STABLE COMPUTATIONS WITH GAUSSIAN RADIAL BASIS FUNCTIONS IN 2-D

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

Abstract. Radial basis function (RBF) approximation is an extremely powerful tool for representing smooth functions in non-trivial geometries, since the method is meshfree 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 only been formulated for the surface of the sphere. This paper contains an RBF-QR formulation for planar two-dimensional problems. The algorithm is perfectly stable for arbitrarily small shape parameters and can be used for up to a thousand node points in double precision and for several thousand node points in quad precision. A sample MATLAB code is provided.

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

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

1. Introduction. When using RBFs (radial basis functions), the best accuracy is often achieved when their shape parameter ε is small, meaning that the basis functions are relatively flat. It was for a number of years mistakenly believed 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 completely stably even in the $\varepsilon \to 0$ (increasingly flat) basis function limit: the Contour-Padé method [10] (see [12] for an algorithmically simplified 'Contour-SVD' version) and the RBF-QR method [8]. These two methods are based on different principles, and have different limitations. The Contour-Padé/SVD method is limited to a relatively low number of RBF nodes (N slightly less than a hundred in 2-D, more in 3-D). 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 in cases of planar 2-D domains. For Gaussian RBFs, it works well for N-values close to a thousand. In case we want full double precision accuracy for the RBF interpolant (all the way into the $\varepsilon \to 0$ limit) but are willing to use quadruple precision arithmetic within the algorithm, the upper limit on N becomes around 2700 node points. If accuracy of the order $1 \cdot 10^{-6}$ is sufficient and we are willing to use quad precision initially, the limit goes up to about 6000 node points.

*University of Colorado, Department of Applied Mathematics, 526 UCB, Boulder, CO 80309, USA (fornberg@colorado.edu). The work was supported by the NSF Grants DMS-0611681, ATM-0620068, and DMS-0914647.

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

‡National Center for Atmospheric Research, Institute for Mathematics Applied to Geosciences, 1850 Table Mesa Drive, Boulder, CO 80305, USA (flyer@ucar.edu). The work was supported by the NSF Grant ATM-0620100. The National Center for Atmospheric Research (NCAR) is sponsored by the National Science Foundation.

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	$\phi_d(r)=\frac{J_{d/2-1}(\varepsilon r)}{(\varepsilon r)^{d/2-1}}$

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 2-D algorithm for Gaussian RBFs is derived in Section 4. Implementation details are covered in Section 5 and numerical results are demonstrated in Section 6. The conclusions in Section 7 are followed by three appendices containing (A) a sample MATLAB code, (B) some details about the polar-Chebyshev expansions, and (C) a discussion of other RBFs than Gaussians.

2. The ill-conditioning of the RBF-Direct algorithm. Given a radial basis function $\phi(r)$ 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 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 depends in a well-conditioned way on the data $\{\underline{x}_k, f_k\}$ [4], [11], [17], [20]. 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/SVD 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 [13]. In the case when the nodes $\underline{x}_i$ are scattered in 2-D, 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 $O(\varepsilon^4)$, etc. until all the N eigenvalues are accounted for. The BE radial functions were shown in [5], [7] 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.

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

2-D non-periodic case (GA, MQ, IMQ, IQ, BE$_{d=3,4,5,...}$)							
1	2	3	4	5	6	7	...
2-D non-periodic case (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 non-periodic case (GA, MQ, IMQ, IQ)							
1	3	6	10	15	21	28	...

Much RBF literature has been devoted to finding 'optimal' values for the shape parameter. In some cases, this optimal value occurs in a shape parameter regime when 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 (seemingly unavoidable) accuracy losses for large ε and (completely avoidable) RBF-Direct accuracy losses for low values of ε. Since it is now understood that the ill-conditioning can be bypassed, all such balances need to be re-assessed. An accuracy-limiting factor other than ill-conditioning then emerges in the decreasing ε-regime. This was seen in [8], [9], [16], and shown in [13] to be related to the polynomial Runge phenomenon.

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 non-periodic case. If an RBF is centered at $\underline{x}_i$, its value at $\underline{x}$ (with both points on the surface of the unit sphere) was shown in [3] to be expressible in a sum

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

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 $c_{\mu,\nu}$ [3], [8]. 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 non-zero values when $\varepsilon \to 0$). In view of (3.1), a column vector of the RBFs, centered at the successive nodes, can be written

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

where C is a matrix with entries of size $O(1)$, which remains well-conditioned as $\varepsilon \to 0$. If we multiply from the left with any non-singular 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 $E^{-1}Q^*$, where E denotes the diagonal matrix with the increasing powers of ε and E^{-1} denotes the transpose of E in which all positive powers of ε are replaced be the corresponding negative ones. The product Q^*Q becomes I and the product $E^{-1}RE$ becomes an upper triangular matrix with a diminishing number of significant upper diagonals as ε decreases. The resulting column vector

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

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 use as our new set of basis functions

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

The matrix $E^{-1}R\,E$ is well conditioned, 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^k in (3.1), $\{1,\ 3,\ 5,\ 7,\ \ldots\}$ for $k = 0, 2, 4, \ldots$ 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 just at the same rate as it otherwise would have deteriorated, i.e. the conditioning remains essentially invariant with both N and ε.

4. RBF-QR in the 2-D non-periodic case. Consider a radial function $\phi(r)$ with Taylor expansion $\phi(r) = \sum_{k=0}^{\infty} c_k(\varepsilon r)^{2k}$. If we center it at a point (x_i, y_i), we have $r = \sqrt{(x-x_i)^2 + (y-y_i)^2}$ and its expansion in powers of ε becomes

$$\psi(x, y, x_i, y_i) = \sum_{k=0}^{\infty} c_k \varepsilon^{2k} ((x-x_i)^2 + (y-y_i)^2)^k. \tag{4.1}$$

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

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

corresponding to the sequence

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

The sequence $\{1,\ 4,\ 8,\ 12,\ 16,\ \ldots\}$ does not match either of the sequences shown for 2-D non-periodic cases in Table 2.2. Re-expansion of the radial functions in the monomials (4.2) will therefore not allow the ill-conditioning to be fully eliminated. 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 (x_i, y_i) becomes

$$\begin{aligned}\psi(x,y,x_i,y_i) &= e^{-\varepsilon^2((x-x_i)^2+(y-y_i)^2)} \\ &= e^{-\varepsilon^2(x^2+y^2)} \cdot e^{-\varepsilon^2(x_i^2+y_i^2)} \cdot e^{2\varepsilon^2(x\,x_i+\,y\,y_i)}\end{aligned} \tag{4.4}$$

Only the last factor above mixes (x, y) and (x_i, y_i) values. It has the Taylor expansion

$$e^{2\varepsilon^2(x\,x_i+\,y\,y_i)} = 1 + 2\varepsilon^2(x\,x_i + \,y\,y_i) + \frac{2^2\varepsilon^4}{2!}(x\,x_i + \,y\,y_i)^2 + \dots . \tag{4.5}$$

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 now are

$$e^{-\varepsilon^2(x^2+y^2)} \cdot \{\{1\}, \{x, y\}, \{x^2, xy, y^2\}, \{x^3, x^2y, xy^2, y^3\}, \dots\} \tag{4.6}$$

and (4.3) has become replaced by

$$\begin{array}{rcccccc} \text{power of } \varepsilon & 0 & 2 & 4 & 6 & 8 & \dots \\ \text{number of functions} & 1 & 2 & 3 & 4 & 5 & \dots , \end{array} \tag{4.7}$$

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

Although we will not pursue the 3-D problem here, we can nevertheless note that the counterparts to (4.4) and (4.6) will involve factoring out $e^{-\varepsilon^2(x^2+y^2+z^2)}$ and then the function set corresponding to $\varepsilon^{2\mu}$ will contain the $\frac{1}{2}(\mu+1)(\mu+2)$ independent homogeneous monomials in (x, y, z) of degree μ. Again, the counting will match what is shown for this case in Table 2.2, and the QR concept will become fully applicable.

4.1.1. Conditioning improvement by conversion to polar coordinates. Many of the monomials in (4.6) become nearly linearly dependent when their degrees increase. Some of this will be circumvented if we express (4.4) in polar rather than in (x, y) coordinates. With $\psi(r, \theta, r_i, \theta_i)$ denoting a GA radial function centered at the polar coordinate location (r_i, θ_i), its value at the location (r, θ) follows from rewriting the last line of (4.4) as $e^{-\varepsilon^2 r_i^2} \cdot e^{-\varepsilon^2 r^2} \cdot e^{2\varepsilon^2 r_i r(\cos\theta_i \cos\theta + \sin\theta_i \sin\theta)}$ (cf. the first part of Appendix B):

$$\begin{aligned}\psi(r,\theta,r_i,\theta_i) = 2 \cdot e^{-\varepsilon^2 r_i^2} \cdot e^{-\varepsilon^2 r^2} \cdot [\\ \varepsilon^0 r_i^0 r^0 \quad &\{\frac{1}{2} \cdot \frac{1}{0!0!}\Theta_0\}+ \\ \varepsilon^4 r_i^2 r^2 \quad &\{\frac{1}{2} \cdot \frac{1}{1!1!}\Theta_0 \; + \frac{1}{2!0!}\Theta_2\}+ \\ \varepsilon^8 r_i^4 r^4 \quad &\{\frac{1}{2} \cdot \frac{1}{2!2!}\Theta_0 \; + \frac{1}{3!1!}\Theta_2 + \frac{1}{4!0!}\Theta_4\}+ \\ \dots \; + \\ \varepsilon^2 r_i^1 r^1 \quad &\{ \quad \frac{1}{1!0!}\Theta_1\}+ \\ \varepsilon^6 r_i^3 r^3 \quad &\{ \quad \frac{1}{2!1!}\Theta_1 + \frac{1}{3!0!}\Theta_3\}+ \\ \varepsilon^{10} r_i^5 r^5 \quad &\{ \quad \frac{1}{3!2!}\Theta_1 + \frac{1}{4!1!}\Theta_3 + \frac{1}{5!0!}\Theta_5\}+ \\ &+ \dots]\end{aligned} \tag{4.8}$$

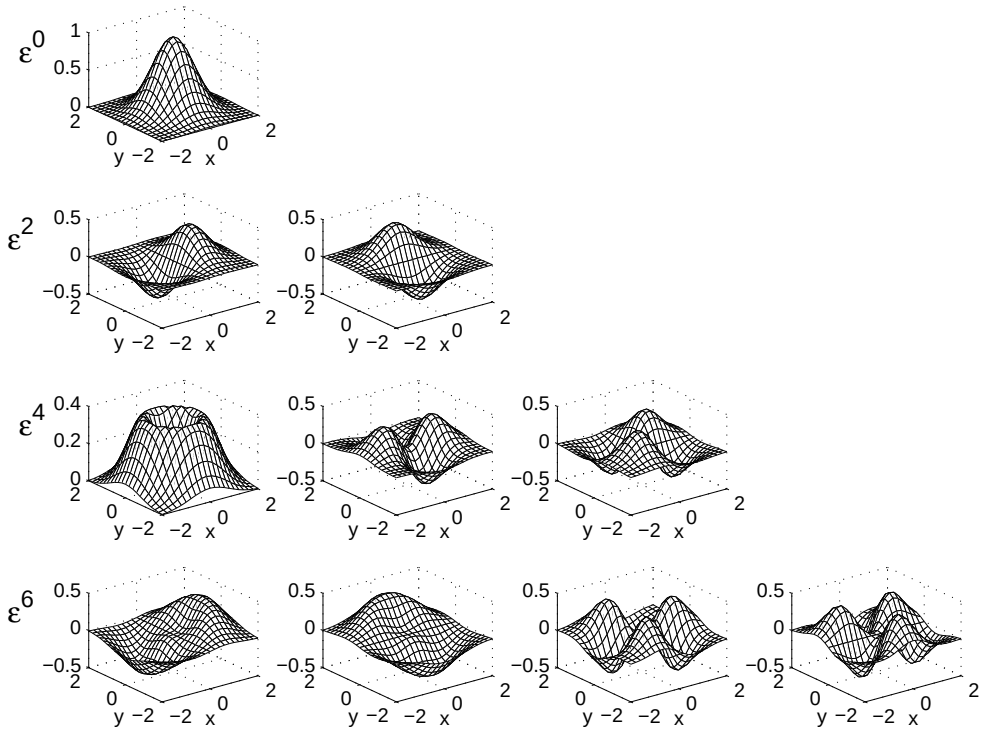

FIG. 4.1. *The expansion functions (4.9) that are needed for representing arbitrary translates of the GA radial function* $\phi(r) = e^{-(\varepsilon r)^2}$*, displayed in the case of* $\varepsilon = 1$.

Here Θ_k abbreviates $(\cos k\theta_i \cos k\theta + \sin k\theta_i \sin k\theta)$. Since their patterns are slightly different, the terms have been split into two groups, containing the powers $\{\varepsilon^0, \varepsilon^4, \varepsilon^8, \ldots\}$ and $\{\varepsilon^2, \varepsilon^6, \varepsilon^{10}, \ldots\}$, respectively. In place of (4.6), we now have

$$e^{-\varepsilon^2 r^2}\{ \quad \{1\}, \\ r\,\{\cos\theta, \quad \sin\theta\}, \\ r^2\{1, \quad \cos 2\theta, \quad \sin 2\theta\}, \\ r^3\{\cos\theta, \quad \sin\theta, \quad \cos 3\theta, \quad \sin 3\theta\}, \\ \ldots \ \} \qquad . \tag{4.9}$$

with (4.7) 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 (4.9). 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, $r^2, \ldots$ for the successive rows in Figure 4.1.

Appendix C contains the expressions corresponding to (4.8) and (4.9) for BE radial functions and a discussion about generalizations to other RBF types.

4.1.2. Further conditioning improvement through use of Chebyshev polynomials. An RBF-QR implementation based on (4.8) performs much better than one based on (4.4) together with (4.5). 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 [21]. 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

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

$$\begin{array}{l|lll}
\varepsilon^2 & T_1\{\cos(\theta),\ \sin(\theta)\} & & \\
\varepsilon^6 & T_3\{\cos(\theta),\ \sin(\theta)\} & r^2T_1\{\cos(3\theta),\ \sin(3\theta)\} & \\
\varepsilon^{10} & T_5\{\cos(\theta),\ \sin(\theta)\} & r^2T_3\{\cos(3\theta),\ \sin(3\theta)\} & r^4T_1\{\cos(5\theta),\ \sin(5\theta)\},
\end{array}$$

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

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

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

$$\phi_k(\underline{x}) = \phi(\|\underline{x} - \underline{x}_k\|) = \sum_{j=0}^{\infty} \sum_{m=0}^{\frac{j-p}{2}} d_{j,m} c_{j,m}(\underline{x}_k) T^c_{j,m}(\underline{x}) + \sum_{j=0}^{\infty} \sum_{m=1-p}^{\frac{j-p}{2}} d_{j,m} s_{j,m}(\underline{x}_k) T^s_{j,m}(\underline{x}), \tag{4.10}$$

where the scale factor $d_{j,m}$ is $\mathcal{O}(\varepsilon^{2j})$ and chosen such that $c_{j,m}$ and $s_{j,m}$ are $\mathcal{O}(1)$. The exact formulas for $c_{j,m}$, $s_{j,m}$ and $d_{j,m}$ are given in Appendix B. Note that conceptually, in the transition to the polar-Chebyshev expansion, we view the coordinates as $r \in [-1, 1]$ across the unit disc and $\theta \in [0, \pi]$. Practically, this has no visible effect and the coordinates are used in the standard sense.

4.2. The RBF-QR algorithm in 2D expressed in matrix form. Consider Gaussian RBFs centered at N different node points $\underline{x}_k$, $k = 1, \ldots, N$ evaluated at a general point $\underline{x} = (x, y) = (r, \theta)$. Then we have the relation

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

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

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

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

$$\Psi(\underline{x}) = D_1^{-1} R_1^{-1} Q^H \Phi(\underline{x}) = \begin{bmatrix} I & D_1^{-1} R_1^{-1} R_2 D_2 \end{bmatrix} \cdot T(\underline{x}) = \begin{bmatrix} I & \tilde{R} \end{bmatrix} \cdot T(\underline{x}),$$

which has a part exactly corresponding 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

$$A = [\Psi(\underline{y}_1) \cdots \psi(\underline{y}_M)]^T = [T(\underline{y}_1) \cdots T(\underline{y}_M)]^T \left[\begin{array}{c} I \\ \hline \tilde{R}^T \end{array}\right] = T_1^T + T_2^T \tilde{R}^T,$$

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

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

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

0. Determine where to truncate the expansions of the RBFs.
1. Given $\{\underline{x}_k\}_{k=1}^N$ and ε, form the matrix C. Also generate index vectors describing the scaling matrix D.
2. Factorize $C = QR$ and then compute $\tilde{R}$.
3. Evaluate the basis functions $T_{j,m}$ at $\{\underline{x}_k\}_{k=1}^N$ and compute the interpolation matrix A.
4. Given data $\underline{f}$, solve $A\underline{\lambda} = \underline{f}$.
5. To find the solution at a point $\underline{x}$, evaluate $\Psi(\underline{x})$ using $T_{j,m}(\underline{x})$ 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 (4.10) must be truncated at some $j = j_{\max}$. The total number of terms retained, M, depends on $j_{\max}$ as

$$M = \frac{1}{2}(j_{\max} + 1)(j_{\max} + 2). \tag{5.1}$$

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. 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.,

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

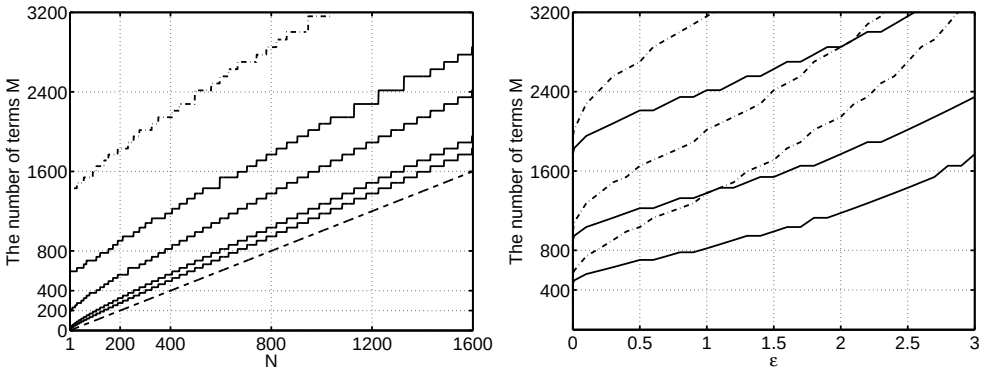

FIG. 5.1. *The number of expansion terms M as a function of N (left) and ε (right). Solid lines correspond to double precision and dash-dot lines correspond to quad precision. To the left, $\varepsilon = 0.01$, 0.1, 1, and 2 for double precision (bottom to top) and $\varepsilon = 2$ for quad precision.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

A formula for finding $j_{\max}$ is provided in Appendix B. The validity of the truncation has been tested numerically and the given formula was 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.

Another practical issue arises because the polar-Chebyshev expansion lives on the unit disk. The computational domain under consideration must be scaled in such a way that both node points and evaluation points fall on the unit disk. 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 [20], [7]. However, the polar-Chebyshev expansion leads to a singular interpolation matrix for certain (non-unisolvent) node configurations, such as all points on a line (see [11] 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 only replace columns that are exactly linearly dependent. Otherwise, the remaining small non-zero 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 non-unisolvent 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. A Fortran 90 implementation was used for the experiments, 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 [10], 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 work also 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 $[-1, 1]$. The functions are

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

Figure 6.1 shows f_1–f_6 evaluated over the unit disc.

Errors are evaluated in maximum norm at a polar grid not coinciding with the node points. When 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

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

For the first set of experiments, we have chosen Halton points [15] restricted to the unit disc. Figure 6.2 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 and 10^{-34} for quad precision, i.e., the difference is 18 orders of magnitude. The results for the constant function is the only case where the actual error is smaller than the limit of obtainable accuracy for the RBF-QR method with quad precision. We can in fact observe a difference of 18 orders of magnitude in the error between double 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 in 2D, the condition number is proportional to ε^{-54} [13]. Accordingly, RBF-Direct can only trace the actual error curve 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. The increase in error seen for functions f_3 and f_5 has to do with a Runge-type phenomenon discussed in [13]. Edge correction methods, as discussed for example in [6] may reduce the effect.

Figure 6.3 shows the same experiment, but now for a fixed ε-value and a range of N. We can make the following observations:

- For each combination of function, shape parameter, and precision in this experiment, RBF-QR yields the most accurate result when the whole range of N-values is considered.

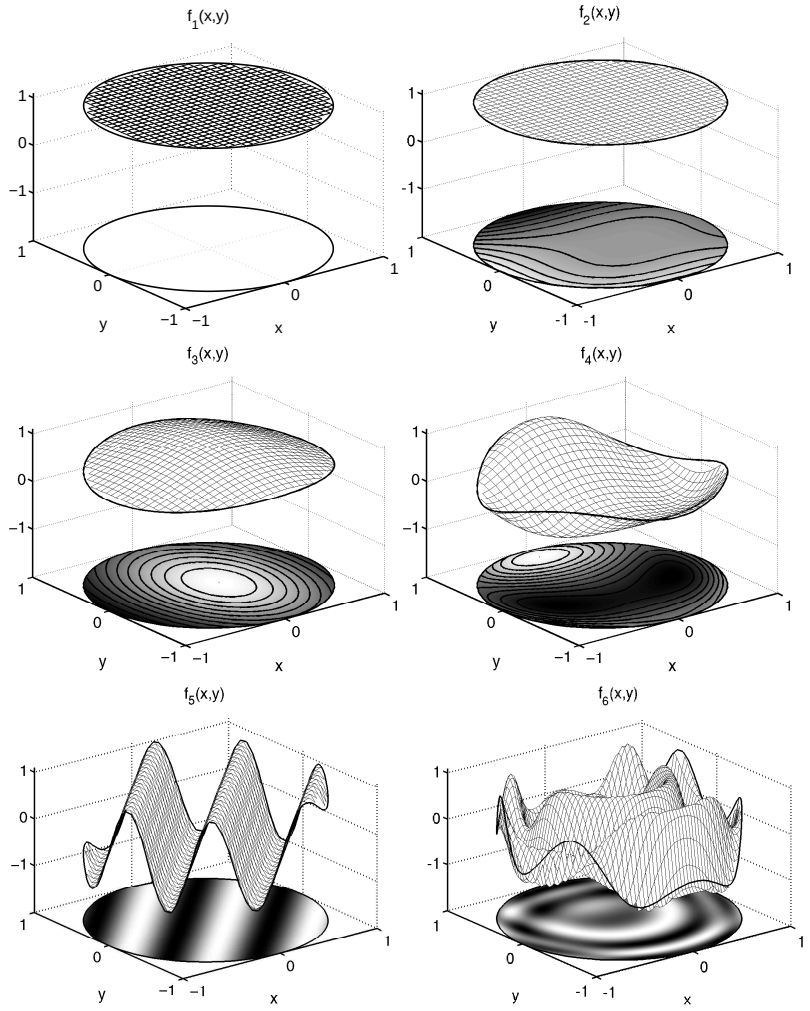

FIG. 6.1. *The test functions* $f_1(\underline{x})$ *to* $f_6(\underline{x})$.

- The limit of obtainable accuracy set by the conditioning depends only on N for the RBF-QR method. The upper limit on N for computations to be of any value is where all significant digits are lost. This happens for $N = 1600$ in double precision and around $N = 7000$ in quad precision. Note that for computations in the small ε-regime these are large numbers of points.
- For RBF-Direct the obtainable accuracy is instead heavily function dependent, favoring the (nearly) constant functions. The limiting N for which computations reproduce the actual error depends on ε and is in the best case here about $N = 200$ for $\varepsilon = 1$. However, it should be noted that for a fixed ε, the error does not grow with N in the catastrophic way it did for decreasing ε in the previous figure.
- Gaussian based RBF-interpolation is spectrally accurate [22] for smooth functions. However, with RBF-Direct it is hard to observe this, since the stable regime is too small. With RBF-QR, convergence patterns are nicely spectral

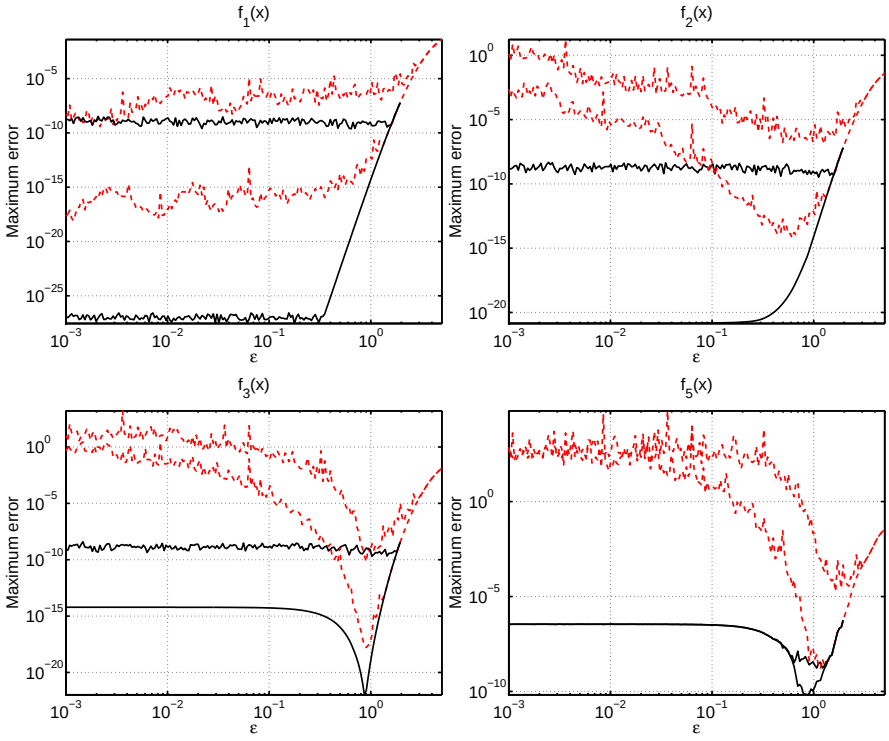

FIG. 6.2. *Interpolation errors as a function of ε using RBF-QR (solid lines) and RBF-Direct (dashed lines) for functions f_1, f_2, f_3, and f_5. In all cases $N = 402$ was used. For each method, results computed in double (larger errors) and quad precision (smaller errors) are shown.*

all the way down to the conditioning limit.

In the previous experiment, the node points were evenly distributed over the unit disc, which is the domain where the Chebyshev polynomials and the trigonometric functions live. Now, we consider a domain that is far from circular. The boundaries of the domain are given by the unit circle and the condition $0 \leq (x-1.2)^2 - 4y^2 \leq 1$. Figure 6.4 shows the interpolation results and the computational domain in this case. The results are very similar, but conditioning is a bit worse than for the unit disc. The trend of the smallest obtainable error seems to improve for larger N, which could be related to the fact that the narrow parts of the domain are poorly sampled when N is small.

Figure 6.5 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 slower than in quad precision. On the specific computer we used, quad precision arithmetic was emulated in software, making it about 70 times slower than double

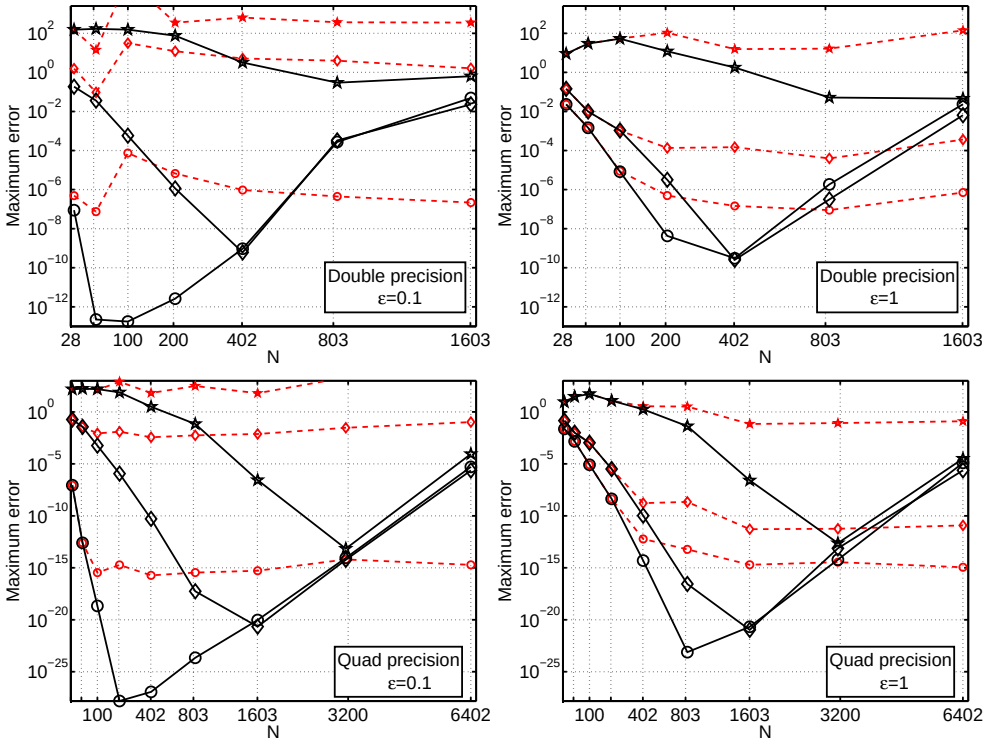

FIG. 6.3. *Interpolation errors when using RBF-QR (solid lines) and RBF-Direct (dashed lines) as a function of N for different combinations of ε and floating point precision. The horizontal axis is linear in $\sqrt{N}$ (the one-dimensional resolution) and $N \approx 25 \cdot 2^j$. Results for test functions $f_1(x,y)$ ($\circ$), $f_4(x,y)$ ($\diamond$), and $f_6(x,y)$ ($\star$) are displayed.*

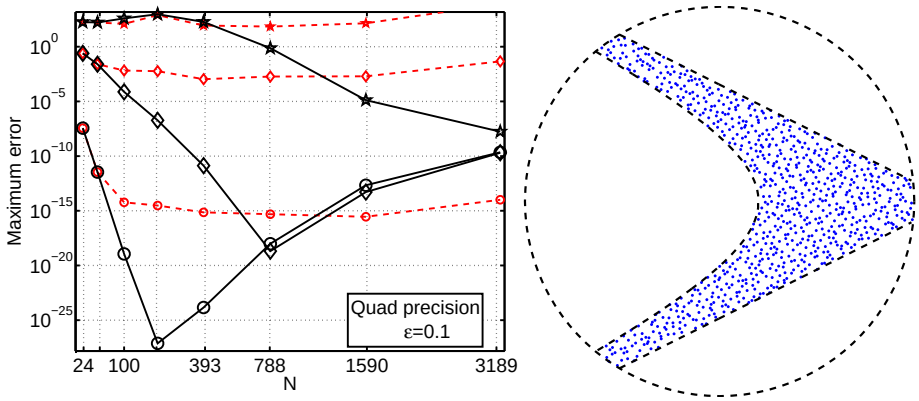

FIG. 6.4. *The left subfigure shows interpolation errors when using RBF-QR (solid lines) and RBF-Direct (dashed lines) as a function of N, computed in quad precision for $\varepsilon = 0.1$ for the functions f_1, f_4, and f_6 indicated by $\circ$,$\diamond$, and $\star$ respectively. (The horizontal axis is linear in $\sqrt{N}$.)*

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 [10] can only be used 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.

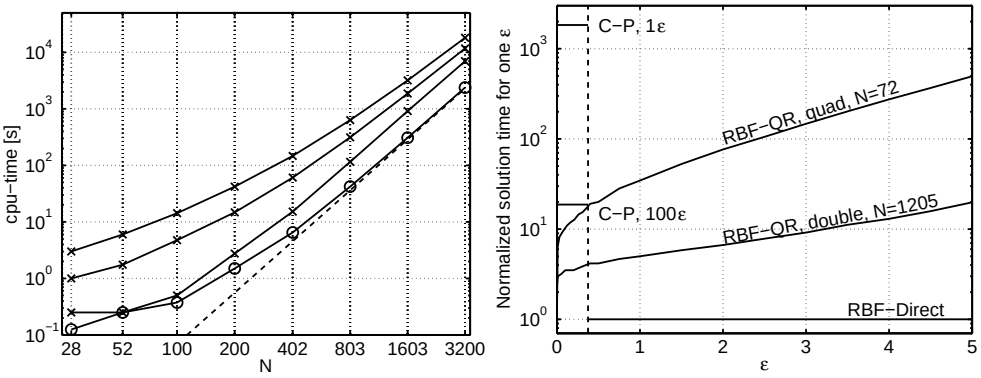

FIG. 6.5. *In the left subfigure, the dashed line indicates the slope of an $\mathcal{O}(N^3)$ algorithm. The cpu-time for RBF-Direct ($\circ$) and RBF-QR ($\times$) for $\varepsilon = 0, 0.1, 1$ (bottom to top) are shown. The right subfigure shows the cost per ε-value to compute an interpolant using different algorithms as a function of ε. The costs are scaled so that an RBF-Direct solve in the same precision has unit cost.*

7. Conclusions. We have derived an RBF-QR algorithm for two-dimensional planar RBF interpolation or approximation with Gaussian RBFs. The algorithm is perfectly numerically stable for small shape parameters, all the way to $\varepsilon = 0$. The conditioning grows moderately with the problem size N. However, we are able to solve for node point numbers up to 1000 in double precision and 6000 in quad precision, which in the context of spectrally convergent interpolation of smooth functions is very large.

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 $100 \leq N \leq 6000$ the RBF-QR method is currently the only algorithm that can deliver an accurate result for small ε.

The algorithm is not very sensitive to the shape of the computational domain, but column pivoting is needed for non-unisolvent node layouts.

The RBF-QR method opens up new possibilities for all methods based on local RBF approximation, such as RBF domain decomposition methods [18], [1] and RBF-generated scattered node finite differences [23], 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. The following is a 100 line sample MATLAB code implementing the RBF-QR algorithm. This code and a version including pivoting together with a short driver routine 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. Details of the polar-Chebyshev expansion. Here, we give a detailed derivation of the expansion functions and expansion coefficients used in the RBF-QR method for Gaussian RBFs in two dimensions. A Gaussian RBF centered at the point $\underline{x}_k$ can be expressed as

$$\phi(\|\underline{x}-\underline{x}_k\|) = \psi(\underline{x},\underline{x}_k) = e^{-\varepsilon^2(\underline{x}-\underline{x}_k)\cdot(\underline{x}-\underline{x}_k)} = e^{-\varepsilon^2(\underline{x}\cdot\underline{x})}e^{-\varepsilon^2(\underline{x}_k\cdot\underline{x}_k)}e^{2\varepsilon^2(\underline{x}\cdot\underline{x}_k)}. \quad \text{(B.1)}$$

The first two factors do not cause any problems in the context of small ε, but the

third factor needs to be expanded. Taylor expansion yields

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

In polar coordinates, $\underline{x} = (r,\theta)$, the powers of the scalar product are

$$\begin{aligned}(\underline{x}\cdot\underline{x}_k)^j &= r^j r_k^j \cos^j(\theta-\theta_k)\\ &= r^j r_k^j \sum_{m=0}^{j}\binom{j}{m}\frac{1}{2^j}\cos\left((j-2m)(\theta-\theta_k)\right)\\ &= r^j r_k^j \sum_{m=0}^{\frac{j-p}{2}}\binom{j}{\frac{j+p}{2}+m}\frac{b_{2m+p}}{2^j}\cos((2m+p)(\theta-\theta_k))\\ &= r^j r_k^j \sum_{m=0}^{\frac{j-p}{2}}\binom{j}{\frac{j+p+2m}{2}}\frac{b_{2m+p}}{2^j}\Theta_{2m+p},\end{aligned} \tag{B.3}$$

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 Equation (4.8). The trigonometric identities that were used to arrive at this expression can for example be found in [19, Section 5.4]. Combining (B.2) and (B.3) yields

$$e^{2\varepsilon^2(\underline{x}\cdot\underline{x}_k)} = \sum_{j=0}^{\infty}\varepsilon^{2j} r^j r_k^j \sum_{m=0}^{\frac{j-p}{2}}\frac{b_{2m+p}\Theta_{2m+p}}{\left(\frac{j+p+2m}{2}\right)!\left(\frac{j-p-2m}{2}\right)!}. \tag{B.4}$$

The next step is to partially convert powers of r to Chebyshev polynomials as described in section 4.1.2.

$$r^j\Theta_{2m+p} = \frac{r^{2m}\Theta_{2m+p}}{2^{j-2m-1}}\sum_{\ell=0}^{\frac{j-p-2m}{2}}\binom{j-2m}{\ell}t_{j-2m-2\ell}T_{j-2m-2\ell}(r), \tag{B.5}$$

where $t_m = \frac{1}{2}$ if $m = 0$ and 1 otherwise. Combining this with the previous expression now yields

$$\begin{aligned}e^{2\varepsilon^2(\underline{x}\cdot\underline{x}_k)} &= \sum_{j=0}^{\infty}\varepsilon^{2j} r_k^j \sum_{m=0}^{\frac{j-p}{2}}\frac{b_{2m+p} r^{2m}\Theta_{2m+p}}{\left(\frac{j+p+2m}{2}\right)!\left(\frac{j-p-2m}{2}\right)!2^{j-2m-1}}\Bigg(\\ &\qquad\sum_{\ell=0}^{\frac{j-p-2m}{2}}\binom{j-2m}{\ell}t_{j-2m-2\ell}T_{j-2m-2\ell}(r)\Bigg)\\ &= \sum_{j=0}^{\infty}\varepsilon^{2j} r_k^j \sum_{m=0}^{\frac{j-p}{2}}\frac{b_{2m+p}t_{j-2m}r^{2m}\Theta_{2m+p}T_{j-2m}(r)}{2^{j-2m-1}\left(\frac{j+p+2m}{2}\right)!\left(\frac{j-p-2m}{2}\right)!}\cdot\\ &\qquad\left(\sum_{\ell=0}^{\infty}\frac{\varepsilon^{4\ell}r_k^{2\ell}}{2^{2\ell}}\binom{j-2m+2\ell}{\ell}\frac{\left(\frac{j+p+2m}{2}\right)!\left(\frac{j-p-2m}{2}\right)!}{\left(\frac{j+p+2m+2\ell}{2}\right)!\left(\frac{j-p-2m+2\ell}{2}\right)!}\right),\end{aligned}$$

where the second equality is obtained by rearranging the sum according to expansion function instead of power of ε. 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

$$\sum_{\ell=0}^{\infty} \varepsilon^{4\ell} r_k^{2\ell} \prod_{q=1}^{\ell} \frac{j+\ell-2m+q}{q(j+2m+p+2q)(j-2m-p+2q)} =$$
$$\sum_{\ell=0}^{\infty} \varepsilon^{4\ell} r_k^{2\ell} \prod_{q=0}^{\ell-1} \frac{\frac{j+p-2m+1}{2}+q}{(j-2m+1+q)(\frac{j+p+2m+2}{2}+q)} = {}_1F_2(\alpha,\beta,\varepsilon^4 r_k^2),$$

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 (B.1) and inserting the derived expression, we arrive at the final expansion of the Gaussian RBF

$$\psi(\underline{x},\underline{x}_k) = \sum_{j=0}^{\infty} \sum_{m=0}^{\frac{j-p}{2}} \frac{\varepsilon^{2j} b_{2m+p} t_{j-2m} e^{-\varepsilon^2(\underline{x}_k\cdot\underline{x}_k)} r_k^j \, {}_1F_2(\alpha,\beta,\varepsilon^4 r_k^2)}{2^{j-2m-1}\left(\frac{j+p+2m}{2}\right)!\left(\frac{j-p-2m}{2}\right)!} \cdot$$
$$\left(e^{-\varepsilon^2(\underline{x}\cdot\underline{x})} r^{2m} T_{j-2m}(r)\Theta_{2m+p}\right).$$

For each expansion function, we extract a scaling coefficient that depends on j and m, but not on $\underline{x}_k$, and is chosen such that the remaining coefficient is $\mathcal{O}(1)$. It is given by

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

and the coefficients for the respective basis functions are given by

$$c_{j,m}(\underline{x}) = b_{2m+p} t_{j-2m} e^{-\varepsilon^2(\underline{x}_k\cdot\underline{x}_k)} r_k^j \cos((2m+p)\theta_k) \, {}_1F_2(\alpha,\beta,\varepsilon^4 r_k^2),$$
$$s_{j,m}(\underline{x}) = b_{2m+p} t_{j-2m} e^{-\varepsilon^2(\underline{x}_k\cdot\underline{x}_k)} r_k^j \sin((2m+p)\theta_k) \, {}_1F_2(\alpha,\beta,\varepsilon^4 r_k^2).$$

The truncation of the expansion needed for the numerical implementation is carried out in terms of j and based on the magnitude of the scaling coefficients. The variation within a block with a fixed j is estimated and then the blocks are compared with each other. For each block we have

$$\min_{0\le m\le\frac{j-p}{2}} d_{j,m} = d_{j,0} = \frac{\varepsilon^{2j}}{2^{j-1}\left(\frac{j+p}{2}\right)!\left(\frac{j-p}{2}\right)!}.$$

The magnitude of the largest scaling coefficient for a fixed j is well approximated by

$$\max_{0\le m\le\frac{j-p}{2}} d_{j,m} \approx e^{0.223j+0.212(1-3.097p)} \, d_{j,0}.$$

If $\varepsilon \le \sqrt{2}$, the coefficients $d_{j,0}$ are decreasing with j, but if $\varepsilon > \sqrt{2}$ there is a local maximum, leading to

$$\min_{1\le i\le N} D_{ii} = \min(d_{0,0},\, d_{j_N,0}) = \min(2,\, d_{j_N,0}),$$

where $j_N = \lceil -\frac{3}{2} + \sqrt{\frac{9}{4} + 2N - 2} \rceil$ (see equation (5.1)) corresponds to the block containing the Nth column. The first 'insignificant' block has a j such that

$$\frac{e^{0.223j+0.212(1-3.097p)}\, d_{j,0}}{\min(d_{0,0},\, d_{j_N,0})} < \delta_M,$$

and truncation is effected right before this block. The ratios can be evaluated recursively by using

$$\frac{d_{j+1,0}}{d_{j,0}} = \frac{\varepsilon^2}{j-p+2}.$$

Appendix C. Expansions for other radial functions than GA.

C.1. BE radial functions. This class of functions was studied extensively in [7], [5]. In particular, $\phi_d(r)$ always gives a non-singular interpolant when used in D dimensions whenever $d \geq \max(D, 2)$. Since the $d = 2$ case differs fundamentally from $d > 2$, we start by discussing that case separately.

C.1.1. BE with $d = 2$. With $\psi(r, \theta, r_i, \theta_i)$ denoting a $J_0(\varepsilon r)$ function centered at the polar coordinate location (r_i, θ_i), its value at the location (r, θ) becomes

$$\psi(r,\theta,r_i,\theta_i) = J_0(\varepsilon r_i)J_0(\varepsilon r) + \\ + 2\sum_{k=1}^{\infty} \varepsilon^{2k} r_i^k r^k \left(\frac{J_k(\varepsilon r_i)}{(\varepsilon r_i)^k}\right)\left(\frac{J_k(\varepsilon r)}{(\varepsilon r)^k}\right)(\cos k\theta_i \cos k\theta + \sin k\theta_i \sin k\theta)\,, \tag{C.1}$$

where we can note that $\left(\frac{J_k(\varepsilon r_i)}{(\varepsilon r_i)^k}\right)$ and $\left(\frac{J_k(\varepsilon r)}{(\varepsilon r)^k}\right)$ are completely regular around $\varepsilon = 0$, approaching $\frac{1}{2^k k!}$ when $\varepsilon \to 0$. The formula (C.1) is the simplest special case of Gegenbauer's addition formulas for Bessel functions (dating back to 1874). For derivations, see for ex. Section 4.10 in [2] or Chapter VI, §3 in [14]. The expansion functions at levels ε^0, ε^2, ε^4,... thus become

$$\{J_0(\varepsilon r)\},\, r\,\frac{J_1(\varepsilon r)}{(\varepsilon r)^1}\{\cos\theta, \sin\theta\},\, r^2\frac{J_2(\varepsilon r)}{(\varepsilon r)^2}\{\cos 2\theta, \sin 2\theta\},\, r^2\frac{J_3(\varepsilon r)}{(\varepsilon r)^3}\{\cos 3\theta, \sin 3\theta\}, \ldots \tag{C.2}$$

respectively, and there is a perfect match with the sequence $\{1, 2, 2, 2, 2, \ldots\}$ that was shown for this case in Table 2.2. However, as was noted in [7], each translate of a 2-D BE $\phi_2(r)$ radial function, and therefore any interpolant $s(x, y)$ that is obtained by linearly combining these, will satisfy $\frac{\partial^2 s}{\partial x^2} + \frac{\partial^2 s}{\partial y^2} + \varepsilon^2 s = 0$. Such a function cannot have a local maximum at a point where s is negative, making it useless for general interpolation in 2-D. The $\phi_2(r)$ case is here of interest to us only because the series expansion (C.1) can be generalized to cases with $d > 2$, which do not suffer from a corresponding flaw in their ability to approximate 2-D functions.

C.1.2. BE with $d > 2$. The Gegenbauer addition formulas can again be applied. The equations (4.10.5) and (6.4.11) in [2] show that all cases with $d > 2$ result in very similar expansions, with the result below for $d = 5$ typical. In place of (4.8), we now

obtain

$$
\begin{aligned}
\psi(r,\theta,r_i,\theta_i) = 2\sqrt{2\pi}\cdot[\qquad & \qquad (C.3)\\
& \varepsilon^0 r_i^0 r^0\ \frac{J_{3/2}(\varepsilon r_i)}{(\varepsilon r_i)^{3/2}}\frac{J_{3/2}(\varepsilon r)}{(\varepsilon r)^{3/2}} \quad \frac{3}{2^0}\{\frac{1}{2}\cdot\frac{1!1!}{0!^2 0!^2}\}+\\
& \varepsilon^4 r_i^2 r^2\ \frac{J_{7/2}(\varepsilon r_i)}{(\varepsilon r_i)^{7/2}}\frac{J_{7/2}(\varepsilon r)}{(\varepsilon r)^{7/2}} \quad \frac{7}{2^4}\{\frac{1}{2}\cdot\frac{3!3!}{1!^2 1!^2} + \frac{5!1!}{2!^2 0!^2}\Theta_2\}+\\
& \varepsilon^8 r_i^4 r^4 \frac{J_{11/2}(\varepsilon r_i)}{(\varepsilon r_i)^{11/2}}\frac{J_{11/2}(\varepsilon r)}{(\varepsilon r)^{11/2}}\ \frac{11}{2^8}\{\frac{1}{2}\cdot\frac{5!5!}{2!^2 2!^2} + \frac{7!3!}{3!^2 1!^2}\Theta_2 + \frac{9!1!}{4!^2 0!^2}\Theta_4\}+\\
& .... +\\
& \varepsilon^2\ r_i^1 r^1\ \frac{J_{5/2}(\varepsilon r_i)}{(\varepsilon r_i)^{5/2}}\frac{J_{5/2}(\varepsilon r)}{(\varepsilon r)^{5/2}} \quad \frac{5}{2^2}\ \{ \qquad \frac{3!1!}{1!^2 0!^2}\Theta_1\}+\\
& \varepsilon^6\ r_i^3 r^3 \frac{J_{9/2}(\varepsilon r_i)}{(\varepsilon r_i)^{9/2}}\frac{J_{9/2}(\varepsilon r)}{(\varepsilon r)^{9/2}} \quad \frac{9}{2^6}\ \{ \qquad \frac{5!3!}{2!^2 1!^2}\Theta_1 + \frac{7!1!}{3!^2 0!^2}\Theta_3\}+\\
& \varepsilon^{10} r_i^5 r^5 \frac{J_{13/2}(\varepsilon r_i)}{(\varepsilon r_i)^{13/2}}\frac{J_{13/2}(\varepsilon r)}{(\varepsilon r)^{13/2}}\ \frac{13}{2^{10}}\{ \qquad \frac{7!5!}{3!^2 2!^2}\Theta_1 + \frac{9!3!}{4!^2 1!^2}\Theta_3 + \frac{11!1!}{5!^2 0!^2}\Theta_5\}+\\
& \quad + \dots]
\end{aligned}
$$

where again Θ_k abbreviates $(\cos k\theta_i \cos k\theta + \sin k\theta_i \sin k\theta)$. Like in the GA case, the terms have been split into two groups, containing the powers $\{\varepsilon^0, \varepsilon^4, \varepsilon^8, \varepsilon^{12}, \dots\}$ and $\{\varepsilon^2, \varepsilon^6, \varepsilon^{10}, \varepsilon^{14}, \dots\}$, respectively. The counterpart to (C.2) becomes in this case

$$
\begin{aligned}
& \frac{J_{3/2}(\varepsilon r)}{(\varepsilon r)^{3/2}} \quad \{1\},\\
& \frac{J_{5/2}(\varepsilon r)}{(\varepsilon r)^{5/2}}\ r\ \{\cos\theta, \quad \sin\theta\},\\
& \frac{J_{7/2}(\varepsilon r)}{(\varepsilon r)^{7/2}}\ r^2\ \{1, \qquad \cos 2\theta, \quad \sin 2\theta\}, \qquad .\\
& \frac{J_{9/2}(\varepsilon r)}{(\varepsilon r)^{9/2}}\ r^3\ \{\cos\theta, \quad \sin\theta, \quad \cos 3\theta, \quad \sin 3\theta\},\\
& \dots
\end{aligned}
\qquad (C.4)
$$

The similarity in form to (4.9) is striking. Although there is no obvious reason to reformulate the half-integer order Bessel functions that appear in (C.3) and (C.4) in terms of standard trigonometric functions, it is entirely possible to do so. For example

$$
\begin{aligned}
F_{1/2}(\xi) &= \frac{J_{1/2}(\xi)}{\xi^{1/2}} &&= \sqrt{\frac{2}{\pi}}\frac{\sin\xi}{\xi}\\
F_{3/2}(\xi) &= \frac{J_{3/2}(\xi)}{\xi^{3/2}} &&= \sqrt{\frac{2}{\pi}}\frac{(\sin\xi - \xi\cos\xi)}{\xi^3}\\
F_{5/2}(\xi) &= \frac{J_{5/2}(\xi)}{\xi^{5/2}} &&= \sqrt{\frac{2}{\pi}}\frac{((3-\xi^2)\sin\xi - 3\xi\cos\xi)}{\xi^2}\\
F_{7/2}(\xi) &= \frac{J_{7/2}(\xi)}{\xi^{7/2}} &&= \sqrt{\frac{2}{\pi}}\frac{((15-6\xi^2)\sin\xi - (15\xi-\xi^3)\cos\xi)}{\xi^7}\\
&\dots
\end{aligned}
$$

The functions $F_k(\xi)$ satisfy a large number of useful identities and recursions, as indicated (in slightly different notation) in Chapter III, §5 of [14]. In cases when d is

even rather than odd, all the Bessel functions entering in the equivalent formulas to (C.3) will be of integer rather than of half-integer orders, and they cannot similarly be replaced by trigonometric functions.

C.2. Other types of radial functions. From the discussions above, it would be quite natural to assume that, corresponding to a general radial function

$$\phi(r) = a_0 + a_1(\varepsilon r)^2 + a_2(\varepsilon r)^4 + a_3(\varepsilon r)^6 + \dots \tag{C.5}$$

there would exist a matching expansion of the form

$$\begin{aligned}\psi(r,\theta,r_i,\theta_i) = [\quad & \\ & \varepsilon^0 r_i^0 r^0 \ f_0(\varepsilon r_i) \ \ f_0(\varepsilon r) \ \ \{c_{0,0}\}+ \\ & \varepsilon^4 r_i^2 r^2 \ f_2(\varepsilon r_i) \ \ f_2(\varepsilon r) \ \ \{c_{1,0} \ + c_{1,1}\Theta_2\}+ \\ & \varepsilon^8 r_i^4 r^4 \ f_4(\varepsilon r_i) \ \ f_4(\varepsilon r) \ \ \{c_{2,0} \ + c_{2,1}\Theta_2 + c_{2,2}\Theta_4\}+ \\ & .... \ + \\ & \varepsilon^2 \ r_i^1 r^1 \ f_1(\varepsilon r_i) \ \ f_1(\varepsilon r) \ \ \{ \qquad d_{0,0}\Theta_1\}+ \\ & \varepsilon^6 \ r_i^3 r^3 \ f_3(\varepsilon r_i) \ \ f_3(\varepsilon r) \ \ \{ \qquad d_{1,0}\Theta_1 + d_{1,1}\Theta_3\}+ \\ & \varepsilon^{10} r_i^5 r^5 \ f_5(\varepsilon r_i) \ \ f_5(\varepsilon r) \ \ \{ \qquad d_{2,0}\Theta_1 + d_{2,1}\Theta_3 + d_{2,2}\Theta_5\}+ \\ & \qquad + \dots] \end{aligned} \tag{C.6}$$

We can assume Taylor expansions with unknown coefficients for the different functions $f_0,\ f_1,\ f_2,\dots$ With the help of a symbolic algebra system such as Mathematica, we can substitute these expansions into (C.6) and then equate coefficients against the expansion for $\psi(r,\theta,r_i,\theta_i)$. This turns out to become free of conflicts only if the coefficients in (C.5) obey a sequence of relations

$$\begin{array}{l} a_1a_2^2 - 2\cdot\frac{3}{2}a_1^2a_3 + \frac{3}{1}a_0a_2a_3 = 0 \\ a_2a_3^2 - 2\cdot\frac{4}{3}a_2^2a_4 + \frac{4}{2}a_1a_3a_4 = 0 \\ a_3a_4^2 - 2\cdot\frac{5}{4}a_3^2a_5 + \frac{5}{3}a_2a_4a_5 = 0 \\ \dots \end{array}$$

Exactly the same sequence of relations arose in [7] and (assuming that it continues indefinitely following the same pattern) was there shown to imply that the radial function $\phi(r)$ has to be exactly of the Bessel type discussed above, which, in the $d \to \infty$ limit, includes GA as a special case, cf. (1.5) in [7].

It remains an open question whether expansions corresponding to (4.8) and (C.3) exist for all standard RBF types (such as MQ, IMQ, IQ). If they do, their forms will need to be somewhat different from that assumed in (C.6).

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] G. E. Andrews, R. Askey, and R. Roy, *Special functions*, vol. 71 of Encyclopedia of Mathematics and its Applications, Cambridge University Press, Cambridge, 1999.

[3] B. J. C. Baxter and S. Hubbert, *Radial basis functions for the sphere*, in Recent progress in multivariate approximation (Witten-Bommerholz, 2000), vol. 137 of Internat. Ser. Numer. Math., Birkhäuser, Basel, 2001, pp. 33–47.

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

[5] N. Flyer, *Exact polynomial reproduction for oscillatory radial basis functions on infinite lattices*, Comput. Math. Appl., 51 (2006), pp. 1199–1208.

[6] B. Fornberg, T. A. Driscoll, G. Wright, and R. Charles, *Observations on the behavior of radial basis function approximations near boundaries*, Comput. Math. Appl., 43 (2002), pp. 473–490.

[7] B. Fornberg, E. Larsson, and G. Wright, *A new class of oscillatory radial basis functions*, Comput. Math. Appl., 51 (2006), pp. 1209–1222.

[8] 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.

[9] ———, *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.

[10] 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.

[11] 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.

[12] B. Fornberg, G. B. Wright, and L. N. Trefethen, *An improved algorithm for stable computations with flat radial basis functions.* in preparation.

[13] B. Fornberg and J. Zuev, *The Runge phenomenon and spatially variable shape parameters in RBF interpolation*, Comput. Math. Appl., 54 (2007), pp. 379–398.

[14] A. Gray and G. B. Mathews, *A treatise on Bessel functions and their appliations to physics*, Second edition prepared by A. Gray and T. M. Mac-Robert, MacMillan and Co. Ltd, London (1922), reprinted Dover Publications Inc., New York, 1966.

[15] 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.

[16] 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.

[17] ———, *Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions*, Comput. Math. Appl., 49 (2005), pp. 103–130.

[18] L. Ling and E. J. Kansa, *Preconditioning for radial basis functions with domain decomposition methods*, Math. Comput. Modelling, 40 (2004), pp. 1413–1427 (2005).

[19] L. Råde and B. Westergren, *Mathematics handbook for science and engineering*, Springer-Verlag, Berlin, fifth ed., 2004. Sections 15.2–15.4 by Michael Patriksson.

[20] R. Schaback, *Multivariate interpolation by polynomials and radial basis functions*, Constr. Approx., 21 (2005), pp. 293–317.

[21] H. C. Thacher, Jr., *Conversion of a power to a series of Chebyshev polynomials*, Commun. ACM, 7 (1964), pp. 181–182.

[22] H. Wendland, *Scattered data approximation*, vol. 17 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2005.

[23] 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.