SIAM J. SCI. COMPUT.
Vol. 33, No. 2, pp. 869–892

STABLE COMPUTATIONS WITH GAUSSIAN RADIAL BASIS FUNCTIONS*

BENGT FORNBERG†, ELISABETH LARSSON‡, AND NATASHA FLYER§

Abstract. Radial basis function (RBF) approximation is an extremely powerful tool for representing smooth functions in nontrivial geometries since the method is mesh-free and can be spectrally accurate. A perceived practical obstacle is that the interpolation matrix becomes increasingly ill-conditioned as the RBF shape parameter becomes small, corresponding to flat RBFs. Two stable approaches that overcome this problem exist: the Contour-Padé method and the RBF-QR method. However, the former is limited to small node sets, and the latter has until now been formulated only for the surface of the sphere. This paper focuses on an RBF-QR formulation for node sets in one, two, and three dimensions. The algorithm is stable for arbitrarily small shape parameters. It can be used for thousands of node points in two dimensions and still more in three dimensions. A sample MATLAB code for the two-dimensional case is provided.

Key words. radial basis function, ill-conditioning, shape parameter, stable

AMS subject classifications. 65D05, 65D15, 65F35

DOI. 10.1137/09076756X

1. Introduction. When using radial basis functions (RBFs), the best accuracy is often achieved when their shape parameter ε is small, meaning that the basis functions are relatively flat. It was mistakenly believed for a number of years by many RBF practitioners that this computational regime inevitably was associated with numerical ill-conditioning when, in fact, the only thing that was ill-conditioned was the most immediate numerical algorithm (denoted RBF-Direct in some of the current literature). So far, only two numerical algorithms have been presented that are able to compute stably even in the $\varepsilon \to 0$ (increasingly flat) basis function limit: the Contour-Padé method [13] (see [31] for the significantly improved RBF-RA version) and the RBF-QR method [11]. These two methods are based on different principles and have different limitations. The Contour-Padé/RBF-RA method is limited to a relatively low number of RBF nodes (N slightly less than a hundred in two dimensions, more in three dimensions). The RBF-QR algorithm has previously been developed only for the case when the nodes are distributed over the surface of a sphere, then allowing N-values in the thousands. The present paper describes how the RBF-QR approach can also be implemented for general domains in one, two, and three dimensions. With quasi-uniform nodes, the algorithm works well for tens of points in one dimension, hundreds of points in two dimensions, and thousands of points in

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 8, 2009; accepted for publication (in revised form) January 20, 2011; published electronically April 7, 2011.

http://www.siam.org/journals/sisc/33-2/76756.html

†Department of Applied Mathematics, University of Colorado, 526 UCB, Boulder, CO 80309 (fornberg@colorado.edu). This author's work was supported by NSF grants DMS-0611681, ATM-0620068, and DMS-0914647.

‡Department of Information Technology, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden (bette@it.uu.se). This author's work was supported by the Swedish Research Council and the Göran Gustafsson Foundation.

§Institute for Mathematics Applied to Geosciences, National Center for Atmospheric Research, 1850 Table Mesa Drive, Boulder, CO 80305 (flyer@ucar.edu). This author's work was supported by NSF grants ATM-0620100 and DMS-0934317. The National Center for Atmospheric Research (NCAR) is sponsored by the National Science Foundation.

three dimensions using double precision arithmetics. Using quad precision increases the ranges significantly. We can, for example, solve for $N = 2700$ points in two dimensions retaining full double precision accuracy [8]. With quasi-uniform node distributions, errors in typical RBF implementations will eventually grow with the number of nodes for two reasons: (i) increasing condition number, and (ii) an intrinsic ill-conditioning of spectrally accurate methods on quasi-uniform node sets leading to large errors near boundaries. Making the basis functions less flat is frequently used to address (i), then trading conditioning against accuracy. The present algorithm resolves the ill-conditioning issue without any such trade-offs. The issue (ii) can then be addressed separately. Here, this is done by clustering nodes towards the domain boundaries. RBF-QR in double precision then works for thousands of node points in all three cases.

The outline of the paper is as follows: section 2 gives the background to the ill-conditioning of the RBF-Direct approach. The RBF-QR method for the sphere is briefly reviewed in section 3, and then the planar two-dimensional (2-D) algorithm for Gaussian RBFs is derived in section 4. The corresponding expansions for the one-dimensional (1-D) and three-dimensional (3-D) cases are also given in this section. Implementation details are covered in section 5, and numerical results are demonstrated in section 6. The conclusions in section 7 are followed by two appendices containing (A) a sample MATLAB code, and (B) a brief overview of the steps in forming the 2-D polar-Chebyshev expansions of the Gaussian RBFs and details concerning the truncation of the Chebyshev expansion in one dimension, the polar-Chebyshev expansion in two dimensions, and the spherical-Chebyshev expansion in three dimensions.

2. The ill-conditioning of the RBF-Direct algorithm. Given an RBF $\phi(r)$ and scattered data $\{\underline{x}_k, f_k\},\ k = 1, 2, \ldots, N$, the RBF-Direct approach for finding the interpolant

$$s(\underline{x}, \varepsilon) = \sum_{k=1}^{N} \lambda_k \phi(||\underline{x} - \underline{x}_k||) \tag{2.1}$$

simply computes the coefficients λ_k as the solution of the linear system

$$A\,\underline{\lambda} = \underline{f}, \tag{2.2}$$

where the matrix A has the entries $A_{j,k} = \phi(||\underline{x}_j - \underline{x}_k||)$ and the column vectors $\underline{\lambda}$ and $\underline{f}$ contain the λ_k and the f_j values, respectively. When the basis functions are made increasingly flat, the A-matrix becomes very ill-conditioned. As a result, the λ_k values become extremely large in magnitude (some positive and others negative), and a vast amount of numerical cancellation then occurs when the $O(1)$-sized quantity $s(\underline{x}, \varepsilon)$ is obtained in (2.1) through the combination of these large quantities. Thus, in the flat basis function regime, (2.2) and (2.1) form two successive ill-conditioned numerical steps in obtaining a quantity $s(\underline{x}, \varepsilon)$ that we know in general depends in a well-conditioned way on the data $\{\underline{x}_k, f_k\}$ [4], [6], [14], [19], [26]. Although (2.2) and (2.1) mathematically define the RBF interpolant $s(\underline{x}, \varepsilon)$ for any value of ε, these equations are very unsuitable for numerical use when ε is small. The Contour-Padé/RBF-RA and the RBF-QR algorithms both compute exactly the same quantity $s(\underline{x}, \varepsilon)$ but instead follow sequences of steps that all remain completely stable even when $\varepsilon \to 0$.

The exact rate by which the ill-conditioning of the A-matrix worsens for ε small and N increasing was studied in [15]. In the case when the nodes $\underline{x}_i$ are scattered in

TABLE 2.1
The definitions of some infinitely smooth radial functions

Name	Abbreviation	Definition
Gaussian	GA	$\phi(r)=e^{-(\varepsilon r)^2}$
Multiquadric	MQ	$\phi(r)=\sqrt{1+(\varepsilon r)^2}$
Inverse multiquadric	IMQ	$\phi(r)=1/\sqrt{1+(\varepsilon r)^2}$
Inverse quadratic	IQ	$\phi(r)=1/(1+(\varepsilon r)^2)$
Bessel	BE	$\phi_d(r)=\dfrac{J_{d/2-1}(\varepsilon r)}{(\varepsilon r)^{d/2-1}}$

TABLE 2.2
Number of eigenvalues of sizes $O(1)$, $O(\varepsilon^2)$, $O(\varepsilon^4)$, $O(\varepsilon^6)$, ...

1-D nonperiodic case (GA, MQ, IMQ, IQ, $\text{BE}_{d=2,3,4,\ldots}$)							
1	1	1	1	1	1	1	...
2-D nonperiodic case (GA, MQ, IMQ, IQ, $\text{BE}_{d=3,4,5,\ldots}$)							
1	2	3	4	5	6	7	...
2-D nonperiodic case ($\text{BE}_{d=2}$)							
1	2	2	2	2	2	2	...
On the surface of a sphere (GA, MQ, IMQ, IQ)							
1	3	5	7	9	11	13	...
3-D nonperiodic case (GA, MQ, IMQ, IQ)							
1	3	6	10	15	21	28	...

two dimensions, and using any of the standard RBF choices such as GA, MQ, IMQ, IQ (see Table 2.1), the A-matrix will have 1 eigenvalue of size $O(1)$, 2 of size $O(\varepsilon^2)$, 3 of size $O(\varepsilon^4)$, ... until all the N eigenvalues are accounted for. The BE radial functions were shown in [7], [9] to have a number of unusual properties. The case with $d = 2$ is seen in Table 2.2 to be anomalous in terms of conditioning, featuring particularly severe ill-conditioning if implemented with RBF-Direct.

Much RBF literature has been devoted to finding "optimal" values for the shape parameter [5], [16], [23]. In some cases, this optimal value occurs in a shape parameter regime where the ill-conditioning of RBF-Direct is not an issue. At other times, these attempts have amounted to trying to strike a favorable balance between unavoidable accuracy losses for large ε and avoidable RBF-Direct accuracy losses for low values of ε. Since it is now understood that the ill-conditioning can be bypassed, such balances need to be reassessed. An accuracy-limiting factor other than ill-conditioning then emerges in the decreasing ε-regime. This has previously been observed and discussed from different perspectives several times; see, for example, [3], [11], [12], [15], [18], [19].

3. RBF-QR in the case of nodes on the surface of the unit sphere. This case offers the algebraically simplest implementation of the RBF-QR method, and we recall it briefly as a background to our following description of the 2-D nonperiodic case. If an RBF is centered at $\underline{x}_k$, its value at $\underline{x}$ (with both points on the surface of the unit sphere) was shown in [2] to be expressible in a sum

$$\begin{aligned}(3.1)\qquad \phi(||\underline{x}-\underline{x}_k||) = {} & c_{0,0}Y_0^0(\underline{x}) + \varepsilon^2\{c_{1,-1}Y_1^{-1}(\underline{x}) + c_{1,0}Y_1^0(\underline{x}) + c_{1,1}Y_1^1(\underline{x})\} \\ & + \varepsilon^4\{c_{2,-2}Y_2^{-2}(\underline{x}) + c_{2,-1}Y_2^{-1}(\underline{x}) + \cdots + c_{2,2}Y_2^2(\underline{x})\} \\ & + \varepsilon^6\{c_{3,-3}Y_3^{-3}(\underline{x}) + c_{3,-2}Y_3^{-2}(\underline{x}) + \cdots + c_{3,3}Y_3^3(\underline{x})\} \\ & + \cdots,\end{aligned}$$

where $Y_\mu^\nu(\underline{x})$ are increasing order spherical harmonics (SPH). For all the standard RBF types, simple explicit forms are available for all the coefficients $c_{\mu,\nu}$ [2], [11]. It should be noted that the expansion (3.1) is not quite a Taylor expansion in ε since the coefficients $c_{\mu,\nu}$ have a weak ε-dependence (however, they converge to finite nonzero values when $\varepsilon \to 0$). In view of (3.1), a column vector of the RBFs, centered at the successive nodes, can be written as

$$\begin{bmatrix} \phi(||\underline{x}-\underline{x}_1||) \\ \phi(||\underline{x}-\underline{x}_2||) \\ \vdots \\ \vdots \\ \phi(||\underline{x}-\underline{x}_N||) \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & C & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \varepsilon^0 & & & & & \\ & \varepsilon^2 & & & & \\ & & \ddots & & & \\ & & & \varepsilon^4 & & \\ & & & & \ddots & \end{bmatrix} \begin{bmatrix} Y_0^0(\underline{x}) \\ Y_1^{-1}(\underline{x}) \\ \vdots \\ \vdots \\ \vdots \end{bmatrix},$$

where C is a matrix with entries of size $O(1)$. At this intermediate stage, the ill-conditioning due to the scaling of the RBFs has been confined to the diagonal matrix E above. If we multiply from the left with any nonsingular matrix, we obtain new basis functions but do not change the space that is spanned by them. We want to utilize this observation to create a well-conditioned basis in exactly the same space. To achieve this, we split $C = QR$, where Q is unitary and R is upper triangular, and then multiply with $E_N^{-1}Q^*$, where E denotes the diagonal matrix with the increasing powers of ε and E_k denotes the first $(k \times k)$ part of E. The product Q^*Q becomes I, and the product $E_N^{-1}RE$ becomes an upper triangular matrix with a diminishing number of significant upper diagonals as ε decreases. The resulting column vector

$$\underline{\Psi}(\underline{x}) = E_N^{-1}R\ E\ \underline{Y}(\underline{x})$$

provides the well-conditioned basis functions that we will use numerically. Expressed in equation form, the steps just described amounted to starting with

$$\underline{\Phi}(\underline{x}) = C\ E\ \underline{Y}(\underline{x})$$

and then using as our new set of basis functions

$$\underline{\Psi}(\underline{x}) = E_N^{-1}Q^*\underline{\Phi}(\underline{x}) = E_N^{-1}\ \overbrace{Q^*C\ E}^{QR}\ \ \underline{Y}(\underline{x}) = E_N^{-1}R\ E\ \underline{Y}(\underline{x}).$$

The matrix $E_N^{-1}R\ E$ is well-conditioned, is upper triangular, has a main diagonal with elements of $\mathcal{O}(1)$, and has only a few significant superdiagonals. No unstable numerics was used in forming this new basis function set (even in the $\varepsilon \to 0$ limit), and it still spans exactly the same space as the original RBF set.

The number of independent functions associated with each power ε^j in (3.1), $\{1,\ 3,\ 5,\ 7, \dots\}$ for $j = 0, 2, 4, \dots$, respectively, determines the rate by which the powers enter in the diagonal of the E-matrix, and we see that this sequence perfectly matches the counts for the sphere case given in Table 2.2. Thus, the RBF-QR algorithm improves the conditioning at just the same rate as the one describing how it otherwise would have deteriorated; i.e., the conditioning remains essentially invariant with both N and ε.

4. RBF-QR in nonperiodic Euclidean space. Consider a radial function $\phi(r)$ with Taylor expansion $\phi(r) = \sum_{j=0}^{\infty} c_j(\varepsilon r)^{2j}$. If we center it at a point $\underline{x}_k$, we

have $r = \|\underline{x} - \underline{x}_k\|$. For example, in two dimensions, with $\underline{x}_k = (x_k, y_k)$, this yields $r = \sqrt{(x - x_k)^2 + (y - y_k)^2}$ with the expansion in powers of ε

$$\psi(x, y, x_k, y_k) = \sum_{j=0}^{\infty} c_j \varepsilon^{2j} ((x - x_k)^2 + (y - y_k)^2)^j. \tag{4.1}$$

The new functions of x and y that enter for each even power of ε are in turn

$$\{1\}, \{x^2, x, y^2, y\}, \{x^4, x^3, x^2 y, x^2 y^2, xy, xy^2, y^3, y^4\}, \ldots, \tag{4.2}$$

corresponding to the sequence

$$\begin{array}{cccccccc} \text{power of } \varepsilon & 0 & 2 & 4 & 6 & 8 & \ldots, \\ \text{number of additional functions} & 1 & 4 & 8 & 12 & 16 & \ldots. \end{array} \tag{4.3}$$

The sequence $\{1,\ 4,\ 8,\ 12,\ 16, \ldots\}$ does not match either of the sequences shown for 2-D nonperiodic cases in Table 2.2. Reexpansion of the radial functions in the monomials (4.2) will therefore not allow the ill-conditioning to be eliminated fully. This is also true for the 1-D case and in higher dimensions. The challenge thus becomes to find alternative expansion functions for which the counting works out correctly. We have so far found such expansions only in the GA and BE cases.

4.1. Expansion of the GA radial function. The radial function $\phi(r) = e^{-\varepsilon^2 r^2}$ centered at the point $\underline{x}_k$ becomes

$$\psi(\underline{x}, \underline{x}_k) = e^{-\varepsilon^2 \|\underline{x} - \underline{x}_k\|^2} = e^{-\varepsilon^2 (\underline{x} - \underline{x}_k) \cdot (\underline{x} - \underline{x}_k)} = e^{-\varepsilon^2 (\underline{x} \cdot \underline{x})} e^{-\varepsilon^2 (\underline{x}_k \cdot \underline{x}_k)} e^{2\varepsilon^2 (\underline{x} \cdot \underline{x}_k)}. \tag{4.4}$$

Only the last factor above mixes $\underline{x}$ and $\underline{x}_k$ values. It has the Taylor expansion

$$e^{2\varepsilon^2 (\underline{x} \cdot \underline{x}_k)} = 1 + 2\varepsilon^2 (\underline{x} \cdot \underline{x}_k) + \frac{(2\varepsilon^2)^2}{2!} (\underline{x} \cdot \underline{x}_k)^2 + \cdots = \sum_{j=0}^{\infty} \frac{(2\varepsilon^2)^j}{j!} (\underline{x} \cdot \underline{x}_k)^j. \tag{4.5}$$

Our first derivation of the RBF-QR algorithm in the nonspherical case was for 2-D nonperiodic planar geometries. The 1-D and 3-D cases then followed by applying similar techniques. Therefore, the description of the algorithm below focuses on the 2-D case and outlines only the main features for other dimensions.

In two dimensions, the radial function $\phi(r) = e^{-\varepsilon^2 r^2}$ centered at the point (x_k, y_k) becomes

$$\psi(x, y, x_k, y_k) = e^{-\varepsilon^2 (x^2 + y^2)} \cdot e^{-\varepsilon^2 (x_k^2 + y_k^2)} \cdot e^{2\varepsilon^2 (x\, x_k + y\, y_k)}, \tag{4.6}$$

where the last factor has the Taylor expansion

$$e^{2\varepsilon^2 (x\, x_k + y\, y_k)} = 1 + 2\varepsilon^2 (x\, x_k + y\, y_k) + \frac{2^2 \varepsilon^4}{2!} (x\, x_k + y\, y_k)^2 + \cdots. \tag{4.7}$$

Thanks to having factored out $e^{-\varepsilon^2 (x^2 + y^2)}$ (a "harmless" factor as $\varepsilon \to 0$), the degrees of the polynomials in the subsequent Taylor expansion increase by just one order at a time (rather than by two orders, as in (4.1) and (4.2)). In place of (4.2) the expansion functions are now

$$e^{-\varepsilon^2 (x^2 + y^2)} \cdot \{\{1\}, \{x, y\}, \{x^2, xy, y^2\}, \{x^3, x^2 y, xy^2, y^3\}, \ldots\}, \tag{4.8}$$

and (4.3) has become replaced by

$$\begin{array}{lcccccc} \text{power of } \varepsilon & 0 & 2 & 4 & 6 & 8 & \dots, \\ \text{number of functions} & 1 & 2 & 3 & 4 & 5 & \dots, \end{array} \tag{4.9}$$

now in perfect agreement with the corresponding sequence shown in Table 2.2.

This part of the algorithm, involving the factorization (4.4) and the subsequent Taylor expansion (4.5), is valid in any number of dimensions. In three dimensions, we can note that the counterparts to (4.6) and (4.8) involve factoring out $e^{-\varepsilon^2(x^2+y^2+z^2)}$, and then the function set corresponding to $\varepsilon^{2\mu}$ contains the $\frac{1}{2}(\mu+1)(\mu+2)$ independent homogeneous monomials in (x,y,z) of degree μ. Again, the counting matches what is shown for this case in Table 2.2, and the QR concept becomes fully applicable.

4.1.1. Conditioning improvement by conversion to polar coordinates. Many of the monomials in (4.8) become nearly linearly dependent when their degrees increase. Some of this will be circumvented if we express (4.6) in polar rather than in (x,y) coordinates. With $\psi(r,\theta,r_k,\theta_k)$ denoting a GA radial function centered at the polar coordinate location (r_k,θ_k), its value at the location (r,θ) follows from rewriting (4.6) as $e^{-\varepsilon^2 r_k^2}\cdot e^{-\varepsilon^2 r^2}\cdot e^{2\varepsilon^2 r_k r(\cos\theta_k\cos\theta+\sin\theta_k\sin\theta)}$ (cf. the first part of Appendix B):

$$\begin{aligned} \psi(r,\theta,r_k,\theta_k) = 2\cdot e^{-\varepsilon^2 r_k^2}\cdot e^{-\varepsilon^2 r^2}\cdot [& \\ & (\varepsilon^2 r_k r)^0 \ \left\{\tfrac{1}{2}\cdot\tfrac{1}{0!0!}\Theta_0\right\} \\ & +(\varepsilon^2 r_k r)^2 \ \left\{\tfrac{1}{2}\cdot\tfrac{1}{1!1!}\Theta_0 \ +\tfrac{1}{2!0!}\Theta_2\right\} \\ & +(\varepsilon^2 r_k r)^4 \ \left\{\tfrac{1}{2}\cdot\tfrac{1}{2!2!}\Theta_0 \ +\tfrac{1}{3!1!}\Theta_2 \ +\tfrac{1}{4!0!}\Theta_4\right\} \\ & +\cdots + \\ & (\varepsilon^2 r_k r)^1 \ \left\{ \ \tfrac{1}{1!0!}\Theta_1\right\} \\ & +(\varepsilon^2 r_k r)^3 \ \left\{ \ \tfrac{1}{2!1!}\Theta_1 \ +\tfrac{1}{3!0!}\Theta_3\right\} \\ & +(\varepsilon^2 r_k r)^5 \ \left\{ \ \tfrac{1}{3!2!}\Theta_1 \ +\tfrac{1}{4!1!}\Theta_3 \ +\tfrac{1}{5!0!}\Theta_5\right\} \\ & +\cdots]. \end{aligned} \tag{4.10}$$

Here Θ_m abbreviates $(\cos m\theta_k\cos m\theta+\sin m\theta_k\sin m\theta)$. Since their patterns are slightly different, the terms have been split into two groups, containing the powers $\{\varepsilon^0,\varepsilon^4,\varepsilon^8,\dots\}$ and $\{\varepsilon^2,\varepsilon^6,\varepsilon^{10},\dots\}$, respectively. In place of (4.8), we now have

$$\begin{array}{lllll} e^{-\varepsilon^2 r^2}\{ & \{1\}, & & & \\ & r\,\{\cos\theta, & \sin\theta\}, & & \\ & r^2\{1, & \cos 2\theta, & \sin 2\theta\}, & \\ & r^3\{\cos\theta, & \sin\theta, & \cos 3\theta, & \sin 3\theta\}, \\ & & \dots\ \} & & \end{array} \quad . \tag{4.11}$$

with (4.9) again valid.

Figure 4.1 displays the first four levels of the expansion functions in the case of $\varepsilon=1$, in the order they are listed in (4.11). They are all pure trigonometric modes in the θ-direction. As ε is made smaller, the $e^{-\varepsilon^2 r^2}$ factor will become increasingly close to one at finite distance from the origin, and their amplitude will approach $1,\ r,\ r^2,\dots$ for the successive rows in Figure 4.1.

The conversion to polar coordinates is only valid in two dimensions. For the 3-D case, spherical harmonic expansion functions are used in the angular directions, and

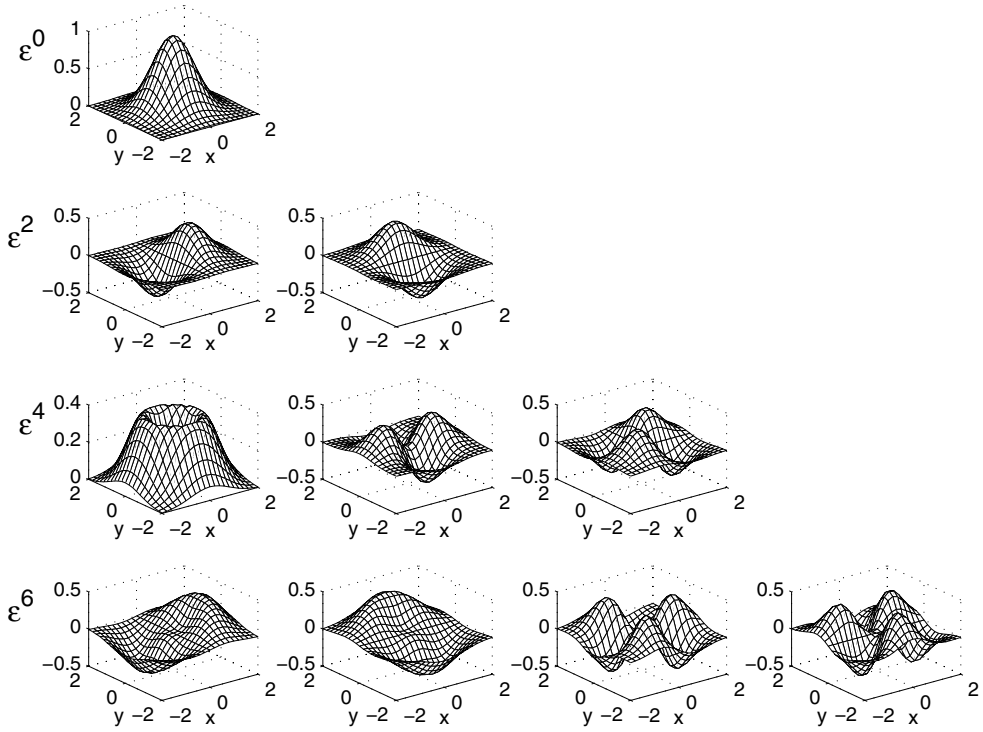

FIG. 4.1. *The expansion functions* (4.11) *that are needed for representing arbitrary translates of the GA radial function* $\phi(r) = e^{-(\varepsilon r)^2}$*, displayed in the case of* $\varepsilon = 1$.

the key formula is given by

$$
\begin{aligned}
\psi(r,\theta,\phi,r_k,\theta_k,\phi_k) = 4 \cdot e^{-\varepsilon^2 r_k^2} \cdot e^{-\varepsilon^2 r^2} \cdot [& \\
& (2\varepsilon^2 r_k r)^0 \ \left\{ \tfrac{2^0}{1!}\tfrac{0!}{0!} S_0 \right\} \\
& + (2\varepsilon^2 r_k r)^2 \ \left\{ \tfrac{2^0}{3!}\tfrac{1!}{1!} S_0 \ + \tfrac{2^2}{5!}\tfrac{2!}{0!} S_2 \right\} \\
& + (2\varepsilon^2 r_k r)^4 \ \left\{ \tfrac{2^0}{5!}\tfrac{2!}{2!} S_0 \ + \tfrac{2^2}{7!}\tfrac{3!}{1!} S_2 \ + \tfrac{2^4}{9!}\tfrac{4!}{0!} S_4 \right\} \\
& + \cdots + \\
& (2\varepsilon^2 r_k r)^1 \ \left\{ \tfrac{2^1}{3!}\tfrac{1!}{0!} S_1 \right\} \\
& + (2\varepsilon^2 r_k r)^3 \ \left\{ \tfrac{2^1}{5!}\tfrac{2!}{1!} S_1 \ + \tfrac{2^3}{7!}\tfrac{3!}{0!} S_3 \right\} \\
& + (2\varepsilon^2 r_k r)^5 \ \left\{ \tfrac{2^1}{7!}\tfrac{3!}{2!} S_1 \ + \tfrac{2^3}{9!}\tfrac{4!}{1!} S_3 \ + \tfrac{2^5}{11!}\tfrac{5!}{0!} S_5 \right\} \\
& + \cdots],
\end{aligned}
\tag{4.12}
$$

where $S_\mu = \Sigma'^{\mu}_{\nu=-\mu} Y_\mu^\nu(\theta_k,\phi_k) Y_\mu^\nu(\theta,\phi)$ with the prime indicating that the $\nu = 0$ term is halved. We have so far not investigated coordinate transformations in higher dimensions.

The expressions corresponding to (4.10) and (4.11) for BE radial functions and a discussion about generalizations to other RBF types can be found in [8].

4.1.2. Further conditioning improvement through use of Chebyshev polynomials. An RBF-QR implementation based on (4.10) performs much better than one based on (4.6) together with (4.7). This is because the trigonometric modes provide good independence in the θ-direction. We are still facing the problem that high powers of r tend to be nearly linearly dependent. A more attractive basis than monomials in r would be the Chebyshev polynomials. The relation between pure powers and Chebyshev polynomials has the general form

$$r^{2j+p} = \sum_{\ell=0}^{j} b_\ell T_{2\ell+p}(r), \quad p = 0, 1, \quad j = 0, 1, \ldots,$$

where the coefficients b_ℓ also depend on j and p. Explicit expressions can be found in [28]. However, if we were to directly convert all powers by this formula, we would increase the number of expansion functions at each level (except the first two), and the counting would be off again. Instead we need to look at which combinations are admissible. For the even powers of r, excluding the factor $e^{-\varepsilon^2 r^2}$, the first four levels are

$$\begin{array}{l|llll}
\varepsilon^0 & r^0 & & & \\
\varepsilon^4 & r^2 & r^2\{\cos(2\theta),\ \sin(2\theta)\} & & \\
\varepsilon^8 & r^4 & r^4\{\cos(2\theta),\ \sin(2\theta)\} & r^4\{\cos(4\theta),\ \sin(4\theta)\} & \\
\varepsilon^{12} & r^6 & r^6\{\cos(2\theta),\ \sin(2\theta)\} & r^6\{\cos(4\theta),\ \sin(4\theta)\} & r^6\{\cos(6\theta),\ \sin(6\theta)\}.
\end{array}$$

The corresponding set for odd powers of r is

$$\begin{array}{l|lll}
\varepsilon^2 & r^1\{\cos(\theta),\ \sin(\theta)\} & & \\
\varepsilon^6 & r^3\{\cos(\theta),\ \sin(\theta)\} & r^3\{\cos(3\theta),\ \sin(3\theta)\} & \\
\varepsilon^{10} & r^5\{\cos(\theta),\ \sin(\theta)\} & r^5\{\cos(3\theta),\ \sin(3\theta)\} & r^5\{\cos(5\theta),\ \sin(5\theta)\}.
\end{array}$$

By factoring out the lowest power of r in each column and then converting the remaining powers, we arrive at a new set of expansion functions,

$$\begin{array}{l|llll}
\varepsilon^0 & T_0 & & & \\
\varepsilon^4 & T_2 & r^2T_0\{\cos(2\theta),\sin(2\theta)\} & & \\
\varepsilon^8 & T_4 & r^2T_2\{\cos(2\theta),\sin(2\theta)\} & r^4T_0\{\cos(4\theta),\sin(4\theta)\} & \\
\varepsilon^{12} & T_6 & r^2T_4\{\cos(2\theta),\sin(2\theta)\} & r^4T_2\{\cos(4\theta),\sin(4\theta)\} & r^6T_0\{\cos(6\theta),\sin(6\theta)\} \\
\varepsilon^2 & T_1\{\cos(\theta),\sin(\theta)\} & & & \\
\varepsilon^6 & T_3\{\cos(\theta),\sin(\theta)\} & r^2T_1\{\cos(3\theta),\sin(3\theta)\} & & \\
\varepsilon^{10} & T_5\{\cos(\theta),\sin(\theta)\} & r^2T_3\{\cos(3\theta),\sin(3\theta)\} & r^4T_1\{\cos(5\theta),\sin(5\theta)\},
\end{array}$$

where the expansion coefficients are now ε-dependent, but the counting for the leading power of ε remains intact. As an example of how the conversion of a term with a higher power of r to the Chebyshev basis does not introduce any new expansion functions, consider the following example:

$$\varepsilon^{12} r^6 \cos(2\theta) = \left(b_3\left\{\varepsilon^{12} r^2 T_4(r)\right\} + \varepsilon^4 b_2\left\{\varepsilon^8 r^2 T_2(r)\right\} + \varepsilon^8 b_1\left\{\varepsilon^4 r^2 T_0(r)\right\}\right)\cos(2\theta),$$

where the functions in all three terms can be found in the same column in the display above and where the lower order ones have extra powers of ε in the coefficient. Let the new expansion functions be denoted by

$$\begin{cases}
T^c_{j,m}(\underline{x}) = e^{-\varepsilon^2 r^2} r^{2m} T_{j-2m}(r)\cos((2m+p)\theta), \\
T^s_{j,m}(\underline{x}) = e^{-\varepsilon^2 r^2} r^{2m} T_{j-2m}(r)\sin((2m+p)\theta), \quad 2m+p \neq 0,
\end{cases}$$

for $j \geq 0$ and $0 \leq m \leq \lfloor \frac{j}{2} \rfloor = \frac{j-p}{2}$, where $p = 0$ if j is even and $p = 1$ if j is odd. Then we have an expansion of the Gaussian RBF that takes the general form

$$\begin{aligned}\phi_k(\underline{x}) = \phi(\|\underline{x} - \underline{x}_k\|) = &\sum_{j=0}^{\infty} \sum_{m=0}^{\frac{j-p}{2}} d_{j,m} c_{j,m}(\underline{x}_k)\, T^c_{j,m}(\underline{x}) \\ &+ \sum_{j=0}^{\infty} \sum_{m=1-p}^{\frac{j-p}{2}} d_{j,m} s_{j,m}(\underline{x}_k)\, T^s_{j,m}(\underline{x}),\end{aligned} \tag{4.13}$$

where the scale factor $d_{j,m}$ is $\mathcal{O}(\varepsilon^{2j})$ and is chosen such that $c_{j,m}$ and $s_{j,m}$ are $\mathcal{O}(1)$. For a brief derivation of the coefficients, see Appendix B. The scale factors are

$$d_{j,m} = \frac{\varepsilon^{2j}}{2^{j-2m-1}\, \left(\frac{j+2m+p}{2}\right)!\, \left(\frac{j-2m-p}{2}\right)!}, \tag{4.14}$$

and the coefficients are given by

$$c_{j,m}(\underline{x}_k) = b_{2m+p} t_{j-2m}\, e^{-\varepsilon^2 r_k^2}\, r_k^j \cos((2m+p)\theta_k)\, {}_1F_2(\alpha_{j,m}, \beta_{j,m}, \varepsilon^4 r_k^2), \tag{4.15}$$

$$s_{j,m}(\underline{x}_k) = b_{2m+p} t_{j-2m}\, e^{-\varepsilon^2 r_k^2}\, r_k^j \sin((2m+p)\theta_k)\, {}_1F_2(\alpha_{j,m}, \beta_{j,m}, \varepsilon^4 r_k^2), \tag{4.16}$$

where $b_0 = 1$ and $b_m = 2$, $m > 0$, $t_0 = \frac{1}{2}$, and $t_j = 1$, $j > 0$, and the parameters for the hypergeometric function are $\alpha_{j,m} = \frac{j-2m+p+1}{2}$ and $\beta_{j,m} = \left[j - 2m + 1,\ \frac{j+2m+p+2}{2}\right]$.

4.1.3. The final form of the expansions in the 1-D and 3-D cases. The 1-D expansion is the simplest case, where we need only convert powers into Chebyshev polynomials. This leads to

$$\phi_k(x) = \sum_{j=0}^{\infty} d_j c_j(x_k)\, \tilde{T}_j(x) \tag{4.17}$$

with expansion functions

$$\tilde{T}_j(x) = e^{-\varepsilon^2 x^2} T_j(x). \tag{4.18}$$

The scale factors and coefficients in this case are

$$d_j = \frac{2\varepsilon^{2j}}{j!} \quad \text{and} \quad c_j(x_k) = t_j e^{-\varepsilon^2 x_k^2}\, x_k^j\, {}_0F_1([\,], j+1, \varepsilon^4 x_k^2). \tag{4.19}$$

In three dimensions, starting from (4.12), the final form becomes

$$\phi_k(\underline{x}) = \sum_{j=0}^{\infty} \sum_{m=0}^{\frac{j-p}{2}} d_{j,m} \sum_{\nu=-(2m+p)}^{2m+p} c_{j,m,\nu}(\underline{x}_k)\, T_{j,m,\nu}(\underline{x}), \tag{4.20}$$

with the spherical–Chebyshev expansion functions

$$T_{j,m,\nu}(\underline{x}) = e^{-\varepsilon^2 r^2} r^{2m}\, Y^{\nu}_{2m+p}(\theta, \phi)\, T_{j-2m}(r), \tag{4.21}$$

where the spherical coordinates are defined with θ as the colatitude, i.e., $\theta = 0$ at the north pole, and

$$\begin{aligned} Y^{\nu}_{\mu}(\theta, \phi) &= P^{\nu}_{\mu}(\cos\,\theta)\cos(\nu\phi), \quad \nu = 0, \ldots, \mu, \\ Y^{-\nu}_{\mu}(\theta, \phi) &= P^{\nu}_{\mu}(\cos\,\theta)\sin(\nu\phi), \quad \nu = 1, \ldots, \mu, \end{aligned}$$

where $P_\mu^\nu(z)$ are the normalized associated Legendre functions. The scale factors are

$$d_{j,m} = 2^{3+p+4m}\varepsilon^{2j}\frac{(\frac{j+p+2m}{2})!}{(\frac{j-p-2m}{2})!(j+1+p+2m)!}, \tag{4.22}$$

and the coefficients become

$$c_{j,m,\nu}(\underline{x}_k) = t_{j-2m}\, y_\nu\, e^{-\varepsilon^2 r_k^2}\, r_k^j\, Y_{2m+p}^\nu(\theta_k,\phi_k)\, {}_2F_3(\rho_{j,m},\sigma_{j,m},\varepsilon^4 r_k^2), \tag{4.23}$$

where $y_0 = \frac{1}{2}$ and $y_\nu = 1$, $\nu > 0$, and the parameters to the hypergeometric function are $\rho_{j,m} = \left[\frac{j-2m+1}{2}, \frac{j-2m+2}{2}\right]$ and $\sigma_{j,m} = \left[j-2m+1, \frac{j-2m-p+2}{2}, \frac{j+2m+p+3}{2}\right]$.

4.2. The RBF-QR algorithm in two dimensions expressed in matrix form. Consider Gaussian RBFs centered at N different node points $\underline{x}_k$, $k = 1,\dots,N$, evaluated at a general point $\underline{x} = (x,y) = (r,\theta)$. Then we have the relation

$$\begin{bmatrix} \phi_1(\underline{x}) \\ \vdots \\ \phi_N(\underline{x}) \end{bmatrix} = C\cdot D\cdot \begin{bmatrix} T_{0,0}^c(\underline{x}) \\ T_{1,0}^c(\underline{x}) \\ T_{1,0}^s(\underline{x}) \\ T_{2,0}^c(\underline{x}) \\ T_{2,1}^c(\underline{x}) \\ T_{2,1}^s(\underline{x}) \\ \vdots \end{bmatrix}, \quad \text{or} \quad \Phi(\underline{x}) = C\cdot D\cdot T(\underline{x}),$$

where C is a rectangular matrix containing the coefficients $c_{j,m}$ and $s_{j,m}$ and D is a diagonal matrix with the scaling coefficients $d_{j,m}$. By QR-factorizing the coefficient matrix C, we get the corresponding relation

$$\begin{aligned}\Phi(\underline{x}) &= Q\cdot R\cdot D\cdot T(\underline{x}) = Q\cdot \begin{bmatrix} R_1 & R_2 \end{bmatrix}\begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}\cdot T(\underline{x}) \\ &= Q\cdot \begin{bmatrix} R_1D_1 & R_2D_2 \end{bmatrix}\cdot T(\underline{x}),\end{aligned}$$

where R_1 is upper triangular and both R_1 and D_1 are $N\times N$.

The original basis $\{\phi_k(\underline{x})\}_{k=1}^N$ is not a good choice for small ε. We have chosen the expansion functions $T_{j,m}$ to be better conditioned and insensitive to the value of ε. Accordingly, we want to change the basis to be more similar to the expansion functions, and we are allowed to take any linearly independent combination of $\{\phi_k(\underline{x})\}_{k=1}^N$ without changing the approximation space. We choose a new basis

$$\Psi(\underline{x}) = D_1^{-1}R_1^{-1}Q^H\Phi(\underline{x}) = \begin{bmatrix} I & D_1^{-1}R_1^{-1}R_2D_2 \end{bmatrix}\cdot T(\underline{x}) = \begin{bmatrix} I & \tilde{R} \end{bmatrix}\cdot T(\underline{x}),$$

which has a part corresponding exactly to the expansion functions plus a correction part. The correction $\tilde{R}$ has to be computed with some care. First $R_1^{-1}R_2$ is computed through backward substitution. Then the scaling effects of D_1^{-1} and D_2 should be combined analytically to avoid over- and/or underflow. All dangerous effects of the leading powers of ε are contained in D_1 and D_2, but the resulting effect in $\tilde{R}$ is harmless. Schematically, the elements are subjected to a scaling with the following

block structure:

$$\left[\begin{array}{c|c|c|c} \vdots & & & \\ \hline \mathcal{O}(\varepsilon^4) & \vdots & & \\ \hline \mathcal{O}(\varepsilon^2) & \mathcal{O}(\varepsilon^4) & \cdots & \\ \hline \mathcal{O}(\varepsilon^0) & \mathcal{O}(\varepsilon^2) & \mathcal{O}(\varepsilon^4) & \cdots \end{array}\right],$$

where the lowest power of ε is 0 (as above) or 2.

To interpolate or approximate data f_j, given at locations $\underline{y}_j$, $j = 1, \ldots, M$, we need to solve $A\underline{\lambda} = \underline{f}$, where $a_{i,j} = \psi_j(\underline{y}_i)$. The matrix A is computed as

$$A = [\Psi(\underline{y}_1) \cdots \Psi(\underline{y}_M)]^T = [T(\underline{y}_1) \cdots T(\underline{y}_M)]^T \begin{bmatrix} I \\ \tilde{R}^T \end{bmatrix} = T_1^T + T_2^T \tilde{R}^T,$$

where T_1 contains the first N expansion functions evaluated at $\underline{y}_j$, $j = 1, \ldots, M$, and T_2 the remaining expansion functions evaluated at the same locations. When we have solved for $\underline{\lambda}$, the RBF approximant can be evaluated at any location $\underline{x}$ as

$$s(\underline{x}, \varepsilon) = \Psi(\underline{x})^T \underline{\lambda}.$$

The matrix versions of the RBF-QR algorithms in one and three dimensions are completely analogous to the 2-D case.

5. Numerical implementation. The RBF-QR algorithm can be implemented in fewer than 100 lines of MATLAB code. Such an implementation for the 2-D case is provided for reference in Appendix A. The basic steps in the algorithm are the following:

1. Determine where to truncate the expansions of the RBFs.
2. Given $\{\underline{x}_k\}_{k=1}^N$ and ε, form the matrix C. Also generate index vectors describing the scaling matrix D.
3. Factorize $C = QR$, and then compute $\tilde{R}$.
4. Evaluate the expansion functions $T_{j,m}$ at $\{\underline{x}_k\}_{k=1}^N$, and compute the interpolation matrix A.
5. Given data $\underline{f}$, solve $A\underline{\lambda} = \underline{f}$.
6. To find the solution at a point $\underline{x}$, evaluate $\Psi(\underline{x})$ using the expansion functions, and form a linear combination using the coefficients $\underline{\lambda}$.

There are some practical issues to consider in the implementation, and we briefly comment on these here. First of all, the infinite expansion (4.13) must be truncated at some $j = j_{\max}$. The total number of terms retained, M, depends on $j_{\max}$ as

$$M = \begin{pmatrix} j_{\max} + d \\ d \end{pmatrix}, \tag{5.1}$$

where d is the number of dimensions. The number of terms M is also the number of columns in the matrix C, the size of the matrix D, and the number of rows in $T(\underline{x})$. The truncation is based on the scaling applied to the correction matrix $\tilde{R}$. This scaling consists of a multiplication by D_1^{-1} from the left and by D_2 from the right.

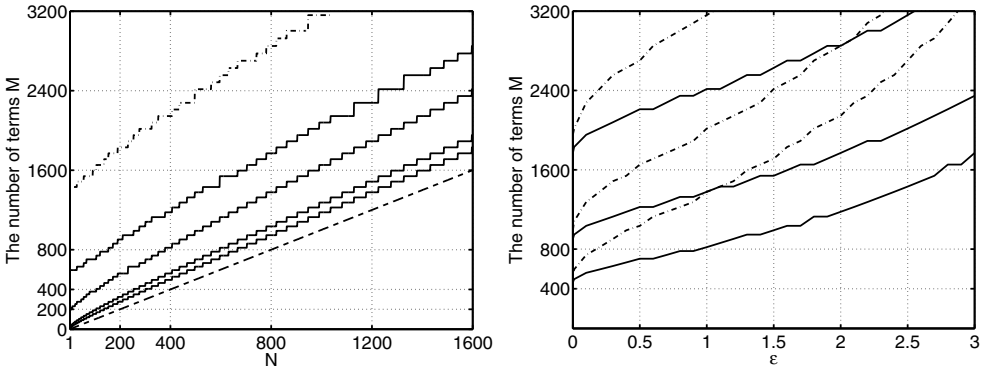

FIG. 5.1. *The number of expansion terms M as a function of N (left) and ε (right) in two dimensions. Solid lines correspond to double precision, and dash-dot lines correspond to quad precision. To the left, $\varepsilon = 0.01$, 0.1, 1, and 2 for double precision (bottom to top) and $\varepsilon = 2$ for quad precision. The dashed line shows $M = N$. To the right, $N = 400$, 800, and 1600.*

The truncation point is determined in such a way that the compound scaling of the largest element in the truncated part is less than the machine precision δ_M, i.e.,

$$\frac{\max\limits_{i>M} D_{ii}}{\min\limits_{1\le i\le N} D_{ii}} < \delta_M. \tag{5.2}$$

Formulas for finding $j_{\max}$ are provided in Appendix B. The validity of the truncation has been tested numerically, and the given formulas were found to accurately predict which terms influence the final result. Figure 5.1 shows how the number of terms M depends of N and ε in two dimensions.

Another practical issue arises because the Chebyshev polynomials are defined for $r \in [-1, 1]$. Therefore, the computational domain under consideration must be scaled in such a way that both node points and evaluation points fall within $[-1, 1]$ in one dimension, the unit disk in two dimensions, and the unit sphere in three dimensions. In the provided reference code, we assume that this scaling has been performed beforehand. Furthermore, we assume that the point locations are given in polar coordinates.

For Gaussian RBFs, the limit interpolant as the shape parameter $\varepsilon \to 0$ exists for any distinct set of node points [9], [26]. However, the polar–Chebyshev expansion in two dimensions and the spherical–Chebyshev expansion in three dimensions can lead to a singular interpolation matrix for certain (nonunisolvent) node configurations, such as all points on a line (see [14] for examples like this for other types of RBFs). However, this problem can be overcome by including "selective" column pivoting in the QR-factorization. Selective indicates that we preserve the order of the basis functions as far as possible since the ordering is linked to the magnitude of the scaling coefficients in the matrix D. Furthermore, we can replace only columns that are exactly linearly dependent. Otherwise, the remaining small nonzero components in the corresponding column in $\tilde{R}$ are scaled with a negative power of ε, causing divergence as $\varepsilon \to 0$. Determining a criterion for exact linear dependence in floating point arithmetic is a delicate problem, and this is not included in the MATLAB code provided here. A version with pivoting (significantly slower due to an extra for-loop) can be downloaded from the second author's web site. It should be noted that nonuni-

solvent node configurations are rare unless the node points are based on some special structure.

6. Numerical experiments. In this section, we present computational results for the RBF-QR method. We will first cover node clustering, which is shown to have a significant influence on the performance for large N. Then the RBF-QR approach is compared with RBF-Direct, and finally we give some examples of the computational cost for using RBF-QR.

The numerical experiments in one and three dimensions are performed in MATLAB. For the 2-D algorithm, a Fortran 90 implementation was used, which enabled us to run tests both with double (64 bit) and quad (128 bit) floating point precision. Comparisons are performed against the RBF-Direct method and also the Contour-Padé approach [13], which was the first method allowing stable computation for small ε.

In order to display the stability and accuracy of the respective methods, a number of smooth test functions have been selected. Even though the methods also work for less smooth functions; these have been excluded since for these it is rarely advantageous to use small ε, which is the shape parameter range we are addressing here. The first function is constant, and then the amount of variation is gradually increased. All function values lie within the range $[-1, 1]$. For experiments in one and two dimensions, the functions are evaluated with $y = z = 0$ or $z = 0$. The functions are

$$
\begin{aligned}
f_1(x, y, z) &= 1, \\
f_2(x, y, z) &= \frac{165}{165 + (x - 0.2)^3 + 2(y + 0.1)^3 + 0.5z^3}, \\
f_3(x, y, z) &= \exp(-(x - 0.1)^2 - 0.5y^2 + 2z^2), \\
f_4(x, y, z) &= \sin(x^2 + 2y^2) - \sin(2x^2 + (y - 0.5)^2 + z^2), \\
f_5(x, y, z) &= \sin(2\pi(x - y - 0.5z)), \\
f_6(x, y, z) &= \sin(2\pi(x^2 + 2y^2)) - \sin(2\pi(2x^2 + (y - 0.5)^2 + z^2)).
\end{aligned}
$$

Figure 6.1 shows f_1–f_6 evaluated over the unit disc ($z = 0$).

Errors are evaluated in maximum norm at a uniform/polar/spherical grid not coinciding with the node points. As an example, in two dimensions, we have used polar grids with $N_r = 15$, 20, or 30 points uniformly distributed in the radial direction using $r_k = \frac{2k-1}{2}h$, $k = 1, \ldots, N_r$, with $h = \frac{2}{2N_r - 1}$, and $N_\theta = 40$, 60, or 80 points uniformly distributed in the angular direction. No significant differences in the experimental results were observed for the different choices of evaluation grids.

For those experiments in two dimensions where the computational domain is not the unit disc, the evaluation points are restricted to fall within the computational domain Ω. Hence, the displayed results approximate the error

$$E(\varepsilon) = \max_{\underline{x} \in \Omega} |s(\underline{x}, \varepsilon) - f(\underline{x})|.$$

6.1. Node clustering and convergence with N. The starting point for the discussions of this section are the theoretical results given by Platte, Trefethen, and Kuijlaars in [22]. The authors prove that an approximation procedure with exponential convergence for analytic functions, involving uniform node locations, inevitably must lead to exponential ill-conditioning in terms of N.

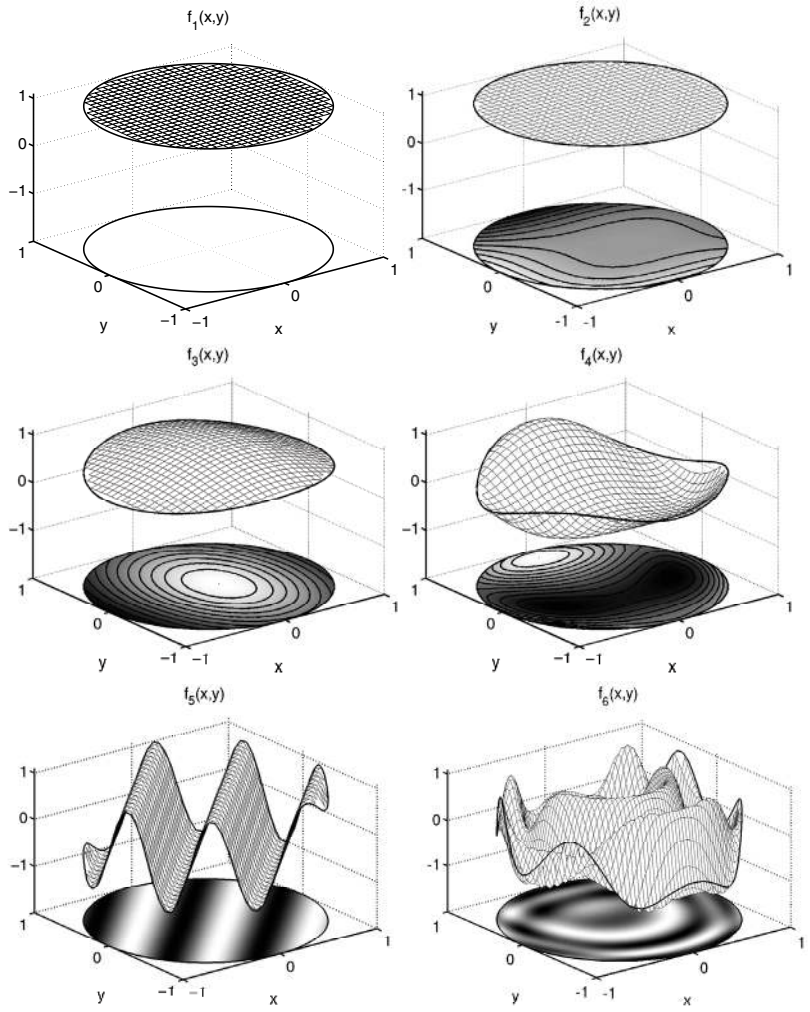

FIG. 6.1. *The test functions* $f_1(\underline{x})$ *to* $f_6(\underline{x})$.

6.1.1. The 1-D case. If we consider RBF approximation in the 1-D case, the exponential convergence is there in theory, both for RBF-Direct [21], [29] and RBF-QR, which are different algorithms to compute the same approximation. The proof in [22] is given for uniform nodes, but in the discussion section the authors argue that the same principle should also hold for quasi-uniform node distributions. The left part of Figure 6.2, where increasing numbers of (quasi-uniform) Halton points [17] are used for interpolation in one dimension, illustrates the result. Errors go down exponentially with N as far as the conditioning allows and then grow exponentially. The exponentially growing part closely matches the loss of accuracy predicted by the conditioning of the expansion function matrix.

By replacing the node points x_k in the Halton sequence with

$$\tilde{x}_k = \sin(\pi x_k/2),$$

we get node points that are clustered similarly to Chebyshev points without being restricted to specific locations. The result for clustered node points is shown in the right part of Figure 6.2. In this case, convergence is limited only by the machine

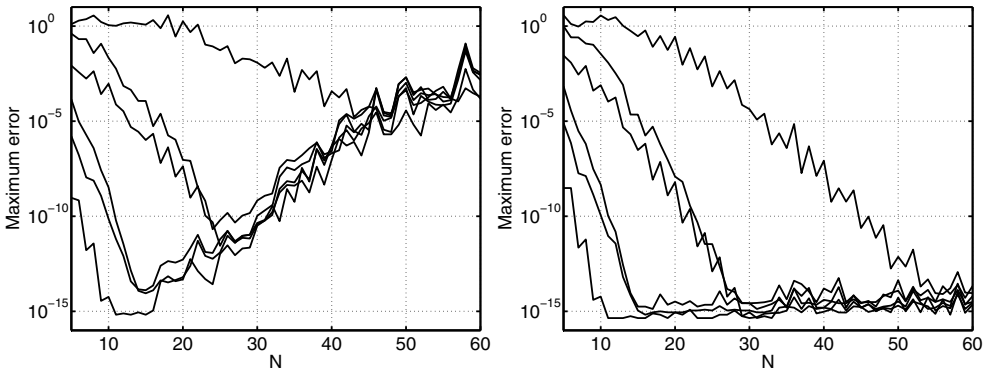

FIG. 6.2. *Interpolation over the interval* $[-1, 1]$ *of* f_1–f_6 *using the RBF-QR approach with Halton points* x_k *(left) and clustered Halton points* $\tilde{x}_k$ *(right). The shape parameter here was* $\varepsilon = 0.1$.

precision. The plot stops at $N = 60$, but, when the same experiment is run for $N = 1000$, the errors are still not more than 10^{-13}.

6.1.2. The 2-D case. Moving to two dimensions, the question of how to cluster arises. Should clustering be connected with the expansion functions, which are Chebyshev polynomials in the radial direction and trigonometric functions in the angular direction, or should the clustering depend only on the computational domain? For the unit disc, either viewpoint leads to the conclusion that clustering is needed in the radial direction. Note that if we conceptually (no practical difference) view the polar coordinates as $r \in [-1, 1]$ across the unit disc and $\theta \in [0, \pi]$, it is apparent that clustering is needed only at the boundary (not at the origin). Starting from (Cartesian) Halton points restricted to the unit disc, switching to polar coordinates, and then modifying the radial coordinate as

$$\tilde{r}_k = \sin(\pi r_k/2),$$

we arrive at the node set used in Figure 6.3. The left and right subfigures show results for two different values of the shape parameter. The shape parameter does affect the size of the error, but the qualitative behavior of the RBF-QR method is similar. The figures show that clustering allows us to compute highly accurate interpolation results for large numbers of points. However, even without clustering, we can, for example, solve for $N = 200$ nodes with an accuracy of 10^{-12}. This is relevant for methods relying on local RBF approximations, where clustering might not be an option.

For the second 2-D experiment, we have chosen a domain that is far from circular. The boundaries of the domain are given by the unit circle and the condition $0 \leq (x-1.2)^2 - 4y^2 \leq 1$. The shape of the domain is shown in the right part of Figure 6.4. The first question to answer is how RBF-QR works when the domain does not coincide with the unit disc over which the expansion functions are defined. In the left part of Figure 6.4, the interpolation results for regular and clustered Halton nodes are shown. In both cases the performance is comparable to the results for the unit disc.

For the clustering, we explored different approaches. Clustering according to the expansion functions, i.e., as for the unit disc, is not the right approach, but it does reduce the error to about 10^{-4} for large N, perhaps because some of the boundaries are at the edge of the unit disc. Next, we tried an ad hoc clustering based on the hyperbolic curves $(x-1.2)^2 - 4y^2 = c^2$ and constant angle measured from an off center location. This gave good results in most parts of the domain, but it did not treat

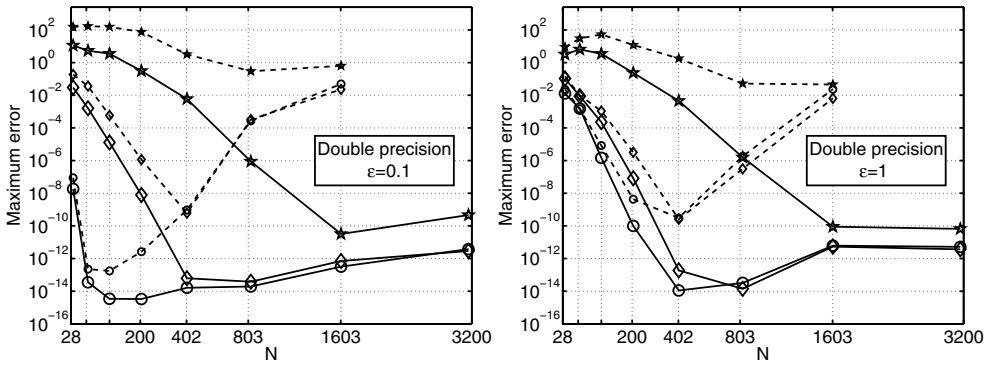

FIG. 6.3. *Interpolation over the unit disc using RBF-QR with Halton nodes (dashed lines) and radially clustered Halton nodes (solid lines). Errors as a function of N are shown for test functions* $f_1(x,y)$ (○), $f_4(x,y)$ (◇), *and* $f_6(x,y)$ (⋆). *The horizontal axis is linear in* $\sqrt{N}$ *(the* 1*-D resolution).*

the three locations where the unit circle constitutes the boundary properly. Hence, the errors were large there, and the overall maximum norm of the error, although lower than in the previous case for intermediate values of N, was again of order 10^{-4} at 3200 nodes. The third approach was successful and corresponds to the result in Figure 6.4. We introduce a coordinate system, shown in the figure, such that all domain boundaries coincide with coordinate lines. This is based on an arc length parameterization along the domain and the hyperbolic constant c^2 across, with a slight modification to accommodate the area around $(r,\theta) = (1, 0)$. Then clustering is performed as before in both of these coordinates.

The conclusion from these experiments is that as long as we can come up with a measure of distance to the boundary and direction to the boundary, where all boundary segments are included, clustering can also be completely successful for irregularly shaped domains.

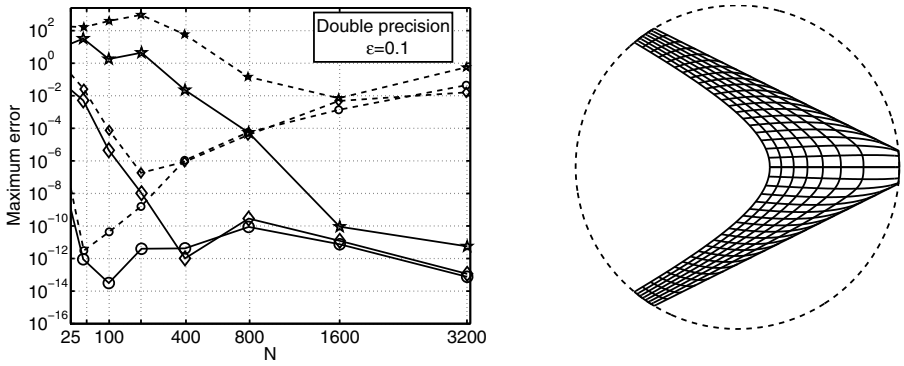

FIG. 6.4. *The left subfigure shows interpolation errors when using RBF-QR with Halton nodes (dashed lines) and hyperbolically clustered Halton nodes (solid lines). Errors as a function of N are shown for test functions* $f_1(x,y)$ (○), $f_4(x,y)$ (◇), *and* $f_6(x,y)$ (⋆). *The horizontal axis is linear in* $\sqrt{N}$. *The right subfigure shows the computational domain and the coordinate lines used for clustering nodes.*

6.1.3. The 3-D case. Figure 6.5 shows results for clustered and quasi-uniform nodes in three dimensions. The deterioration in the quasi-uniform case is notably smaller than in one and two dimensions. For the unit disc and the unit sphere, the potential ill-conditioning arises from the radial direction (the angular directions are periodic). Using $N = 20$ points in one dimension corresponds to polynomial degree $K = 19$. For the corresponding degree in two and three dimensions, we need $N = 210$ and $N = 1540$ points, respectively. Comparing these N-values in Figures 6.2, 6.3, and 6.5 shows that the accuracy is similar and around 10^{-12}. Hence, in three dimensions, we can solve for very large numbers of points, both with regular nodes and clustered nodes.

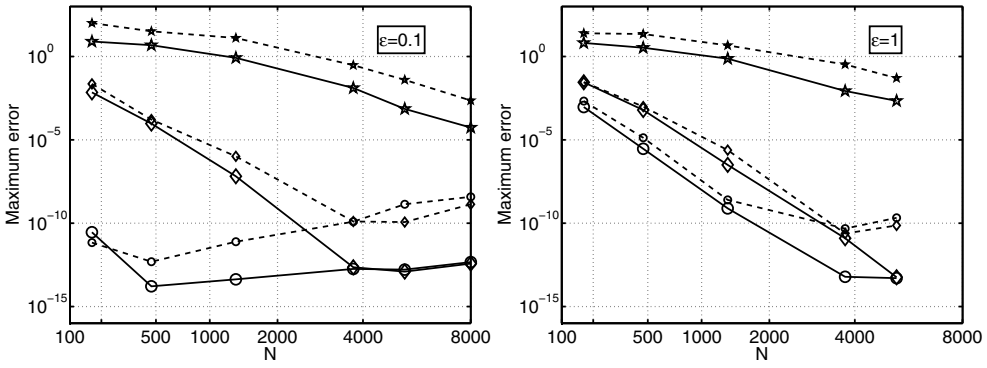

FIG. 6.5. *Interpolation inside the unit sphere using RBF-QR with Halton nodes (dashed lines) and radially clustered Halton nodes (solid lines). Errors as a function of N are shown for test functions $f_1(x, y)$ ($\circ$), $f_4(x, y)$ ($\diamond$), and $f_6(x, y)$ ($\star$). The horizontal axis is linear in $\sqrt[3]{N}$ (the 1-D resolution).*

6.2. Comparisons between RBF-QR and RBF-Direct and convergence with ε. In this section, we limit the investigations to the 2-D case. However, the results are similar also in one and three dimensions. In the previous section, we established that clustering nodes improves the numerical stability of interpolation with RBF-QR significantly. Figure 6.6 shows that for RBF-Direct the error behaviors for quasi-uniform nodes compared with clustered nodes are similar. As can be seen in the left subfigure, for RBF-Direct, the error as a function of N levels off after the point where the interpolant can no longer be stably computed. This is a property that we have observed consistently in all numerical experiments that we have performed here. Even though the condition number of the interpolation matrix grows with N [15], the final result corresponds to the best stable result for that particular choice of ε.

Using higher precision arithmetic can mitigate the effects of ill-conditioning to some extent. In the following experiment, we compare the results for RBF-QR and RBF-Direct using double and quad precision. Figure 6.7 shows results for a fixed N and a range of ε-values. For large ε, RBF-QR and RBF-Direct give the same results, whereas, for small ε, RBF-Direct fails to produce meaningful results unless $f(\underline{x})$ is constant. In the implementation we used, the machine precision is of order 10^{-16} for double precision and 10^{-34} for quad precision; i.e., the difference is 18 orders of magnitude.

The constant function is the only case where the actual error is smaller than the quad machine precision. There, we can in fact observe a difference of 18 orders of

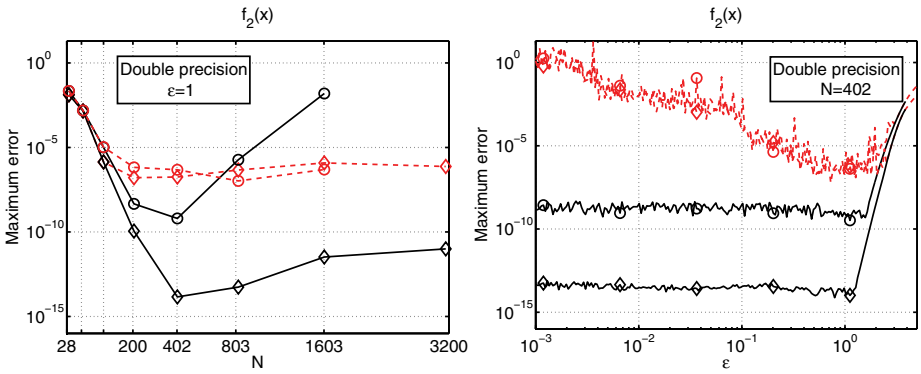

FIG. 6.6. *Interpolation over the unit disc for RBF-QR (solid lines) and RBF-Direct (dashed lines) using Halton nodes (○) and radially clustered Halton nodes (◇) for the test function* $f_2(x)$.

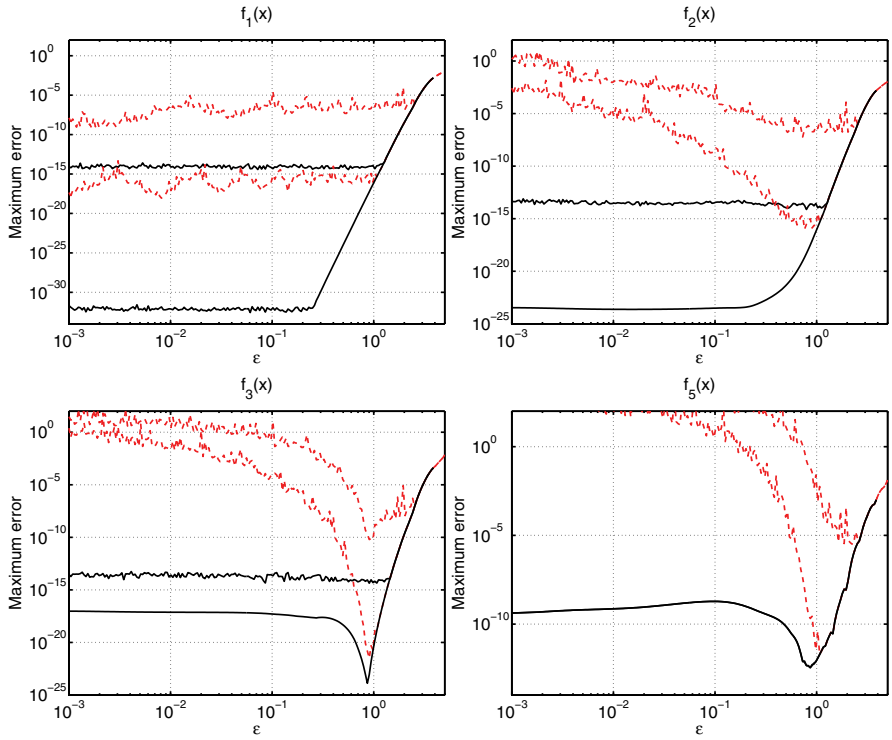

FIG. 6.7. *Interpolation errors as a function of* ε *using RBF-QR (solid lines) and RBF-Direct (dashed lines) for functions* f_1, f_2, f_3, *and* f_5. *In all cases* $N = 402$ *was used. For each method, results computed in double precision (larger errors) and quad precision (smaller errors) are shown. All results were computed using radially clustered Halton nodes. Note that for* $f_5(x)$ *the double precision and quad precision results using RBF-QR are indistinguishable.*

magnitude in the error between double precision and quad precision for RBF-QR. For RBF-Direct, the difference in error is only about 9 orders of magnitude, even though the increase in precision is the same. This is a result of the severe ε-dependence of the conditioning of the interpolation matrix for RBF-Direct. For $N = 402$ points

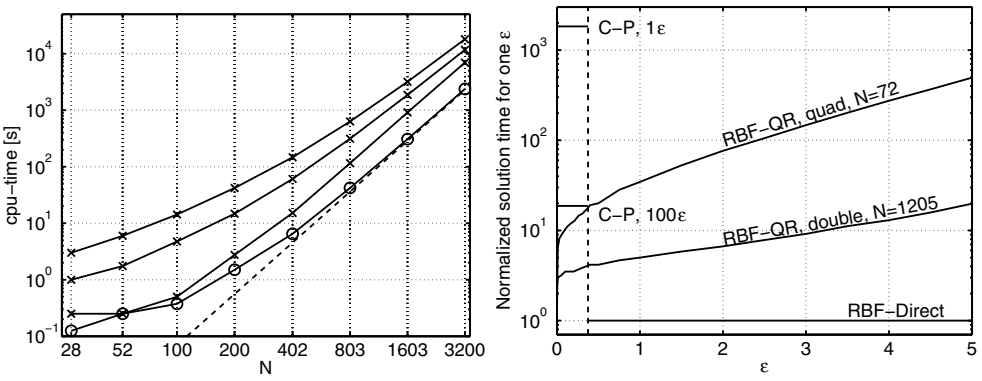

FIG. 6.8. *In the left subfigure, the dashed line indicates the slope of an $\mathcal{O}(N^3)$ algorithm. The CPU time for RBF-Direct (∘) and RBF-QR (×) for $\varepsilon = 0, 0.1, 1$ (bottom to top) are shown. The right subfigure shows the cost per ε-value to compute an interpolant using different algorithms as a function of ε. The costs are scaled so that an RBF-Direct solve in the same precision has unit cost.*

in two dimensions, the condition number is proportional to ε^{-54} [15]. Accordingly, RBF-Direct can trace the actual error curve only up to a certain point, and the obtainable accuracy depends on where this point is in relation to the best ε-value. The conclusion to draw from this is that because RBF-QR is uniformly stable for all small ε-values it pays off to increase the precision, but for RBF-Direct it would be too costly to increase the precision to counteract the ill-conditioning when ε is decreased. Furthermore, for RBF-Direct, the growth rate of the condition number in terms of ε increases with N.

6.3. Computational cost. All timings are performed using the Fortran 90 implementation of the 2-D algorithm. Figure 6.8 illustrates the computational cost of RBF-QR compared with RBF-Direct and Contour-Padé. The left subfigure shows that the computational cost grows as N^3, as expected with direct matrix factorizations. For $\varepsilon = 0$ the cost of the RBF-QR method approaches 3 times the cost of RBF-Direct, which is also expected. Both methods perform an LU-factorization, and RBF-QR performs an additional QR-factorization, which is twice the cost of an LU-factorization. For $\varepsilon = 0.1$ the cost is approximately 5 times larger and for $\varepsilon = 1$ it is 7.6 times larger, asymptotically. The CPU times shown are for quad precision computations. The ratios would be smaller in double precision since the number of columns used by the RBF-QR methods grows with precision. On the specific computer we used, quad precision arithmetic was emulated in software, making it about 70 times slower than double precision arithmetic. When available in hardware, a speed ratio of 4 would be more typical. With the problem sizes under consideration, it was still feasible to use. Furthermore, the evaluation matrices can be precomputed and stored for repeated use.

The right subfigure shows how the computational cost depends on the shape parameter. All times are normalized against the cost for RBF-Direct using the same precision. The Contour-Padé method [13] can be used only for small ε and N up to about 70. It has a large initial cost, but evaluating for many ε-values is almost free, as indicated by the differing times per ε-value when computing for 100 values and for only one value. The cost of RBF-QR grows with ε, but the growth is much less pronounced for larger N and for double precision.

7. Conclusions. We have derived an RBF-QR method for interpolation or approximation with Gaussian RBFs in up to three space dimensions.

The algorithm is numerically stable for small shape parameters, all the way to $\varepsilon = 0$. The conditioning of the interpolation matrix grows with the problem size N if nodes are uniformly distributed. However, the growth is less pronounced in higher dimensions. If nodes are instead clustered towards the boundary, RBF-QR can be used for problem sizes in the thousands without any significant loss of accuracy.

The algorithm is not very sensitive to the shape of the computational domain, but, in order to avoid ill-conditioning for larger N, clustering towards the (irregular) boundary must be performed. In the case of nonunisolvent node layouts, the algorithm must be modified to include column pivoting.

The computational cost is higher than for RBF-Direct, but only by a factor that is asymptotically independent of N and is decreasing when $\varepsilon \to 0$. In the range of intermediate to large values of N, RBF-QR is currently the only numerical algorithm that can deliver an accurate result for small ε.

The RBF-QR method opens up new possibilities for all methods based on local RBF approximation, such as RBF domain decomposition methods [1], [20] and RBF-generated scattered node finite differences [10], [24], [25], [27], [30], since it is now possible to compute for the best shape parameter value even if it lies in the small ε regime.

Appendix A. MATLAB code for the RBF-QR algorithm in two dimensions. The following is a 100 line sample MATLAB code implementing the 2-D case of the RBF-QR algorithm. Implementations of RBF-QR in one, two, and three dimensions together with sample driver routines and subroutines for hypergeometric functions will be available for downloading from the second author's web site.

```
function [u]=RBF_QR_2D(ep,xk,xe,f)
%--- ep (scalar)  : The shape parameter
%--- xk(1:N,1:2)  : The center points in polar coordinates (r_k,theta_k)
%--- xe(1:Ne,1:2) : The evaluation points in polar coordinates
%--- f(1:N,1:Nf)  : The data to interpolate, Nf different functions
%--- u(1:Ne,1:Nf) : The RBF interpolant evaluated at xe for each function
  N = size(xk,1);  Ne = size(xe,1);
  mp = eps; % machine precision
%--- Find out how many columns are needed. (jN+1)(jN+2)/2=N
  jN=ceil(-3/2+sqrt(9/4+2*N-2));
  jmax = 1;  ratio = ep^2/2;
  while (jmax<jN & ratio > 1) % See if d_00 is smaller
    jmax = jmax + 1;
    ratio = ep^2/(jmax+mod(jmax,2))*ratio;
  end
  if (ratio < 1)
    jmax = jN;  ratio = 1;
  end
  ratio = ep^2/(jmax+1+mod(jmax+1,2))*ratio;  % Look one step ahead
  while (ratio*exp(0.223*(jmax+1)+0.212*(1-3.097*mod(jmax+1,2))) > mp)
    jmax = jmax + 1;
    ratio = ep^2/(jmax+1+mod(jmax+1,2))*ratio;
  end
  j = zeros(0,1); m=zeros(0,1); p=zeros(0,1); odd=1;
  for k=0:jmax
    odd = 1-odd;
```

```
    j = [j; k*ones(k+1,1)];        p = [p; odd*ones(k+1,1)];
    q(1:2:k+1,1) = (0:(k-odd)/2)'; q(2:2:k+1,1) = ((1-odd):(k-odd)/2)';
    m = [m;q(1:k+1)];
  end
  co_si = zeros(size(j));
  pos = find(2*m+p>0);
  co_si(pos(1:2:end))=1;    co_si(pos(2:2:end))=2;
 %--- Precompute T_j(r), cos/sin(m*theta), and powers of r
 %--- Note the usage of outer products in the arguments
  Tk = cos(acos(xk(:,1))*(0:jmax));  Te = cos(acos(xe(:,1))*(0:jmax));
  Hkc = cos(xk(:,2)*(1:jmax));       Hec = cos(xe(:,2)*(1:jmax));
  Hks = sin(xk(:,2)*(1:jmax));       Hes = sin(xe(:,2)*(1:jmax));
  Pk = ones(N,jmax+1);
  for k=1:jmax
    Pk(:,k+1) = xk(:,1).*Pk(:,k);
  end
  re2 = xe(:,1).^2;  % Only even powers are needed for evaluation points
  Pe = ones(Ne,(jmax-p(end))/2+1);
  for k=1:(jmax-p(end))/2
    Pe(:,k+1) = re2.*Pe(:,k);
  end
%--- Compute the coefficient matrix
  M = length(j);
  rscale = exp(-ep^2*xk(:,1).^2); % Row scaling of C = exp(-ep^2*r_k^2)
  cscale = 2*ones(1,M);           % Column scaling of C = b_{2m+p}t_{j-2m}
  pos = find(2*m+p == 0);   cscale(pos) = 0.5*cscale(pos);
  pos = find(j-2*m == 0);   cscale(pos) = 0.5*cscale(pos);
  C = Pk(:,j+1); % The powers of r_k and then the trig part
  pos = find(co_si == 1);  C(:,pos) = C(:,pos).*Hkc(:,2*m(pos)+p(pos));
  pos = find(co_si == 2);  C(:,pos) = C(:,pos).*Hks(:,2*m(pos)+p(pos));
  C = C.*(rscale*cscale);
  a = (j-2*m+p+1)/2; b=[j-2*m+1 (j+2*m+p+2)/2];
  z = ep.^4*xk(:,1).^2;
  for k=1:M
    C(:,k) = C(:,k).*hypergeom(a(k),b(k,:),z);
  end
%--- QR-factorize the coefficient matrix and compute \tilde{R}
  [Q,R] = qr(C);
  Rt = R(1:N,1:N)\R(1:N,N+1:M);
  p1 = (1:N);    p2 = (N+1):M;   [pp2,pp1]=meshgrid(p2,p1);
  if (M>N)
    D = EvalD(ep,pp1,pp2,j,m,p);
    Rt = D.*Rt;
  end
%--- Evaluate the basis functions and compute the interpolation matrix.
  V = exp(-ep^2*xk(:,1).^2)*ones(1,M).*Pk(:,2*m+1).*Tk(:,j-2*m+1);
  pos = find(co_si == 1);   V(:,pos) = V(:,pos).*Hkc(:,2*m(pos)+p(pos));
  pos = find(co_si == 2);   V(:,pos) = V(:,pos).*Hks(:,2*m(pos)+p(pos));
  A = V(:,1:N) + V(:,N+1:M)*Rt.';
%--- Solve the interpolation problem to obtain the coefficients lambda
  lambda = A\f;
%--- Compute basis functions at evaluation points
  Ve = exp(-ep^2*xe(:,1).^2)*ones(1,M).*Pe(:,m+1).*Te(:,j-2*m+1);
  pos = find(co_si == 1);   Ve(:,pos) = Ve(:,pos).*Hec(:,2*m(pos)+p(pos));
```

```
  pos = find(co_si == 2);   Ve(:,pos) = Ve(:,pos).*Hes(:,2*m(pos)+p(pos));
%--- Evaluate the solution
  B = Ve(:,1:N) + Ve(:,N+1:M)*Rt.';
  u = B*lambda;

function D=EvalD(ep,p1,p2,j,m,p)
%--- Compute the scaling effect of D_1^{-1} and D_2
  sz = size(p1);
  p1 = p1(:);  p2 = p2(:);
  D = ep.^(2*(j(p2)-j(p1))./2.^(j(p2)-j(p1)-2*(m(p2)-m(p1)));
  f1 = (j(p2)+2*m(p2)+p(p2))/2;  f2 = (j(p2)-2*m(p2)-p(p2))/2;
  f3 = (j(p1)+2*m(p1)+p(p1))/2;  f4 = (j(p1)-2*m(p1)-p(p1))/2;
  for k=1:length(D)
    v1 = sort([(f1(k)+1):f3(k) (f2(k)+1):f4(k)]);
    v2 = sort([(f3(k)+1):f1(k) (f4(k)+1):f2(k)]);
    l1 = length(v1);            l2 = length(v2);
    v1 = [ones(1,l2-l1) v1];    v2 = [ones(1,l1-l2) v2];
    D(k) = D(k)*prod(v1./v2);
  end
  D = reshape(D,sz);
```

Appendix B. Expansion and truncation details. Here, we will briefly describe the steps in deriving the final expansion of the Gaussian RBFs in the 2-D case. Then we provide rules for truncating the expansions in the 1-D, 2-D, and 3-D cases.

B.1. Expansion steps. Starting from (4.5), we first expand the powers of the scalar product in polar coordinates, $\underline{x} = (r, \theta)$,

$$(\underline{x} \cdot \underline{x}_k)^j = r^j r_k^j \cos^j(\theta - \theta_k) = \frac{r^j r_k^j}{2^j} \sum_{m=0}^{\frac{j-p}{2}} \binom{j}{\frac{j+p+2m}{2}} \underbrace{b_{2m+p}\Theta_{2m+p}}_{\tilde{\Theta}_{2m+p}}, \tag{B.1}$$

where $p = 0$ if j is even and 1 otherwise, $b_m = 1$ if $m = 0$ and 2 otherwise, and $\Theta_m = \cos(m\theta)\cos(m\theta_k) + \sin(m\theta)\sin(m\theta_k)$ as in (4.10). The next step is to partially convert powers of r to Chebyshev polynomials as described in section 4.1.2:

$$r^j \Theta_{2m+p} = \frac{r^{2m}\Theta_{2m+p}}{2^{j-2m-1}} \sum_{\ell=0}^{\frac{j-p-2m}{2}} \binom{j-2m}{\ell} \underbrace{t_{j-2m-2\ell} T_{j-2m-2\ell}(r)}_{\tilde{T}_{j-2m-2\ell(r)}}, \tag{B.2}$$

where $t_m = \frac{1}{2}$ if $m = 0$ and 1 otherwise. We define the scale factors

$$d_{j,m} = \frac{\varepsilon^{2j}}{2^{j-2m-1}\left(\frac{j+2m+p}{2}\right)!\left(\frac{j-2m-p}{2}\right)!} \tag{B.3}$$

and then use them together with (B.1) and (B.2) inserted into (4.5) to get

$$\begin{aligned} e^{2\varepsilon^2(\underline{x}\cdot\underline{x}_k)} &= \sum_{j=0}^{\infty} r_k^j \sum_{m=0}^{\frac{j-p}{2}} d_{j,m} r^{2m} \tilde{\Theta}_{2m+p} \sum_{\ell=0}^{\frac{j-p-2m}{2}} \binom{j-2m}{\ell} \tilde{T}_{j-2m-2\ell}(r) \\ &= \sum_{j=0}^{\infty} r_k^j \sum_{m=0}^{\frac{j-p}{2}} d_{j,m} r^{2m} \tilde{\Theta}_{2m+p} \tilde{T}_{j-2m}(r) \sum_{\ell=0}^{\infty} \frac{d_{j+2\ell,m}}{d_{j,m}} r_k^{2\ell} \binom{j-2m+2\ell}{\ell}. \end{aligned}$$

By noting that the final sum has the form of a standard Taylor expansion of a ${}_1F_2$ hypergeometric function and rearranging the factors, it can be verified that it is equal to ${}_1F_2(\alpha,\beta,\varepsilon^4 r_k^2)$, where $\alpha = \frac{j-2m+p+1}{2}$ and $\beta = \left(j-2m+1, \frac{j+2m+p+2}{2}\right)$. This hypergeometric function can be computed either through its (rapidly converging) series expansion or by using a library subroutine.

By going back to (4.4) and inserting the derived expression, we arrive at the final expansion of the Gaussian RBF:

$$\psi(\underline{x},\underline{x}_k) = \sum_{j=0}^{\infty} \sum_{m=0}^{\frac{j-p}{2}} d_{j,m} e^{-\varepsilon^2(\underline{x}_k\cdot\underline{x}_k)} r_k^j \, {}_1F_2(\alpha,\beta,\varepsilon^4 r_k^2) \left(e^{-\varepsilon^2(\underline{x}\cdot\underline{x})} r^{2m} \tilde{T}_{j-2m}(r) \tilde{\Theta}_{2m+p} \right).$$

The remaining step is to extract the coefficients, given in (4.15) and (4.16), which should be $\mathcal{O}(1)$, assuming the scale factors were appropriately chosen.

B.2. Truncation rules. As briefly indicated in section 5, truncation of the expansions is carried out in terms of j and based on the magnitude of the scaling coefficients. For more than one dimension, we need to consider the variation within each block with fixed j. We have

$$\min_{0\le m\le \frac{j-p}{2}} d_{j,m} = d_{j,0} = \begin{cases} \frac{\varepsilon^{2j}}{2^{j-1}\left(\frac{j+p}{2}\right)!\left(\frac{j-p}{2}\right)!} & \text{in two dimensions,} \\ \frac{2^{3+p}\varepsilon^{2j}\left(\frac{j+p}{2}\right)!}{\left(\frac{j-p}{2}\right)!(j+p+1)!} & \text{in three dimensions.} \end{cases}$$

If ε is small enough, the scale factors d_j or $d_{j,0}$ are monotonically decreasing with j. The limits are $\varepsilon \le 1$ in one dimension, $\varepsilon \le \sqrt{2}$ in two dimensions, and $\varepsilon \le \sqrt[4]{6}$ in three dimensions. Otherwise, there is a local maximum leading to

$$\min_{1\le i\le N} D_{ii} = \begin{cases} \min(d_0,\, d_{j_N}) & = & \min(1,\, d_{j_N}) & \text{in one dimension,} \\ \min(d_{0,0},\, d_{j_N,0}) & = & \min(2,\, d_{j_N,0}) & \text{in two dimensions,} \\ \min(d_{0,0},\, d_{j_N,0}) & = & \min(8,\, d_{j_N,0}) & \text{in three dimensions,} \end{cases}$$

where j_N indicates the block containing the Nth column. In order to test a block to determine if it is significant, we also need the largest scaling factor in that block. These are well approximated by the following expressions:

$$\max_{0\le m\le \frac{j-p}{2}} d_{j,m} \approx \begin{cases} e^{0.223j+0.212-0.657p}\, d_{j,0} & \text{in two dimensions,} \\ e^{0.223j-0.012-0.649p}\, d_{j,0} & \text{in three dimensions.} \end{cases}$$

Based on these relations, truncation is performed according to criterion (5.2) to obtain $j_{\max}$.

REFERENCES

[1] H. Adibi and J. Es'haghi, *Numerical solution for biharmonic equation using multilevel radial basis functions and domain decomposition methods*, Appl. Math. Comput., 186 (2007), pp. 246–255.

[2] B. J. C. Baxter and S. Hubbert, *Radial basis functions for the sphere*, in Recent Progress in Multivariate Approximation (Witten-Bommerholz, 2000), Internat. Ser. Numer. Math. 137, Birkhäuser, Basel, 2001, pp. 33–47.

[3] J. P. Boyd, *Error saturation in Gaussian radial basis functions on a finite interval*, J. Comput. Appl. Math., 234 (2010), pp. 1435–1441.

[4] M. D. Buhmann, S. Dinew, and E. Larsson, *A note on radial basis function interpolant limits*, IMA J. Numer. Anal., 30 (2010), pp. 543–554.

[5] R. E. Carlson and T. A. Foley, *The parameter R^2 in multiquadric interpolation*, Comput. Math. Appl., 21 (1991), pp. 29–42.
[6] T. A. Driscoll and B. Fornberg, *Interpolation in the limit of increasingly flat radial basis functions*, Comput. Math. Appl., 43 (2002), pp. 413–422.
[7] N. Flyer, *Exact polynomial reproduction for oscillatory radial basis functions on infinite lattices*, Comput. Math. Appl., 51 (2006), pp. 1199–1208.
[8] B. Fornberg, E. Larsson, and N. Flyer, *Stable Computations with Gaussian Radial Basis Functions in 2-D*, Technical report 2009-020, Department of Information Technology, Uppsala University, Uppsala, Sweden, 2009.
[9] B. Fornberg, E. Larsson, and G. Wright, *A new class of oscillatory radial basis functions*, Comput. Math. Appl., 51 (2006), pp. 1209–1222.
[10] B. Fornberg and E. Lehto, *Stabilization of RBF-generated finite difference methods for convective PDEs*, J. Comput. Phys., 230 (2011), pp. 2270–2285.
[11] B. Fornberg and C. Piret, *A stable algorithm for flat radial basis functions on a sphere*, SIAM J. Sci. Comput., 30 (2007), pp. 60–80.
[12] B. Fornberg and C. Piret, *On choosing a radial basis function and a shape parameter when solving a convective PDE on a sphere*, J. Comput. Phys., 227 (2008), pp. 2758–2780.
[13] B. Fornberg and G. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter*, Comput. Math. Appl., 48 (2004), pp. 853–867.
[14] B. Fornberg, G. Wright, and E. Larsson, *Some observations regarding interpolants in the limit of flat radial basis functions*, Comput. Math. Appl., 47 (2004), pp. 37–55.
[15] B. Fornberg and J. Zuev, *The Runge phenomenon and spatially variable shape parameters in RBF interpolation*, Comput. Math. Appl., 54 (2007), pp. 379–398.
[16] M. A. Golberg, C. S. Chen, and S. R. Karur, *Improved multiquadric approximation for partial differential equations*, Eng. Anal. Boundary Elem., 18 (1996), pp. 9–17.
[17] J. H. Halton, *On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals*, Numer. Math., 2 (1960), pp. 84–90.
[18] E. Larsson and B. Fornberg, *A numerical study of some radial basis function based solution methods for elliptic PDEs*, Comput. Math. Appl., 46 (2003), pp. 891–902.
[19] E. Larsson and B. Fornberg, *Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions*, Comput. Math. Appl., 49 (2005), pp. 103–130.
[20] L. Ling and E. J. Kansa, *Preconditioning for radial basis functions with domain decomposition methods*, Math. Comput. Modelling, 40 (2004), pp. 1413–1427.
[21] W. R. Madych and S. A. Nelson, *Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation*, J. Approx. Theory, 70 (1992), pp. 94–114.
[22] R. B. Platte, L. N. Trefethen, and A. B. J. Kuijlaars, *Impossibility of fast stable approximation of analytic functions from equispaced samples*, SIAM Rev., to appear.
[23] S. Rippa, *An algorithm for selecting a good value for the parameter c in radial basis function interpolation*, Adv. Comput. Math., 11 (1999), pp. 193–210.
[24] Y. V. S. S. Sanyasiraju and G. Chandhini, *Local radial basis function based gridfree scheme for unsteady incompressible viscous flows*, J. Comput. Phys., 227 (2008), pp. 8922–8948.
[25] B. Šarler and R. Vertnik, *Meshfree explicit local radial basis function collocation method for diffusion problems*, Comput. Math. Appl., 51 (2006), pp. 1269–1282.
[26] R. Schaback, *Multivariate interpolation by polynomials and radial basis functions*, Constr. Approx., 21 (2005), pp. 293–317.
[27] C. Shu, H. Ding, and K. S. Yeo, *Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier-Stokes equations*, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 941–954.
[28] H. C. Thacher, Jr., *Conversion of a power to a series of Chebyshev polynomials*, Comm. ACM, 7 (1964), pp. 181–182.
[29] H. Wendland, *Scattered Data Approximation*, Cambridge Monogr. Appl. Comput. Math. 17, Cambridge University Press, Cambridge, UK, 2005.
[30] G. B. Wright and B. Fornberg, *Scattered node compact finite difference-type formulas generated from radial basis functions*, J. Comput. Phys., 212 (2006), pp. 99–123.
[31] G. B. Wright and B. Fornberg, *An Algorithm for Stable Computations with Flat Radial Basis Functions*, manuscript.