M_{\rho,t} = \max_{\frac{z-t}{\beta t} \in E_\rho} M(|z|).$$

For fixed  $\rho$  and  $\beta$  the ellipse  $E_\rho$  in the scaled setting corresponds to values  $z$  in the unscaled setting satisfying an inequality of the form

$$at \leq |z| \leq At$$

for some  $0 < a < A$  in  $\mathbf{R}$ . Hence for fixed  $\rho$  and  $\beta$ ,  $M_{\rho,t}$  will scale as  $t^\nu$  as a function of  $t$ ; in particular, we suppose that  $M_{\rho,t} \leq Ct^\nu$ . Thus the degree  $k$  of the polynomial approximating  $\phi$  to within precision  $\epsilon$  now also depends explicitly on the center of approximation  $t$ . Arguments similar to those used to obtain equation (7.5) show that  $k$  is now given by

$$(7.8) \quad k = \mathcal{O}\left(1 + \frac{|\log \epsilon| + \log C - \nu |\log t|}{\log(\gamma + \delta)}\right).$$

A closer examination of the terms in (7.8) gives insight into factors influencing the operation count of the algorithm that have not been explicitly acknowledged elsewhere. First, if centers are reasonably uniformly distributed, then  $t$  will range from  $\mathcal{O}(1)$  to  $\mathcal{O}(1/N)$ . If  $\nu$  is negative, (7.8) shows that the bound on the degree required for adequate approximation is now an explicit function of  $N$ , with growth rate  $\mathcal{O}(\log N)$  (a growth rate that will increase with increasing nonuniformity in the distribution of centers). Applying the analysis of Appendix B to the approximation of a scale invariant function such as  $\phi(x) = 1/|x|$  shows that in practice  $k$  can indeed achieve this worst-case growth rate.

If  $\nu$  is positive there no longer appears to be a problem; the degree of the approximating polynomials will decrease at lower levels. Nevertheless, even in this case  $k$  can still depend on  $N$ . The reason is that the  $\log C$  term in (7.8) implicitly includes any scale factors that arose in rescaling the problem to  $[0, 1]$ ; these scale factors can be proportional to  $N$  or a power thereof. A proper analysis of this dependence should be done in the context of bounding the relative error  $\|s - \tilde{s}\|/\|s\|$  rather than the absolute error  $\|s - \tilde{s}\|$ . This is beyond the scope of this paper, but is noted here as it can have considerable practical impact.

**8. Detailed description of the algorithm.** We now describe the algorithm in more detail. The algorithm consists of four stages. The initial stage determines the binary tree and associated data structures. In this stage at each level a decision must be made whether or not to continue subdivision of the tree; the algorithm presents some new strategies for this which are justified in the next section. The initial stage need only be fully computed once for any given basic function  $\phi$  and set of centers  $\{x_n\}$ .

The remaining three stages need to be recomputed for each new coefficient vector  $\mathbf{d}$ . The second stage efficiently computes the normalized moments for all panels in an upward sweep through the tree. The third stage computes the approximating polynomials in a downward sweep of the tree. Finally, the last stage evaluates the radial basis function at any given point. It is presented here as a separate stage since, while matrix–vector products can be effectively evaluated at the end of the third stage, general evaluation of  $s$  on, say, a dense grid of points requires a separate final sweep. Note also that in general evaluation of the structure of the tree depends on the ratio of the number of evaluation points to the number of centers, denoted here by  $b$ , as well as on the centers themselves.

**Construction of the binary tree  $\mathcal{P}$ .**

**Step 1** Initialize a binary tree subdivision of  $[0, 1]$  by dividing panels in half down to level 2.

Initialize all direct lists at level 1 to be empty.

Calculate an  $\epsilon$  error polynomial approximation  $q_2$  to  $\phi$  on the interval  $[\frac{1}{4}, 1]$ .

Set  $\ell = 2$  and  $h_2 = \text{degree}(q_2)$ .

**Step 2**

If there are panels at level  $\ell$

    Create interaction and peer lists.

    Initialize direct lists of panels by copying direct lists of parents.

    Calculate an  $\epsilon$  error polynomial approximation  $q_{\ell+1}$ , to  $\phi$  on the interval  $[2^{-(\ell+1)}, 4 \cdot 2^{-(\ell+1)}]$ .

    Set  $h_{\ell+1} = \max(h_\ell, \text{degree}(q_{\ell+1}))$ .

    Set  $n_\ell \approx 2h_{\ell+1}/\sqrt{bc_\phi}$ .

    For each panel  $P$  at level  $\ell$

        If  $\#P \geq n_\ell$  {Make  $P$  a parent}

            Subdivide  $P$  in half forming its two children (in the matrix–vector version do not form empty children).

        else { $P$  is childless}

            Add the peers of  $P$  to the direct list of  $P$ .

            Add  $P$  to the direct list of its peers.

        end

    end

    Set  $\ell = \ell + 1$ .

end { if }.

**Upward sweep generating moments.**

**Step 3** For each childless panel  $P$  calculate the moments  $m(P)$ .

**Step 4** Sweeping up the tree to level 2, generate the moments of parent panels. Do this by the most efficient direct calculation or by merging of moments of child panels using the shifting rule of Lemma 2.

**Downward sweep generating polynomials.**

**Step 5** For all panels  $P$  at level 1, initialize the polynomial  $r_P$  to zero.

**Step 6**

For level  $\ell = 2$  to  $L$

    For each panel  $P$  at level  $\ell$

        Initialize  $r_P$  by recentering the polynomial  $r_{pa(P)}$ .

        For each panel  $Q \in il(P)$

            Use Lemma 1, the polynomial  $q_\ell$ , and the moments  $m(Q)$  (or when both children of  $pa(Q)$  are in  $il(P)$ ,  $m(pa(Q))$ ) to add a

polynomial approximation to the influence of  $Q$ ,  $s_Q$  to the polynomial  $r_P$ .  
 end  
 end  
 end.

**Evaluation sweep.**

To approximate the value of  $s$  at the point  $u$ .

**Step 1** Identify the unique childless panel  $P$  containing  $u$ .

**Step 2** Set  $v = r_P(u)$ . For each panel  $Q$  in the direct list of  $P$  add  $s_Q(u)$  to  $v$ .

**9. Subdivision strategies.** Clearly the strategy that determines when a panel in the tree  $\mathcal{P}$  is subdivided has a major influence on the operation count of the algorithm. Almost all the termination criteria employed in fast evaluation algorithms to date have been of the general form: subdivide panel  $P$  if it contains  $n$  or more centers, where  $n$  is fixed and independent of the level. This section looks at this strategy in more detail. The first subsection shows how the threshold value  $n_\ell$  can be adaptively chosen to minimize the flop count. The second subsection shows that broadening the threshold to include points in neighboring panels ensures that the number of direct evaluations at each point can be bounded independently of the number of centers.

**9.1. An approximate analysis of the single panel center count subdivision strategy.** As the algorithm of the previous section adaptively determines the degree of the approximating polynomials at each level,  $n_\ell$  cannot be chosen a priori to minimize the total operation count. Therefore, we present here a strategy that chooses  $n_\ell$  adaptively at each level  $\ell$ . The strategy is based on comparing an estimate of the extra work required if the tree is terminated at a panel  $P$  with  $n$  centers with an estimate of the work required if  $P$  is subdivided once and the tree is then terminated:  $n_\ell$  is taken to be the value of  $n$  for which these costs balance. The estimates are derived under the assumption that the distributions of centers and evaluation points in  $P$  and its neighbors are locally uniform.

In this section our estimates of work are based on using  $\mathcal{O}(k^2/2)$  processes to recenter polynomials and shift moments rather than using the  $\mathcal{O}(k \log k)$  FFT-based processes that are asymptotically superior. As pointed out previously,  $k$  is usually so small that we would not expect the FFT-based processes to offer much gain in practice.

The assumption of locally uniform density makes it reasonable to suppose that  $P$  is the center of three adjacent panels of the same size, each containing approximately  $n$  centers and  $bn$  evaluation points uniformly distributed over the panels. Subdividing  $P$  then entails the following extra work for the central large panel: two shifts to center moments on the upward sweep taking  $h_{\ell+1}^2$  flops, two shifts to recenter polynomials on the downward sweep taking  $h_\ell^2$  flops, and four more conversions of far field moments into local polynomials taking  $2(\deg(q_{\ell+1}))^2$  flops. There is also some work that is linear in  $h_{\ell+1}$  to combine moments and polynomials. Hence the flop count for the extra work in the setup stages is

$$h_{\ell+1}^2 + h_\ell^2 + 2(\deg(q_{\ell+1}))^2 + \text{lower-order terms.}$$

On the other hand, the extra subdivision may well decrease the work in evaluation. Instead of having to evaluate the direct contribution at  $bn$  evaluation points from the  $3n$  centers in  $P$  and its two neighbors, the algorithm needs only to evaluate the direct

contribution at  $bn/2$  evaluation points in each child from the  $3n/2$  centers in the child and its two neighbors. Therefore, the total evaluation cost on  $P$  “decreases” after subdivision from

$$3bn^2c_\phi + bnh_\ell + \text{lower-order terms}$$

to

$$2\left(\frac{3bn^2}{4}c_\phi + \frac{bn}{2}h_{\ell+1}\right) + \text{lower-order terms.}$$

Under the further assumption that

$$h_{\ell+1} \approx h_\ell \approx \deg(q_{\ell+1}) = k_\ell,$$

the extra setup cost will balance the decreased evaluation time when

$$n_\ell \approx k_\ell \sqrt{8/(3bc_\phi)}.$$

Hence, considering also the nonarithmetic work involved in further subdivision it is reasonable to set the threshold center count for subdividing a panel to

$$n_\ell \approx 2k_\ell/\sqrt{bc_\phi}.$$

**9.2. An alternative multiple panel center count subdivision strategy.**

The basic strategy described above bases the threshold for subdivision of  $P$  solely on the number of centers in  $P$ . Unfortunately, as noted by Powell [16], if the centers are unevenly distributed it is possible for this strategy to produce a tree in which a panel  $Q$  with a large number  $m_1 \gg n$  of centers is adjacent to a childless panel  $P$  with  $m_2 \leq n$  centers. Thus  $Q$  is on the direct list of  $P$ , and the corresponding work in a single approximate evaluation of  $s$  within  $P$  involves at least  $m_1 + m_2 \gg n$  evaluations of  $\phi$ . We therefore present here an alternative strategy in which the threshold is determined by the number of centers in both a panel and its peers. This places an upper bound on the number of direct evaluations of  $\phi$  required for any approximate evaluation of  $s$  regardless of the distribution of centers. The subdivision strategy and the corresponding limit on the number of centers in any direct list are summarized in the following lemma.

LEMMA 4. *Suppose that in forming the tree  $\mathcal{P}$  a panel  $P$  is subdivided whenever  $\sum_{Q \in p(P)} \#Q \geq n$ . Then for any panel  $P$ ,  $\sum_{Q \in d(P)} \#Q \leq 2n - 2$ .*

*Proof.* First note that  $P$  belongs to its own direct list if and only if  $P$  is childless. Also note that a child inherits the direct list of its parent. Hence, childless panels are the extreme cases, and it suffices to prove the lemma for childless panels alone.

Observe also that in the direct list of any panel  $Q$  there are at most two panels (excluding  $Q$  itself) at any one level. Furthermore, if there are two panels in the list at a particular level then they lie on either side of  $Q$ .

Let  $P$  be a childless panel. Consider now the set of panels in the direct list of  $P$  and lying to the right of  $P$ . The case when this set is empty is trivial. Hence we will assume it contains at least one member, and we choose  $Q$  to be the panel in this set with the highest level. Suppose the level of  $Q$  is  $j$ ,  $2 \leq j \leq \ell(P)$ . Then  $Q$  must be childless and adjacent to the ancestor  $P_j(P)$  of  $P$  at level  $j$ . Furthermore, all other panels on the direct list of  $P$  and to the right of  $P$  must be subsets of  $P_j(P)$ . Since  $Q$  has not been subdivided and  $P_j(P) \in p(Q)$ , the sum of the number of centers in

panels on the direct list of  $P$  lying to the right of  $P$  and the number of centers in  $P$  itself is less than  $n$ .

Clearly the argument can be repeated for panels on the direct list of  $P$  lying to the left of  $P$ . Hence the total number of centers in all panels on the direct list of  $P$  does not exceed  $2n - 2$ .  $\square$

Finally, under the assumption that centers and evaluation points are again locally uniformly distributed, an analysis almost identical to that of the previous subsection shows that the threshold  $n$  in Lemma 4 should be adaptively set at

$$n_\ell \approx 6k_\ell / \sqrt{bc_\phi}.$$

### Appendix A. Tighter estimates for functions analytic in $0 < \Re(z) < 2$ .

In this appendix we give the tighter version of Lemma 3 for functions  $\phi(z)$  real on  $[0, 2] \subset \mathbf{R}$ , analytic in the strip  $0 < \Re(z) < 2$ , and continuous at  $(0, 0)$  and  $(2, 0)$  previously promised (see near equation (7.7)). If  $\mathcal{G}$  denotes the set of such functions, then  $\mathcal{G}$  is in a sense the limit of  $Z(M, \alpha)$  as  $\alpha \rightarrow (\pi/2)^-$ .  $\mathcal{G}$  includes most basic functions of interest, in particular the Gaussian, generalized multiquadric, and thin-plate spline.

The result that we are about to prove is the natural extension of the lemma to  $\alpha = \pi/2$  but without the unnecessary assumption that  $\phi$  be bounded on the strip. Rather all that is required is that  $\phi$  be continuous, and therefore bounded, on the closed elliptic disk  $F_1$  defined below and analytic on its interior.

An examination of the proof of the lemma shows if  $\alpha = \pi/2$ ,  $0 < \beta < 1$  is fixed and varying  $0 < t \leq 1$ , all the maximal ellipses are again identical in the transformed setting. Indeed they reduce to that ellipse  $E_\rho$  of the form (7.3) which passes through the point  $(\tilde{x}, \tilde{y}) = (-1/\beta, 0)$ . This observation implies

$$\rho + \rho^{-1} = \left| 1 + \frac{1}{\beta} \right| + \left| 1 - \frac{1}{\beta} \right|,$$

which has as its largest root

$$\rho = \frac{1}{\beta} \left( 1 + \sqrt{1 - \beta^2} \right).$$

Now if we define the ellipses  $F_t$ ,  $0 < t \leq 1$ , to be the images of the maximal ellipse  $E_\rho$  under the inverse transforms from the  $\tilde{x}$ - $\tilde{y}$  plane back to the  $x$ - $y$  plane, then  $\{F_t\}$  is a family of similar ellipses with  $F_t$  having its major axis from  $(0, 0)$  to  $(2t, 0)$  (see Figure 2). Thus  $F_1$  encloses all the other ellipses  $F_t$ , and we can bound the maximum modulus of  $\phi$  on  $F_t$  by its maximum modulus on  $F_1$ . Hence best approximations to  $\phi$  satisfy the estimate (7.2) with  $\rho$  now given by (7.7) and  $M$  being the maximum modulus of  $\phi$  over the boundary of  $F_1$ .

**Appendix B. Modifications and extensions.** There are clearly many possible ways to improve the algorithm's performance in general or in certain special cases. In this section we briefly look at two such possibilities: using more than one approximating polynomial at each level and taking advantage of scale invariance in certain basic functions.

**B.1. Multiple approximating polynomials.** The algorithm of section 8 uses only one polynomial approximation to  $\phi$  at each level. This has the advantage of enabling approximation of the influence of two of the typical three panels on the

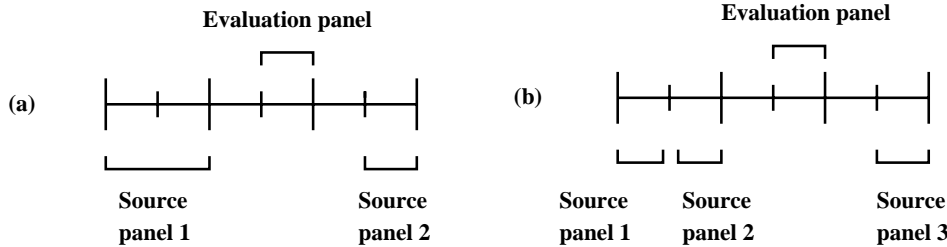


FIG. 3. Two different ways of incorporating the influence of panels on the interaction list.

interaction list of  $P$  via a single polynomial formed using the moments of their common parent panel (see Figure 3a).<sup>5</sup> An alternative possibility is to use two polynomial approximations to  $\phi$ . This has the advantage of allowing separate, and probably lower-degree, approximations to the far field influence of each panel on the interaction list of  $P$  (see Figure 3b) but requires approximately  $3/2$  times as many polynomials to be formed from moments at each level.

Table 2 below shows the polynomial degree sufficient for an economized polynomial to yield approximations of the specified accuracy to  $\phi$  on the specified intervals. The table shows that, as predicted above, the degree of the approximating polynomial can often be lowered when the approximation is required on proper subsets. The timings given in Table 1 are derived from a *two approximating polynomial* implementation of the algorithm. Our opinion is that for one-dimensional codes it pays to use multiple approximating polynomials as in Figure 3b and not to clump as in Figure 3a. In higher dimensions, however, it probably pays to clump.

**B.2. Efficient approximation of scale invariant basic functions.** Savings in the tree construction stage are possible when the basic function is effectively scale invariant. By effectively scale invariant we mean that the function is either scale invariant or that a change of scale can be compensated for by a low-degree polynomial correction. Examples are  $\phi(x) = x^2 \log x$  or  $\phi(x) = 1/x$ . For such functions a collection of polynomial approximations need only be calculated once at the top level in the algorithm and can then simply be reused in modified form at subsequent levels in the tree.

This is most easily seen with the aid of an example. Consider the thin-plate spline  $\phi(x) = x^2 \log x$  and suppose that  $0 < \beta < 1$  is fixed. Next let  $p_k$  be a polynomial of degree  $k \geq 2$  that approximates  $\phi$  on  $[1 - \beta, 1 + \beta]$ . Now for any  $0 < t \leq 1$  we wish to find a polynomial approximation  $q_k$  of degree  $k$  to  $\phi$  on  $[(1 - \beta)t, (1 + \beta)t]$ . To do this let  $u = rt$ , so that

$$\phi(u) = \phi(rt) = r^2 t^2 \ln r + r^2 t^2 \log t = t^2 \phi(r) + r^2 t^2 \log t.$$

Then a polynomial  $q_k$  of degree  $k$  that approximates  $\phi$  on  $[(1 - \beta)t, (1 + \beta)t]$  is

$$(B.1) \quad q_k(u) = t^2 p_k(u/t) + u^2 \log t,$$

and the associated error is given by

$$(B.2) \quad \|\phi - q_k\|_{L^\infty[(1-\beta)t, (1+\beta)t]} = t^2 \|\phi - p_k\|_{L^\infty[1-\beta, 1+\beta]}.$$

<sup>5</sup>Zhao [19] uses such clusters of source panels in the three-dimensional fast multipole setting. [6] analyzes the effect of clumping for thin-plate splines in  $\mathbf{R}^2$ .

TABLE 2

Degree of the economized polynomial sufficient to yield the specified absolute accuracy when approximating  $\phi$  on the given interval.

$\phi$	Interval	Accuracy		
		$10^{-4}$	$10^{-8}$	$10^{-12}$
Gaussian $e^{-10x^2}$	$[2^{-2}, 1]$	8	14	20
	$[2^{-5}, 2^{-3}]$	3	6	9
	$[2^{-8}, 2^{-6}]$	2	3	5
	$[2^{-1}, 1]$	5	11	14
	$[2^{-4}, 2^{-3}]$	3	5	8
	$[2^{-7}, 2^{-6}]$	1	3	4
Multiquadric $\sqrt{x^2 + .001}$	$[2^{-2}, 1]$	3	11	19
	$[2^{-5}, 2^{-3}]$	4	8	15
	$[2^{-8}, 2^{-6}]$	2	5	8
	$[2^{-1}, 1]$	1	6	12
	$[2^{-4}, 2^{-3}]$	2	7	11
	$[2^{-7}, 2^{-6}]$	2	4	7
Thin-plate $x^2 \ln(x)$	$[2^{-2}, 1]$	4	10	17
	$[2^{-5}, 2^{-3}]$	3	7	14
	$[2^{-8}, 2^{-6}]$	1	5	11
	$[2^{-1}, 1]$	3	7	12
	$[2^{-4}, 2^{-3}]$	2	5	10
	$[2^{-7}, 2^{-6}]$	1	4	8

From equations (B.1), (B.2), and the uniqueness of best uniform approximations, it follows that if either of  $q_k(u)$  or  $p_k(t)$  is a best approximation then so is the other. Thus we cannot expect direct approximations at each level to improve upon the approximations obtained via this rescaling technique.

## REFERENCES

- [1] B. K. ALPERT AND V. ROKHLIN, *A fast algorithm for the evaluation of Legendre expansions*, SIAM J. Sci. Comput., 12 (1991), pp. 158–179.
- [2] B. K. ALPERT, G. BEYLKIN, R. COIFMAN, AND V. ROKHLIN, *Wavelet like bases for the fast solution of second-kind integral equations*, SIAM J. Sci. Comput., 14 (1993), pp. 159–184.
- [3] B. K. ALPERT, *A class of bases in  $L^2$  for the sparse representation of integral operators*, SIAM J. Math. Anal., 24 (1993), pp. 246–262.
- [4] A. W. APPEL, *An efficient program for many body simulations*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 85–103.
- [5] J. BARNES AND P. HUT, *A hierarchical  $\mathcal{O}(N \log N)$  force-calculation algorithm*, Nature, 324 (1986), pp. 446–449.
- [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: I*, Comput. Math. Appl., 24 (1992), pp. 7–19.
- [8] G. BEYLKIN, R. COIFMAN AND V. ROKHLIN, *Fast wavelet transforms and numerical algorithms: I*, Comm. Pure Appl. Math., 44 (1991), pp. 141–183.
- [9] J. CARRIER, L. GREENGARD AND V. ROKHLIN, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Comput., 9 (1988), pp. 669–686.
- [10] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.
- [11] L. GREENGARD, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.
- [12] L. GREENGARD AND V. ROKHLIN, *On the Efficient Implementation of the Fast Multipole Algorithm*, Department of Computer Science Report 602, Yale University, New Haven, CT, 1988.

- [13] L. GREENGARD AND J. STRAIN, *The fast Gauss transform*, SIAM J. Sci. Comput., 2 (1991), pp. 79–94.
- [14] W. HACKBUSH AND Z. K. NOWAK, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numer. Math., 54 (1989), pp. 463–491.
- [15] G. G. LORENTZ, *Approximation of Functions*, Holt, Rinehart and Winston, New York, 1966.
- [16] M. J. D. POWELL, *Truncated Laurent expansions for the fast evaluation of thin plate splines*, Algorithms for Approx., 5 (1993), pp. 99–120.
- [17] T. J. RIVLIN, *The Chebyshev Polynomials*, Wiley-Interscience, New York, 1974.
- [18] J. STRAIN, *The fast Gauss transform with variable scales*, SIAM J. Sci. Comput., 12 (1991), pp. 1131–1139.
- [19] F. ZHAO, *An  $\mathcal{O}(N)$  Algorithm for 3-Dimensional  $N$ -Body Simulations*, Technical Report 995, Artificial Intelligence Laboratory, MIT, Cambridge, MA, 1987.