A stable algorithm for flat radial basis functions on a sphere

Bengt Fornberg, *University of Colorado at Boulder*
Cecile M Piret, *Michigan Technological University*

SIAM J. Sci. Comput.
Vol. 30, No. 1, pp. 60–80

A STABLE ALGORITHM FOR FLAT RADIAL BASIS FUNCTIONS ON A SPHERE[*]

BENGT FORNBERG[†] AND CÉCILE PIRET[†]

Abstract. When radial basis functions (RBFs) are made increasingly flat, the interpolation error typically decreases steadily until some point when Runge-type oscillations either halt or reverse this trend. Because the most obvious method to calculate an RBF interpolant becomes a numerically unstable algorithm for a stable problem in the case of near-flat basis functions, there will typically also be a separate point at which disastrous ill-conditioning enters. We introduce here a new method, RBF-QR, which entirely eliminates such ill-conditioning, and we apply it in the special case when the data points are distributed over the surface of a sphere. This algorithm works even for thousands of node points, and it allows the RBF shape parameter to be optimized without the limitations imposed by stability concerns. Since interpolation in the flat RBF limit on a sphere is found to coincide with spherical harmonics interpolation, new insights are gained as to why the RBF approach (with nonflat basis functions) often is the more accurate of the two methods.

Key words. radial basis functions, RBF, shape parameter, sphere, spherical harmonics

AMS subject classifications. 65D15, 65F22

DOI. 10.1137/060671991

1. Introduction.

Numerical computations in spherical geometries are ubiquitous in many application areas, such as geophysics (including weather and climate modeling), astrophysics, and quantum mechanics. The apparent simplicity of such geometries can be very deceptive. The impossibility to place more than 20 nodes in a completely uniform pattern on a spherical surface severely complicates most high-order numerical methods, which usually rely on highly regular lattice-type node layouts. Although double Fourier methods [12], [29], [30], [39], spherical harmonics methods [2], [18], [40], [42], and spectral element methods [17], [41], [43] all can achieve spectral accuracy (meaning that errors decay faster than algebraically with an increasing number of node points), all of these approaches suffer from different computational limitations, as noted in [7].

Radial basis functions (RBFs), when used as a basis for spectral methods in general geometries or on curved surfaces, feature a striking algebraic simplicity. They have recently been used very successfully by Flyer and Wright for purely convection-type problems on a spherical surface [7], with an implementation for the shallow water equations forthcoming [8]. However, challenges include numerical conditioning and computational speed. The purpose of the present study is to introduce a new computational algorithm, which successfully addresses the first of these two issues. Our presentation of this RBF-QR algorithm does not imply that we always recommend the use of very flat basis functions. It will depend entirely on the application whether the best value of the shape parameter falls inside or outside the range that was already previously available. What the RBF-QR algorithm achieves is that it makes also the flat basis function range fully available for exploration (and, if appropriate, for

[*]Received by the editors October 10, 2006; accepted for publication (in revised form) June 21, 2007; published electronically November 7, 2007.
http://www.siam.org/journals/sisc/30-1/67199.html

[†]University of Colorado, Department of Applied Mathematics, 526 UCB, Boulder, CO 80309 (fornberg@colorado.edu, Cecile.Piret@colorado.edu). The work of the first author was supported by NSF grants DMS-0309803, DMS-0611681, and ATM-0620068. The work of the second author was supported by NSF grants DMS-0309803 and DMS-0611681.

exploitation). For the convection-type test problem just mentioned, this has already been investigated [13].

The essential concept behind the RBF-QR algorithm is that a finite set of near-flat RBFs, although forming a terrible base, nevertheless span an excellent approximation space. The RBF-QR algorithm creates a completely different and very well conditioned base within exactly this same space. Using instead this new base will thus lead to identical results for interpolation, etc., apart from the fact that all ill-conditioning now has been eliminated. In order to carry out this base change in a stable way, the RBFs are first reexpressed as certain truncated infinite sums, after which it transpires that the ill-conditioning can be eliminated *analytically*, before any actual numerics is performed. The latter includes, among other steps, a QR factorization.

This paper starts with a very brief introduction to RBF interpolation, and we then quote some relevant results from the literature, such as the potential significance of the flat basis function limit. The subsequent sections include an introduction to the RBF-QR method, a discussion of computational issues related to it, and numerical test results. The ability to compute stably for all values of the shape parameter leads to novel comparisons between RBFs and spherical harmonics (SPH) interpolations (since the latter are found to arise in the limit of flat RBF). The main observations are summarized in a concluding section.

2. RBF methodology.

In order to explain and to motivate the RBF-QR algorithm, we first give a brief introduction to RBFs and then note how they can be used for solving PDEs.

2.1. The form of an RBF interpolant.

In the case of interpolating data values f_i at scattered distinct node locations $\underline{x}_i$, $i = 1, 2, \ldots n$, in d dimensions, the basic RBF interpolant takes the form

$$s(\underline{x}) = \sum_{i=1}^{n} \lambda_i \, \phi(\|\underline{x} - \underline{x}_i\|), \tag{2.1}$$

where $||\cdot||$ denotes the Euclidean norm. The expansion coefficients λ_i are determined by the interpolation conditions $s(\underline{x}_i) = f_i$; i.e., they can be obtained by solving a linear system $A\,\underline{\lambda} = \underline{f}$. Written out in more detail:

$$\begin{bmatrix} \phi(\|\underline{x}_1 - \underline{x}_1\|) & \phi(\|\underline{x}_1 - \underline{x}_2\|) & \cdots & \phi(\|\underline{x}_1 - \underline{x}_n\|) \\ \phi(\|\underline{x}_2 - \underline{x}_1\|) & \phi(\|\underline{x}_2 - \underline{x}_2\|) & \cdots & \phi(\|\underline{x}_2 - \underline{x}_n\|) \\ \vdots & \vdots & & \vdots \\ \phi(\|\underline{x}_n - \underline{x}_1\|) & \phi(\|\underline{x}_n - \underline{x}_2\|) & \cdots & \phi(\|\underline{x}_n - \underline{x}_n\|) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}. \tag{2.2}$$

In this study, we will limit our attention to the radial functions $\phi(r)$ listed in Table 2.1. The parameter ε is known as the *shape parameter*. As $\varepsilon \to 0$, the basis functions become increasingly flat.

On domains with boundaries, polynomial terms are sometimes added to (2.1), together with some constraints on the coefficients [33]. On a spherical surface, the most natural counterpart is to include some low-order SPH [20]. We will not explore such variations here.

2.2. RBFs for interpolation and for solving PDEs.

For about two decades following the introduction of RBFs by Hardy in 1971 [19], they were mainly used for multivariate data interpolation in a rapidly expanding range of applications. In 1990,

Table 2.1
Definitions of some infinitely differentiable radial functions. The shape parameter ε controls their "flatness."

Name of RBF	Abbreviation	Definition
Multiquadric	MQ	$\sqrt{1+(\varepsilon r)^2}$
Inverse multiquadric	IMQ	$\frac{1}{\sqrt{1+(\varepsilon r)^2}}$
Inverse quadratic	IQ	$\frac{1}{1+(\varepsilon r)^2}$
Gaussian	GA	$e^{-(\varepsilon r)^2}$

Kansa introduced a meshless collocation method to solve PDEs using RBF interpolants [21], [22]. In this method, a smooth RBF interpolant to the scattered data is differentiated analytically in order to approximate partial derivatives. Kansa used this approach to solve parabolic, elliptic, and viscously damped hyperbolic PDEs. This approach is typically spectrally accurate (when boundary conditions are implemented appropriately). Another notable advantage lies in the fact that it does not require any kind of a mesh, as opposed to the case with most other types of PDE solvers, such as finite difference, finite element, and finite volume methods. Creating a suitable mesh over an irregular domain in several dimensions can be highly challenging.

The flat basis function limit $\varepsilon \to 0$ would appear to be severely ill-conditioned, since all of the basis functions then become constant, and thus linearly dependent. The expansion coefficients λ_i will then diverge to plus or minus infinity, causing large numbers of cancellations to arise both when solving (2.2) and when evaluating (2.1). The first indication that the limit nevertheless could be of some interest arose in connection with analysis of interpolants on infinite equispaced lattices, as summarized in [3, Chapter 4]. However, especially after the apparent ill-conditioning was expressed in 1993 as a fundamental "uncertainty principle" [37], the limit was not considered seriously for numerical use for almost a decade. This started to change in 2002 when Driscoll and Fornberg [6] proved that, in this flat basis function limit, a one-dimensional (1-D) RBF interpolant in general reduces to Lagrange's interpolation polynomial. For extensions of this result to more dimensions, see [26], [38]. Already the 1-D result led to the realization that the complete task, going from data to interpolant, is well-conditioned even though the separate steps of going from data to RBF expansion coefficients and then from RBF expansion coefficients to interpolant both can be ill-conditioned. Interpolation with near-flat RBFs has in much of the RBF literature been mistaken as an ill-conditioned problem partly because the most obvious numerical method then is unstable.

As noted further in [15], and used to great advantage for solving elliptic PDEs in [24], the polynomial limit results imply that the RBF approach for PDEs can be viewed as a generalization (to irregular domains and scattered nodes) of the pseudospectral (PS) method [1], [10], [44].

Another apparent contradiction is the following: Why would we ever consider nearly (or totally) flat basis functions when solving convective-type PDEs, for which the solutions might not be smooth at all—maybe even discontinuous? The RBF-QR algorithm shows that, as $\varepsilon \to 0$, the flat basis functions span exactly the same space as do the (distinctly nonflat, but still very smooth) SPH. We can then draw a parallel to the very successful Fourier-PS methods which, for long-time integration, perform excellently even in cases of nonsmooth solutions ([10, section 4.2]).

Closed form expressions for RBF interpolation and differentiation errors in 1-D periodic settings are given in [11]. The errors for smooth data are found to not only decrease exponentially fast with an increasing number of nodes but also decrease rapidly in this $\varepsilon \to 0$ limit (unless potentially adverse effects due to the Runge phenomenon are present, as discussed in [16]). These observations all agree very well both with theoretical analysis [27], [47] and with computational experience in multidimensional settings with irregular node layouts [24].

The contour-Padé method [14], based on contour integration in a complex ε-plane, confirmed that RBF interpolants $s(\underline{x})$ can be computed in a stable way, using standard precision arithmetic, even in the limit of $\varepsilon \to 0$. Although this algorithm formed a very successful tool for discovering and further exploring several key features of RBF approximations [15], [24], [46], it was limited to a relatively low number of data points ($n \lesssim 200$ in 2-D). This algorithm demonstrated explicitly that there is no fundamental barrier against stable computation in the flat basis function limit. It thus confirmed that use of (2.2) followed by (2.1) can be viewed merely as a potentially ill-conditioned approach for computing something that is intrinsically well-conditioned.

Spherical geometries are of particular interest in many geophysical and astrophysical applications. As noted in the introduction, Flyer and Wright [7] were the first to use RBFs to solve purely convective (i.e., nondissipative) PDEs over a spherical surface. Their implementation followed what we here denote by "RBF-Direct," i.e., direct use of (2.2) followed by (2.1).

The comments above set the context which motivates the present work. We introduce here a new computational algorithm RBF-QR, which, for RBF computations on a sphere, eliminates the ill-conditioning of RBF-Direct for small values of ε (at least for up to several thousands of points). However, it will still depend on the application whether the low ε regime, now made computationally available, is advantageous or not.

3. The RBF-QR method.

Compared to the contour-Padé method, the RBF-QR method is faster and algorithmically simpler and it can be used for much larger numbers of points. Although we introduce it here only for the special case of nodes located on the surface of a sphere, it is being developed also for general domains in a parallel research effort [25]. In this present case with nodes on a sphere, we measure all distances that appear in (2.1) and in (2.2) as is customary between points in a 3-D space and not geodesically along great circle arcs.

3.1. The concept of an equivalent basis.

The key idea behind the RBF-QR method is to replace, in the case of small ε, the extremely ill-conditioned RBF basis with a well-conditioned one that spans exactly the same space. It turns out to be possible to do this in a way that does not at any stage involve numerical cancellations. The concept of the base change is somewhat reminiscent of how $\{1, x, x^2, \dots, x^n\}$ forms a very ill-conditioned basis over $[-1, 1]$, whereas the Chebyshev basis $\{T_0(x), T_1(x), T_2(x), \dots, T_n(x)\}$ is much better conditioned. Since the spaces spanned by the two bases are identical, the results of interpolation using the two bases will also be identical, except for the fact that computations with the latter are vastly more stable with respect to the influence of truncation and rounding errors.

In the present case of RBF-QR applied on the surface of a sphere, the new equivalent bases that we introduce will be seen to converge to the spherical harmonics basis as $\varepsilon \to 0$. We therefore next give a brief introduction to spherical harmonics.

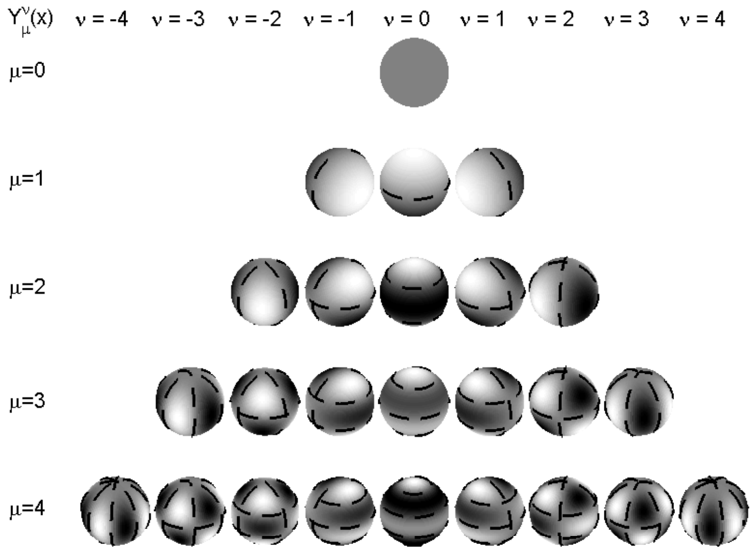

FIG. 3.1. *Spherical harmonics basis functions of the first five orders (cf. the functional forms for the first three orders, given in Table* 3.1*). The shades of gray reflect numerical values; dashed lines mark zero contours.*

3.2. SPH. These functions are the counterparts on the surface of the unit sphere S^2 (defined by $x^2+y^2+z^2=1$) to Fourier modes around the periphery of the unit circle S^1 (defined by $x^2+y^2=1$). Although both of these function sets form orthonormal bases, they differ significantly when used numerically, especially when it is needed to switch between spectral coefficients and node values. A Fourier expansion with n coefficients corresponds naturally to node values at n equispaced points. In numerical SPH calculations, it is most common to use in physical space latitude-longitude–type node sets involving about three times as many nodes as there are SPH coefficients and then rely on least squares when transferring data from node values to coefficients. Although no direct counterpart to the FFT algorithm is available, several relatively fast algorithms for large numbers of modes have been proposed, e.g., [5], [31], [32], [34].

Closed form expressions for the SPH basis functions tend to be fairly complicated. The definition that we adhere to here agrees for $(x,y,z)\in S^2$ with

(3.1)
$$Y_\mu^{\ \nu}(x,y,z)=\begin{cases}\sqrt{\frac{2\mu+1}{4\pi}}\sqrt{\frac{(\mu-\nu)!}{(\mu+\nu)!}}P_\mu^\nu(z)\cos(\nu\tan^{-1}(\frac{y}{x})), & \nu=0,1,\ldots,\mu,\\ \sqrt{\frac{2\mu+1}{4\pi}}\sqrt{\frac{(\mu+\nu)!}{(\mu-\nu)!}}P_\mu^{-\nu}(z)\sin(-\nu\tan^{-1}(\frac{y}{x})), & \nu=-\mu,\ldots,-1.\end{cases}$$

Here $P_\mu^\nu(z)$ are the associated Legendre functions. The functions $Y_\mu^\nu(\underline{x})$ corresponding to $\mu=0,1,\ldots,4$ are illustrated in Figure 3.1.

As indicated in Table 3.1, the SPH can alternatively be viewed as simple polynomials restricted to $(x,y,z)\in S^2$. For each value of μ, the μ^2 SPH of that and lower orders span the space of all independent polynomials in (x,y,z) of degree μ (after the dependence $x^2+y^2+z^2=1$ has been accounted for).

TABLE 3.1

SPH basis functions of the first few orders, expressed as low degree polynomials in x, y, z, which are then evaluated over the unit sphere $(x, y, z) \in S^2$, i.e., $x^2 + y^2 + z^2 = 1$.

$Y_\mu{}^\nu(\underline{x})$	$\nu = -2$	$\nu = -1$	$\nu = 0$	$\nu = 1$	$\nu = 2$
$\mu = 0$			$\frac{1}{2\sqrt{\pi}}$		
$\mu = 1$		$-\frac{1}{2}\sqrt{\frac{3}{2\pi}}y$	$\frac{1}{2}\sqrt{\frac{3}{\pi}}z$	$-\frac{1}{2}\sqrt{\frac{3}{2\pi}}x$	
$\mu = 2$	$\frac{1}{2}\sqrt{\frac{15}{2\pi}}xy$	$-\frac{1}{2}\sqrt{\frac{15}{2\pi}}zy$	$\frac{1}{4}\sqrt{\frac{5}{\pi}}(3z^2-1)$	$-\frac{1}{2}\sqrt{\frac{15}{2\pi}}zx$	$\frac{1}{4}\sqrt{\frac{15}{2\pi}}(x^2-y^2)$

TABLE 3.2

SPH expansion coefficients corresponding to different choices of smooth RBFs.

Radial function	Expansion coefficients $c_{\mu,\varepsilon}$
MQ	$\frac{-2\pi(2\varepsilon^2+1+(\mu+1/2)\sqrt{1+4\varepsilon^2})}{(\mu+3/2)(\mu+1/2)(\mu-1/2)}\left(\frac{2}{1+\sqrt{4\varepsilon^2+1}}\right)^{2\mu+1}$
IMQ	$\frac{4\pi}{(\mu+1/2)}\left(\frac{2}{1+\sqrt{4\varepsilon^2+1}}\right)^{2\mu+1}$
IQ	$\frac{4\,\pi^{3/2}\mu!}{\Gamma(\mu+\frac{3}{2})(1+4\varepsilon^2)^{\mu+1}}\ {}_2F_1(\mu+1,\mu+1;2\mu+2;\frac{4\varepsilon^2}{1+4\varepsilon^2})$
GA	$\frac{4\pi^{3/2}}{\varepsilon^{2\mu+1}}e^{-2\varepsilon^2}I_{\mu+1/2}(2\varepsilon^2)$

A SPH expansion of a function defined over the unit sphere takes the form

$$s(x, y, z) = \sum_{\mu=0}^{\infty}\sum_{\nu=-\mu}^{\mu} c_{\mu,\nu}\, Y_\mu{}^\nu(x, y, z). \tag{3.2}$$

Truncated SPH expansions ($\mu \le \mu_{\max}$) feature a completely uniform resolution over the surface of the sphere. As was noted in the introduction, truncated SPH expansions provide one of the main approaches for reaching spectral accuracy when numerically solving PDEs on a sphere [2], [18], [40], [42]; see especially [7] for a comparison between this and other methodologies (including RBFs).

3.3. Expansion formulas for RBFs in terms of SPH. We next quote some formulas that can be used to transform a basis made up of RBFs to one based on SPH. Hubbert and Baxter [20] give expressions for the coefficients $c_{\mu,\varepsilon}$ in expansions of the form

$$\phi(\|\underline{x} - \underline{x}_i\|) = \sum_{\mu=0}^{\infty}{\sum_{\nu=-\mu}^{\mu}}{}' \{c_{\mu,\varepsilon}\, \varepsilon^{2\mu}\, Y_\mu{}^\nu(\underline{x}_i)\}\, Y_\mu{}^\nu(\underline{x}), \tag{3.3}$$

where the symbol $\sum'$ implies halving the $\nu = 0$ term of the sum. The results for the radial functions in Table 2.1 are shown in Table 3.2 (including IQ, not given in [20]). A key feature of these formulas is that, even for vanishingly small ε, all coefficients can be calculated without any loss of significant digits caused by numerical cancellations. Below are some notes on these expansions:

- In the formula for IQ, ${}_2F_1(\dots)$ denotes the (Gauss) hypergeometric function.
- In the formula for GA, $I_{\mu+1/2}$ denotes a Bessel function of the second kind. It follows from the identity $\frac{I_{\mu+1/2}(2\varepsilon^2)}{\varepsilon^{2\mu+1}} = \frac{1}{\Gamma(\mu+1)\sqrt{\pi}}\int_{-1}^{1} e^{2\varepsilon^2 t}(1-t^2)^k dt$ that the apparent singularity of $c_{\mu,\varepsilon}$ at $\varepsilon = 0$ is a removable one.

TABLE 3.3
Expansion coefficients for two cases of piecewise smooth radial functions.

Radial function	Definition	Expansion coefficients c_μ
Cubic	$\lvert r\rvert^3$	$\frac{36\pi}{(\mu+\frac{5}{2})(\mu+\frac{3}{2})(\mu+\frac{1}{2})(\mu-\frac{1}{2})(\mu-\frac{3}{2})}$
Thin plate splines (TPS)	$r^2\log\lvert r\rvert$	$\frac{16\pi}{(\mu+2)(\mu+1)\mu(\mu-1)}$

- In practice, we truncate the infinite outer sum in (3.3) after a finite number of terms. This process is explained in more detail in section 3.5.2.
- The shape parameter ε appears in (3.3) both in the factors $\varepsilon^{2\mu}$ and also inside the expansion coefficients $c_{\mu,\varepsilon}$. Because the matrix algebra in the RBF-QR algorithm requires numerical values of $c_{\mu,\varepsilon}$, we need to give a numerical value to ε at the beginning of our algorithm. However, to eliminate any danger of numerical underflow, we wait until the very end to introduce the $\varepsilon^{2\mu}$ factors seen in (3.3) (at which point they can be factored out and discarded).
- Expansions are possible also for piecewise smooth RBFs. The expansions then take the form

$$\phi(\|\underline{x}-\underline{x}_i\|)=\sum_{\mu=0}^{\infty}{\sum_{\nu=-\mu}^{\mu}}'\{c_\mu\, Y_\mu{}^\nu(\underline{x}_i)\}\; Y_\mu{}^\nu(\underline{x}),$$

  with some examples of expansion coefficients given in Table 3.3. Since such RBFs do not give spectral accuracy, and also have no ε dependence (and therefore no flat limit), these cases are of less interest in the present context.

3.4. Matrix representation and QR factorization.

3.4.1. Change of basis. Following (3.3), we rewrite the original ill-conditioned basis as expansions in terms of successive SPH as

(3.4)
$$\begin{cases}
\phi(\|\underline{x}-\underline{x}_1\|)= & \frac{c_{0,\varepsilon}}{2}Y_0^0(\underline{x}_1)Y_0^0(\underline{x}) \\
 & +\varepsilon^2 c_{1,\varepsilon}\{Y_1^{-1}(\underline{x}_1)Y_1^{-1}(\underline{x})+\frac{1}{2}Y_1^0(\underline{x}_1)Y_1^0(\underline{x})+Y_1^1(\underline{x}_1)Y_1^1(\underline{x})\} \\
 & +\varepsilon^4 c_{2,\varepsilon}\{......\}+\varepsilon^6 c_{3,\varepsilon}\{.........\}+\varepsilon^8 c_{4,\varepsilon}\{............\}+\cdots, \\
\phi(\|\underline{x}-\underline{x}_2\|)= & \frac{c_{0,\varepsilon}}{2}Y_0^0(\underline{x}_2)Y_0^0(\underline{x}) \\
 & +\varepsilon^2 c_{1,\varepsilon}\{Y_1^{-1}(\underline{x}_2)Y_1^{-1}(\underline{x})+\frac{1}{2}Y_1^0(\underline{x}_2)Y_1^0(\underline{x})+Y_1^1(\underline{x}_2)Y_1^1(\underline{x})\} \\
 & +\varepsilon^4 c_{2,\varepsilon}\{......\}+\varepsilon^6 c_{3,\varepsilon}\{.........\}+\varepsilon^8 c_{4,\varepsilon}\{............\}+\cdots, \\
\vdots & \vdots \\
\phi(\|\underline{x}-\underline{x}_n\|)= & \frac{c_{0,\varepsilon}}{2}Y_0^0(\underline{x}_n)Y_0^0(\underline{x}) \\
 & +\varepsilon^2 c_{1,\varepsilon}\{Y_1^{-1}(\underline{x}_n)Y_1^{-1}(\underline{x})+\frac{1}{2}Y_1^0(\underline{x}_n)Y_1^0(\underline{x})+Y_1^1(\underline{x}_n)Y_1^1(\underline{x})\} \\
 & +\varepsilon^4 c_{2,\varepsilon}\{......\}+\varepsilon^6 c_{3,\varepsilon}\{.........\}+\varepsilon^8 c_{4,\varepsilon}\{............\}+\cdots.
\end{cases}$$

This can be rewritten in matrix×vector form as follows:

$$\begin{bmatrix} \phi(\|\underline{x}-\underline{x}_1\|) \\ \phi(\|\underline{x}-\underline{x}_2\|) \\ \vdots \\ \phi(\|\underline{x}-\underline{x}_n\|) \end{bmatrix}$$

(3.5)

$$= \begin{bmatrix} \frac{c_{0,\varepsilon}}{2}Y_0^0(\underline{x}_1) & \frac{\varepsilon^2 c_{1,\varepsilon}}{1}Y_1^{-1}(\underline{x}_1) & \frac{\varepsilon^2 c_{1,\varepsilon}}{2}Y_1^0(\underline{x}_1) & \frac{\varepsilon^2 c_{1,\varepsilon}}{1}Y_1^1(\underline{x}_1) & \cdots \\ \frac{c_{0,\varepsilon}}{2}Y_0^0(\underline{x}_2) & \frac{\varepsilon^2 c_{1,\varepsilon}}{1}Y_1^{-1}(\underline{x}_2) & \frac{\varepsilon^2 c_{1,\varepsilon}}{2}Y_1^0(\underline{x}_2) & \frac{\varepsilon^2 c_{1,\varepsilon}}{1}Y_1^1(\underline{x}_2) & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{c_{0,\varepsilon}}{2}Y_0^0(\underline{x}_n) & \frac{\varepsilon^2 c_{1,\varepsilon}}{1}Y_1^{-1}(\underline{x}_n) & \frac{\varepsilon^2 c_{1,\varepsilon}}{2}Y_1^0(\underline{x}_n) & \frac{\varepsilon^2 c_{1,\varepsilon}}{1}Y_1^1(\underline{x}_n) & \cdots \end{bmatrix} \begin{bmatrix} Y_0^0(\underline{x}) \\ Y_1^{-1}(\underline{x}) \\ Y_1^0(\underline{x}) \\ Y_1^1(\underline{x}) \\ Y_2^{-2}(\underline{x}) \\ Y_2^{-1}(\underline{x}) \\ Y_2^0(\underline{x}) \\ Y_2^1(\underline{x}) \\ Y_2^2(\underline{x}) \\ \vdots \end{bmatrix}$$

$$= B \cdot Y(\underline{x}).$$

The key observation in what follows is that, if we multiply both sides of (3.5) with any nonsingular matrix from the left, the effect will be that we have formed new linear combinations of existing basis functions; i.e., the space that the functions span has not changed.

A QR factorization of B creates in the upper triangular matrix new linear combinations of the rows of B. In this process, elements in different columns are never combined with each other. Powers of ε will appear in the same pattern in the resulting upper triangular matrix as they did in the B-matrix, and no mixing of large and small elements will occur, no matter the value of ε. We thus factor B into a product $B = Q \cdot E \cdot R$, where Q is unitary, E is diagonal, and R is upper triangular. From the observations above, $B \cdot Y(\underline{x})$ and $R \cdot Y(\underline{x})$ will then span the same space. We will do this factorization in such a way that the ill-conditioning issue becomes entirely confined to the E-matrix and thus has disappeared from the numerical problem when using $R \cdot Y(\underline{x})$ in place of the original basis $B \cdot Y(\underline{x})$. The essential point that makes the RBF-QR algorithm work is that the ill-conditioning of the original base given in the left-hand side of (3.5) and (3.6) has become entirely confined to the E-matrix. This matrix both enters and disappears from the calculation *analytically*; i.e., it never enters into the numerical calculation of the $R \cdot Y(\underline{x})$.

Written in equation form:

$$\begin{bmatrix} \phi(\|\underline{x}-\underline{x}_1\|) \\ \phi(\|\underline{x}-\underline{x}_2\|) \\ \phi(\|\underline{x}-\underline{x}_3\|) \\ \phi(\|\underline{x}-\underline{x}_4\|) \\ \vdots \\ \phi(\|\underline{x}-\underline{x}_n\|) \end{bmatrix} = \begin{bmatrix} & & \\ & Q & \\ & & \end{bmatrix} \begin{bmatrix} 1 & & & & & \\ & \varepsilon^2 & & & & \\ & & \varepsilon^2 & & & \\ & & & \varepsilon^2 & & \\ & & & & \varepsilon^4 & \\ & & & & & \ddots \end{bmatrix}$$

$$
(3.6) \qquad \times \left[\begin{array}{ccccccccccc}
* & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\
 & * & * & * & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\
 & & * & * & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\
 & & & * & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\
 & & & & * & * & * & * & * & \cdot & \cdots \\
 & & & & & * & * & * & * & \cdot & \cdots \\
 & & & & & & \ddots & \cdots & \cdots & \cdot & \cdots
\end{array}\right]
\left[\begin{array}{l}
Y_0^0(\underline{x}) \\
Y_1^{-1}(\underline{x}) \\
Y_1^0(\underline{x}) \\
Y_1^1(\underline{x}) \\
Y_2^{-2}(\underline{x}) \\
Y_2^{-1}(\underline{x}) \\
Y_2^0(\underline{x}) \\
Y_2^1(\underline{x}) \\
Y_2^2(\underline{x}) \\
\vdots
\end{array}\right]
$$

$$= (Q \cdot E \cdot R) \cdot Y(\underline{x}),$$

where Q is a unitary $n \times n$ matrix, E is a $n \times n$ diagonal matrix, and R is an upper triangular $n \times m$ matrix (where the value for m will be discussed shortly in section 3.5.2). The entries marked as "$*$" in the matrix R are of size ε^0. These appear only in upper triangular square blocks along the main diagonal, of sizes 1×1, 3×3, 5×5, etc. All of the other nonzero entries of R, marked as "$\cdot$", contain a higher-order leading power of the form ε^{2k}, $k = 1, 2, \ldots$; i.e., they vanish in significance when $\varepsilon \to 0$. As noted already, the entries in the matrix$\times$vector product $R \cdot Y(\underline{x})$ form a basis which spans exactly the same space as the original (as $\varepsilon \to 0$, extremely ill-conditioned) RBF basis.

Another way to arrive at the same $R \cdot Y(\underline{x})$ representation is described next. Noting the structure of B from (3.5), we can factor it

$$
B = \left[\begin{array}{ccc}
Y_0^0(\underline{x}_1) & Y_1^{-1}(\underline{x}_1) & \cdots \\
Y_0^0(\underline{x}_2) & Y_1^{-1}(\underline{x}_2) & \cdots \\
\cdots & \cdots & \cdots
\end{array}\right]
\left[\begin{array}{ccc}
\varepsilon^0 & & \\
 & \varepsilon^2 & \\
 & & \ddots
\end{array}\right]
\left[\begin{array}{ccc}
\frac{c_{0,\varepsilon}}{2} & & \\
 & c_{1,\varepsilon} & \\
 & & \ddots
\end{array}\right].
$$

After QR decomposing the first factor

$$
\left[\begin{array}{ccc}
Y_0^0(\underline{x}_1) & Y_1^{-1}(\underline{x}_1) & \cdots \\
Y_0^0(\underline{x}_2) & Y_1^{-1}(\underline{x}_2) & \cdots \\
\cdots & \cdots & \cdots
\end{array}\right]
= \left[\begin{array}{c} \\ Q \\ \\ \end{array}\right]
\left[\begin{array}{ccc}
r_{11} & r_{12} & \cdots \\
 & r_{22} & \cdots \\
 & & \ddots
\end{array}\right],
$$

we have

$$
\left[\begin{array}{c} \\ B \\ \\ \end{array}\right]
= \left[\begin{array}{c} \\ Q \\ \\ \end{array}\right]
\left[\begin{array}{ccc}
r_{11} & r_{12} & \cdots \\
 & r_{22} & \cdots \\
 & & \ddots
\end{array}\right]
\left[\begin{array}{ccc}
\varepsilon^0 & & \\
 & \varepsilon^2 & \\
 & & \ddots
\end{array}\right]
\left[\begin{array}{ccc}
\frac{c_{0,\varepsilon}}{2} & & \\
 & c_{1,\varepsilon} & \\
 & & \ddots
\end{array}\right].
$$

Transferring the diagonal matrix with powers of ε from the right-hand side to the left-hand side of the upper triangular matrix gives exactly the same result as shown in (3.6). An advantage of this second description (followed in the code in the appendix) is that it more clearly conveys that the QR decomposition can be carried out in a way that is completely independent of the choice of RBF (and of the value of ε).

In the computational algorithm, we make a minor additional base modification. The matrix R can be represented as $[R_1|R_2]$, where R_1 is a square upper triangular matrix. Therefore, assuming that the diagonal entries in R are nonzero, we can further

factor $R = R_1[\, I\ |(R_1)^{-1}R_2]$. This produces the new basis $[\, I\ |(R_1)^{-1}R_2] \cdot Y$, which we will be using:
(3.7)

$$[\, I\ |(R_1)^{-1}R_2]{\cdot}Y = \begin{bmatrix} 1 &&&&&&&&&&& \cdot & \cdots \\ & 1 &&&&&&&&&& \cdot & \cdots \\ && 1 &&&&&&&&& \cdot & \cdots \\ &&& 1 &&&&&&&& \cdot & \cdots \\ &&&& 1 &&&&&&& \cdot & \cdots \\ &&&&& 1 &&&&&& \cdot & \cdots \\ &&&&&& 1 &&&&& \cdot & \cdots \\ &&&&&&& 1 &&&& \cdot & \cdots \\ &&&&&&&& 1 &&& \cdot & \cdots \\ &&&&&&&&& 1 && \cdot & \cdots \\ &&&&&&&&&& \ddots & \cdot & \cdots \end{bmatrix} \begin{bmatrix} Y_0^0(\underline{x}) \\ Y_1^{-1}(\underline{x}) \\ Y_1^0(\underline{x}) \\ Y_1^1(\underline{x}) \\ Y_2^{-2}(\underline{x}) \\ Y_2^{-1}(\underline{x}) \\ Y_2^0(\underline{x}) \\ Y_2^1(\underline{x}) \\ Y_2^2(\underline{x}) \\ \vdots \\ \vdots \end{bmatrix}.$$

Each new basis function is now a SPH, with a perturbation. These perturbations fade away as $\varepsilon \to 0$ because all of the entries denoted with "$\cdot$" are of size $O(\varepsilon^2)$ or smaller. With the previous assumption that the diagonal entries of R are nonzero, this shows that, as $\varepsilon \to 0$, the terms of the new basis converge to the successive SPH. If the number of nodes (i.e., the number of rows and columns in the collocation matrix A) is a perfect square $n = \mu_0^2$, then the RBF interpolants converge to a unique SPH expansion (3.2) with $\mu < \mu_0$. The computational procedure of using the new (but mathematically equivalent) basis remains stable as $\varepsilon \to 0$ on the further assumption that the nodes are distributed in such a way that SPH interpolation is nonsingular. Although this is very likely in any practical case, it follows from a simple argument that no node-independent basis functions can exist in more than 1-D such that nonsingularity is assured for all distinct node locations [28].

3.5. Computational considerations. In this section, we discuss the two issues of code complexity and the truncation strategy for the infinite expansions in (3.7).

3.5.1. Code complexity. A Matlab code for the RBF-QR algorithm is presented in the appendix. The cost is dominated by the QR factorization, and it will therefore have an $O(n^3)$ operation count, just as the RBF-Direct method (for which the work is dominated by one matrix inversion). Figure 3.2 displays the computational times of both methods versus the number of nodes n. The trends appear as slightly more favorable than $O(n^3)$, since Matlab's data handling becomes more efficient as matrices become larger. The figure also shows that the computational time diminishes with ε. This is due to the fact that a smaller ε allows for earlier truncation in the expansions (3.4) and fewer columns need to be retained, at first in R and then in $R_1^{-1}R_2$. This truncation procedure is described in more detail below.

3.5.2. Truncation. The matrix B in (3.5) features from the left 1 column with elements of size $O(\varepsilon^0)$, 3 columns of size $O(\varepsilon^2)$, 5 columns of size $O(\varepsilon^4)$, etc., corresponding to $\mu = 0, 1, 2, \dots$, respectively, in (3.3). With n data points, the matrix B will have n rows. We construct this matrix B as $\{B_0, B_1, B_2, \dots\}$ where the matrix block B_μ is of size $n \times (2\mu + 1)$. To ensure that all of the new basis functions are obtained to machine precision (16 significant digits), we include enough blocks that the max norm of the last included one is less than 10^{-16} of the max norm of the block that contains the nth column of B.

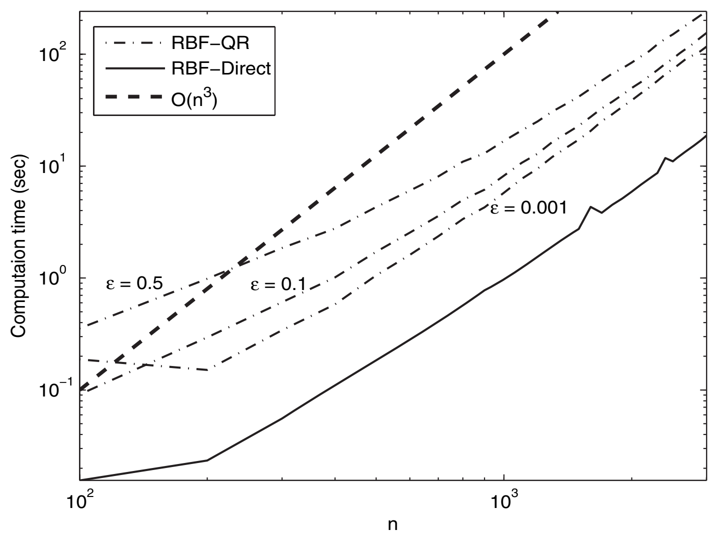

Fig. 3.2. *Plot of the computational time for the RBF-QR method versus the number of nodes for* $\varepsilon = 0.5, 0.1$, *and* 0.001. *These are compared against the times for RBF-Direct and with a line showing the slope corresponding to* $O(n^3)$. *The times are given for a Matlab implementation running under Windows on a* 1.86 *GHz PC.*

4. Numerical tests for interpolation on a sphere. We initially consider two different node distributions, both containing $n = 1849$ nodes: (a) near-uniform distribution, obtained as the solution to a minimum energy problem—as would arise from the equilibrium of freely moving and mutually repelling equal electric charges [45] and (b) uniformly random distribution, as generated, for example, by the Matlab statements

```
n = 1849
z = 2*rand(1,n)-1;
r = sqrt(1-z.^2);
theta = 2*pi*rand(1,n);
x = r.*cos(theta);
y = r.*sin(theta);
```

(the number $n = 1849 = 43^2$ gives the same number of nodes as there are coefficients in a SPH expansion that is truncated to $\mu \leq 42$, as commonly used in SPH tests, and then denoted "T42" [41]). The two types of node distributions are shown in Figure 4.1. We consider the following two test functions:

$$\begin{array}{lll} \text{Gaussian bell:} & g(x,y,z) & = e^{-(\frac{2.25}{R}\arccos x)^2}, \\ \text{Cosine bell:} & c(x,y,z) & = \begin{cases} \frac{1}{2}(1+\cos(\frac{\pi}{R}\arccos x)) & x > \cos R, \\ 0 & x \leq \cos R, \end{cases} \end{array} \tag{4.1}$$

of smoothness C_∞ and C_1, respectively. R is here a parameter which controls how peaked the bells are, going from spikelike at $R = 0$ to flat for increasing R. The cosine bell features a jump in the second derivative at the edge of its region of support. An equivalent way to describe the two bells is to replace $\arccos x$ by ω, where ω is the angle, as seen from the center of the sphere, between a point on the sphere and the

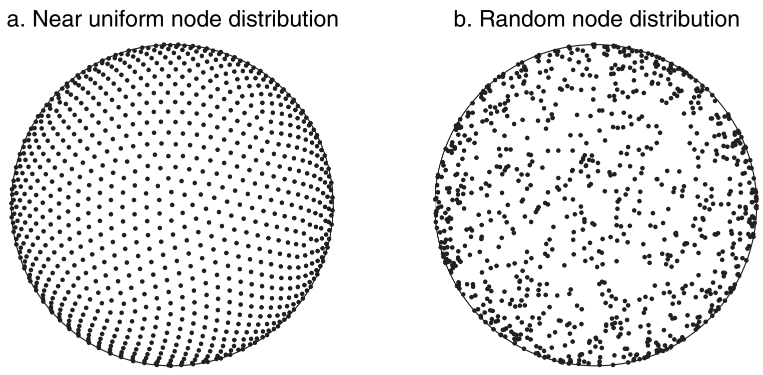

FIG. 4.1. *Different point distributions for $n = 1849$ node points on the unit sphere S^2.*

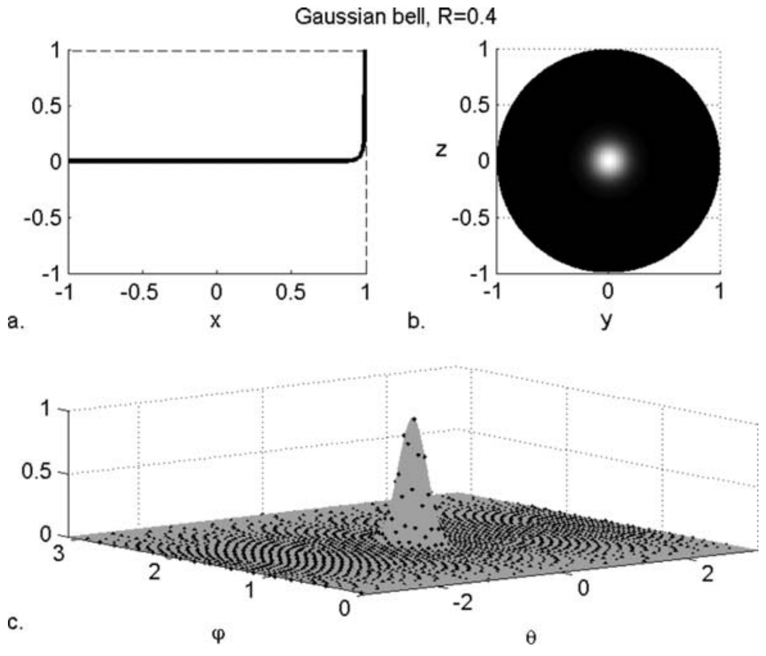

FIG. 4.2. *Three illustrations of the Gaussian bell* (a) *as a function of x, according to* (4.1), (b) *gray scale on a sphere surface, viewed from the positive x-direction, and* (c) *unrolled on a spherical coordinate φ, θ-plane (with the $n = 1849$ near-uniform node locations also marked).*

center of the respective bell. The support of the cosine bell is then given by $\omega < R$. Figures 4.2 and 4.3 show each of these two test functions in three different ways. In parts (c) of these figures, as well as in the rest of this paper, we adhere to the standard definition of spherical coordinates

$$\begin{cases} x &= \rho \sin\varphi \cos\theta, \\ y &= \rho \sin\varphi \sin\theta, \\ z &= \rho \cos\varphi \end{cases}$$

and restrict this to $\rho = 1$ for the unit sphere. The angle φ is the colatitude and is measured from the z-axis.

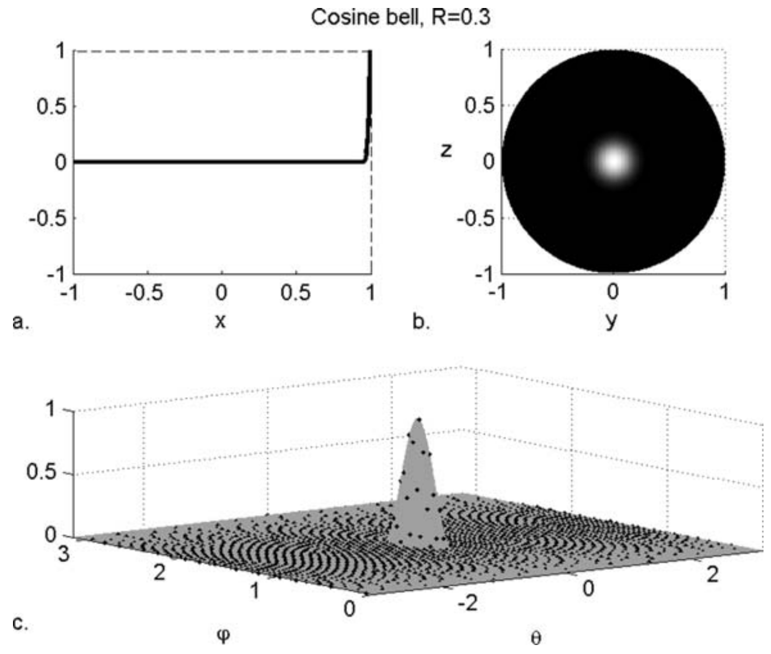

FIG. 4.3. *Cosine bell, displayed in the same manner as the Gaussian bell in Figure* 4.2.

The test functions are sampled over the two node sets, and the max norm errors of the MQ RBF interpolants are then evaluated (by dense sampling over the sphere) for different values of ε, using both RBF-Direct (based on (2.2) and (2.1)) and the new RBF-QR method.

4.1. Results for the Gaussian bell. Figure 4.4 shows the interpolation errors as functions of ε. Near-uniform node distributions are seen to give 2–3 orders of magnitude higher accuracy than random node distributions. Even with ill-conditioning issues eliminated, it is still detrimental to the overall accuracy that some small areas, purely by chance, have become much less resolved than others. We will thus not consider the random node case any further in this study.

When ε is decreased, RBF-Direct fails around $\varepsilon = 1$, whereas the RBF-QR method can be used for the remaining interval $0 \le \varepsilon \le 1$. The rapid improvement in accuracy as ε is lowered from 10^2 to 10^0 is similar to what is described analytically (in a simplified setting) in [11]. This improvement trend ceases around $\varepsilon = 1$. In the case of the wide bell ($R = 0.6$), this is due to the limited precision available in 64-bit floating point. In the case of the narrower bell ($R = 0.4$), the machine rounding level is not reached. The errors increase slightly as ε approaches the SPH case of $\varepsilon = 0$. Figure 4.5 displays in more detail how the interpolation error over the sphere varies with both ε and R. We have here run RBF-Direct to as low ε-values as possible before it breaks down due to ill-conditioning and used RBF-QR for the remaining ε-range. The large flat region for large R and small ε is a direct consequence of the 10^{-16} precision of standard floating point. The lower ε-limit for RBF-Direct is imposed by ill-conditioning. There is no equally sharp upper limit for RBF-QR, but the convergence in (3.4) degrades severely when ε increases above one. In the present case of $n = 1849$, both methods work well in a narrow overlap region for ε slightly larger than one. For lower values of n, the overlap becomes wider, whereas it may vanish for

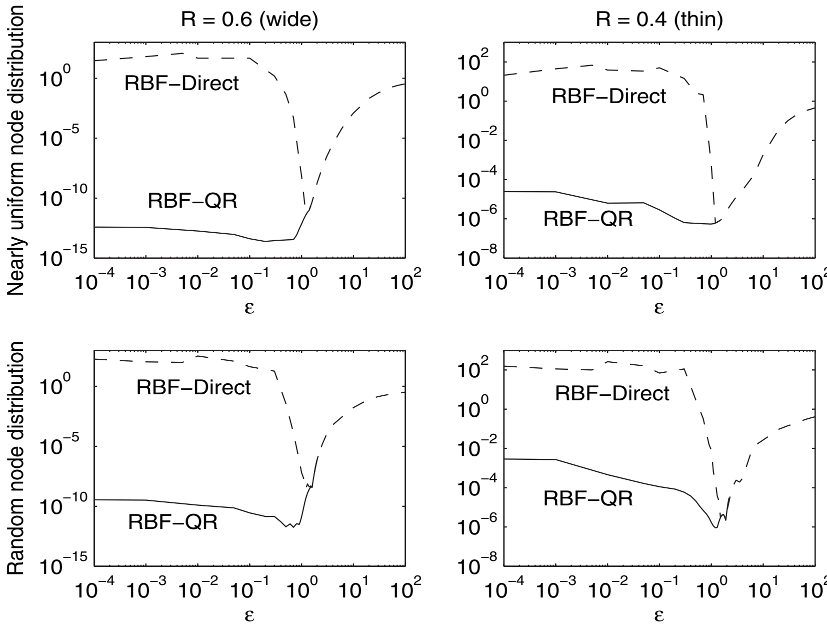

FIG. 4.4. *Log-log plots of the max norm error vs. values of ε for the Gaussian bells of two different widths. The subplots in the top row show the results with nearly uniform nodes and the bottom row with random nodes. In both cases, the number of nodes was $n = 1849$. Note that the vertical scales are different between the $R = 0.6$ and $R = 0.4$ plots.*

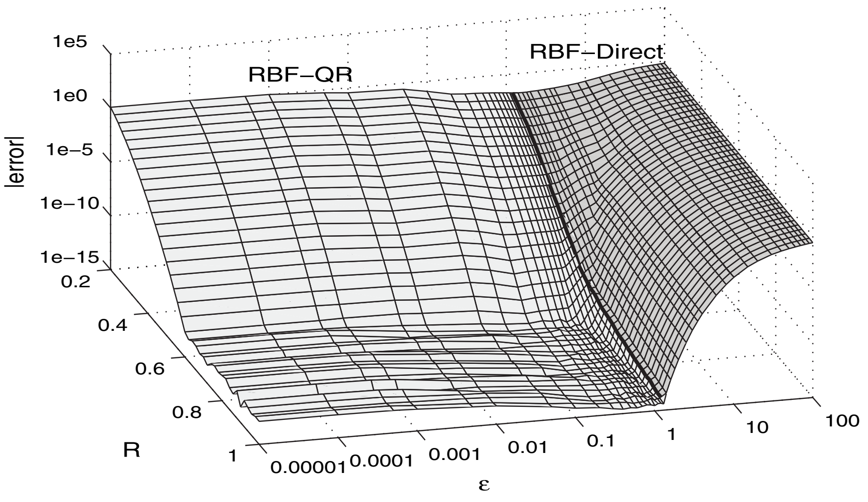

FIG. 4.5. *Gaussian bell interpolation error for different values of the bell width R and MQ shape parameter ε, in the case of $n = 1849$ near-uniform nodes. The dark line at an ε-value slightly larger than one marks where we changed the algorithm in the calculation.*

higher values of n (leaving some gap in the ε-range in which neither of the methods will be practical unless the arithmetic precision is increased beyond standard double precision).

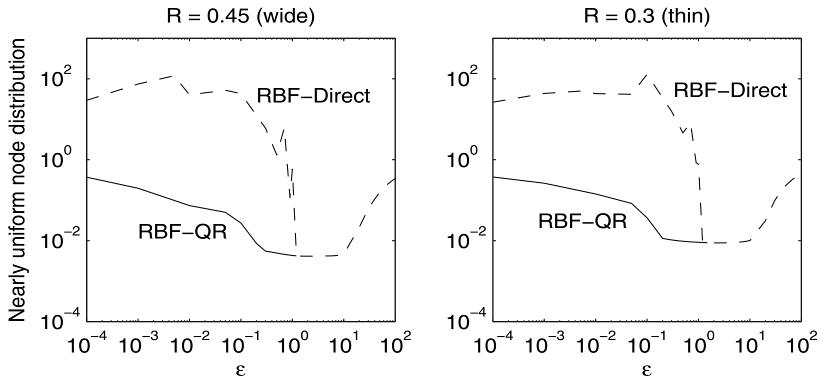

FIG. 4.6. *Log-log plot of the max norm errors in the cosine bell test case, using MQ, with* $n = 1849$ *nodes.*

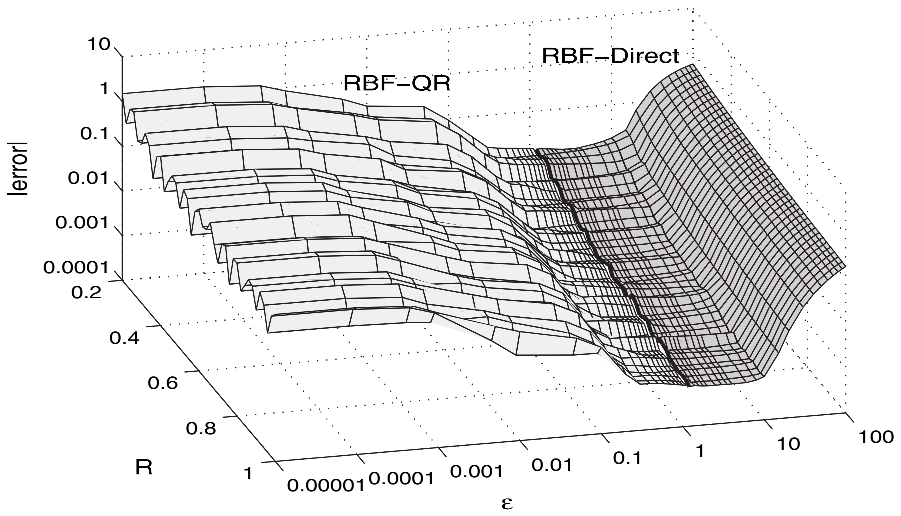

FIG. 4.7. *Cosine bell interpolation error for different values of the bell width R and MQ shape parameter* ε*, in the case of* $n = 1849$ *near-uniform nodes.*

4.2. Results for the cosine bell. Figure 4.6 shows that the lack of smoothness of the cosine bell (featuring a discontinuous second derivative around its edge) somehow causes much larger errors than in the Gaussian bell case, with especially large errors arising as $\varepsilon \to 0$ (the SPH case). Figure 4.7 displays, in the same style as used earlier in the Gaussian bell case, the interpolation error as a function of ε and R. In the Gaussian bell case, errors decrease very rapidly with increasing R (note the different vertical scales in the two columns of subplots in Figure 4.4). The cosine bell case is fundamentally different in that errors drop only weakly with increasing R and also grow significantly as $\varepsilon \to 0$. A more detailed discussion of this seemingly less favorable situation (and two remedies that greatly improve the accuracy at small ε) can be found in [13].

5. Some comments on the choice of "optimal" ε. By using RBF-Direct for large ε and RBF-QR for small ε, we have the capability to compute RBF interpolants (and also to solve PDEs) over a sphere for all values of ε (at least for up to a few thousands of nodes when using standard double precision). This offers new opportunities for exploring issues such as determining an “optimal” ε and assessing whether truncated SPH expansions (i.e., RBF in the $\varepsilon \to 0$ limit) provide a “best possible” representation of functions on a sphere.

As Figures 4.4 and 4.6 illustrated (and which has been seen in many earlier calculations, e.g., [4], [9], [23], [35], [36]), the interpolation error when using RBF-Direct often decreases monotonically with decreasing ε until some point $\varepsilon = \varepsilon_{ic}$ when disastrous ill-conditioning kicks in. This has frequently raised the question of whether still much better accuracy would be attained if the ill-conditioning somehow could be eliminated. Previous results using the contour-Padé algorithm [14], [24] have shown that this sometimes can be the case. With RBF-QR, we can now extend such tests to much larger numbers of nodes. What the results in Figures 4.4 and 4.6 show is that the trend of accuracy improvement (with decreasing ε) can get broken even without ill-conditioning playing a role, although typically in a less abrupt way. There will often be a quite well defined error minimum at some location ε_{opt}. In the presently chosen test cases for interpolation, it so happened that $\varepsilon_{ic} \approx \varepsilon_{opt}$, whereas in other contexts, e.g., solving elliptic equations [24] or generating scattered-node finite-difference-type stencils [46], it often happened that $\varepsilon_{ic} > \varepsilon_{opt}$. Major improvements were then achieved by computing well into a regime that was not reachable with RBF-Direct.

6. Conclusions.

The recent work by Flyer and Wright [7] clearly demonstrated the strengths of RBF methods for solving convective-type PDEs over spherical geometries (computationally, the most difficult type of PDEs since they are dissipation-free; also the most important case for many geophysical applications). The best accuracy was then obtained when the basis functions were so flat (condition number for the RBF-Direct approach often around or above 10^8) that the possibility of adverse effects from ill-conditioning could not be ignored.

We have here presented a new computational algorithm RBF-QR that can overcome this ill-conditioning even in the $\varepsilon \to 0$ limit, thereby allowing a more extensive study of how the choice of this shape parameter will affect computational accuracy. The present test cases for interpolation have been followed up by tests for both short- and long-time integration of a convective PDE [13]. While RBF-QR is the second algorithm (following contour-Padé [14]) that allows stable computations when $\varepsilon \to 0$, it is the first one which is practical in the case of thousands of data points on the surface of the sphere.

It follows from the RBF-QR algorithm that $\varepsilon \to 0$ leads to the same results as when using SPH basis functions. One might therefore ask why not just use SPH as a computational basis on the sphere. There are several reasons for not doing that:

- The limit of $\varepsilon \to 0$ is often not the best parameter choice.
- RBF can combine spectral accuracy with local refinement wherever this is needed (cf. discussion on RBF Runge phenomenon in [16]); SPH offers no such opportunities.
- Nonsingularity is guaranteed whenever $\varepsilon > 0$ but not for all node sets if $\varepsilon = 0$.

7. Appendix: A Matlab code for the RBF-QR algorithm.

The test code below computes and then plots the RBF interpolant to n data points. These data are obtained from evaluating a test function at n random locations, marked as black dots in Figure 7.1. The test function and the error (difference between it and the interpolant) are also plotted. The shape parameter ε, the radial function, the number of data points to interpolate, the resolution of the grid, and the test function can all be easily modified in the initial driver part of the code. In the code listings below, the

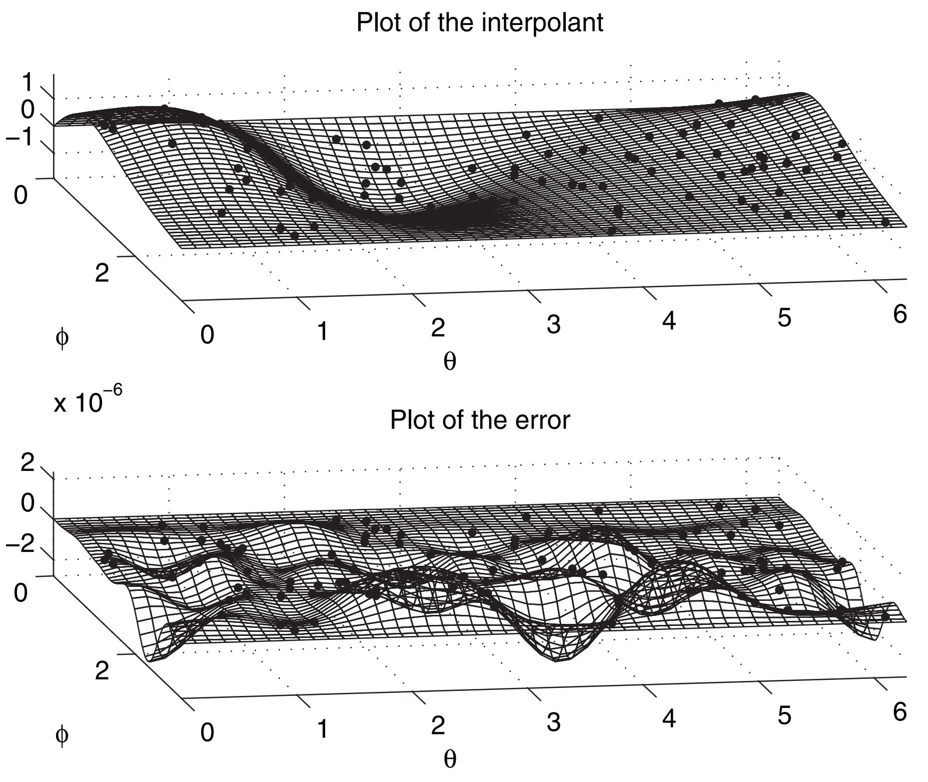

Fig. 7.1. *The graphical outout of the demo code, showing the interpolation error to be of the order* 10^{-6} *when* $\varepsilon = 10^{-8}$ *in the* $n = 100$ *node test problem.*

driver code is given first, followed by the main function RBF-QR and two supporting functions, named COEF and SPH. The function COEF evaluates expansion coefficients according to the formulas in Table 3.2, and the routine SPH evaluates spherical harmonics basis functions at specified locations. The code produces the output shown in Figure 7.1.

```
% =============================== DRIVER CODE ======================================
clear all; close all
epsilon = 10^-6;
rbf = 'IQ';                   % Basis function; valid choices: 'MQ','IMQ','IQ','GA'
n = 100;                      % Number of points to interpolate
rand('seed',4078)             % Create n random node locations
theta = 2*pi*rand(1,n); randCos = 2*rand(1,n)-1; phi = acos(randCos);
[x,y,z] = sph2cart(theta,phi-pi/2,1);
fi = @(x,y,z) x.*exp(y-z);    % Test function to interpolate
res = 50; m = res^2;          % Resolution of the grid for evaluating the interpolant
f = fi(x,y,z);                % Evaluate the data values to interpolate

% ___________________ Evaluation of the interpolant by RBF-QR ______________________
[theta_grid,phi_grid] = meshgrid(linspace(0,2*pi,res),linspace(0,pi,res));
theta_eval = reshape(theta_grid,1,res^2); phi_eval = reshape(phi_grid,1,res^2);
[xe,ye,ze] = sph2cart(theta_eval,phi_eval-pi/2,1);fe = fi(xe,ye,ze);

[beta R] = RBFQR(theta,phi,epsilon,f,rbf); index = 1;
```

```
for mu = 0:sqrt(size(R,1))-1 % Each loop adds a block of columns of SPH of order mu
                             % to Y, evaluated at the grid points
    Y(:,index:2*mu+index) = SPH(mu,theta_eval,phi_eval);
    index = index + 2*mu + 1;
end
f_RBFQR = (Y*R*beta)';       % Call to RBFQR routine

% _________________ Plot of the interpolant and of the error _____________________
colormap(gray);

subplot(2,1,1)
surf(theta_grid,phi_grid,reshape(f_RBFQR,res,res),'FaceColor','none','LineWidth',0.05)
axis([0 2*pi 0 pi min(f_RBFQR) max(f_RBFQR)]); hold on;

plot3(theta,phi,f,'k.','MarkerSize',10); title('Plot of the interpolant');
view([-10,50]);xlabel('\phi'); ylabel('\theta'); set(gca,'ydir','reverse');

subplot(2,1,2)
surf(theta_grid,phi_grid,reshape((f_RBFQR-fe),res,res),'FaceColor','none',...
    'LineWidth',0.05)
axis([0 2*pi 0 pi min((f_RBFQR-fe)) max((f_RBFQR-fe))]); hold on;

plot3(theta,phi,zeros(size(f)),'k.','MarkerSize',10); title('Plot of the error');
view([-10,50]); xlabel('\phi'); ylabel('\theta'); set(gca,'ydir','reverse');



% ============================ FUNCTION RBFQR ==================================
function [beta, R_new] = RBFQR(theta,phi,epsilon,f,rbf)
% This function finds the RBF interpolant, with shape parameter epsilon, through the
% n node points (theta,phi) with function values f. It outputs beta, the expansion
% coefficients of the interpolant with respect to the RBF_QR basis. It calls the
% functions SPH(), which gives spherical harmonic values and COEF() which provides the
% expansion coefficients.

n = length(theta); Y = zeros(n); B = zeros(n);
mu = 0; index = 1; orderDifference = 0;
mu_n = ceil(sqrt(n))-1;             %the order of the n_th spherical harmonic

while orderDifference < -log10(eps) %eps is the machine precision
    % Each loop adds a block of columns of SPH of order mu to Y and to B.

    % Compute the spherical harmonics matrix
    Y(:,index:2*mu+index) = SPH(mu,theta,phi);

    % Compute the expansion coefficients matrix
    B(:,index:2*mu+index) = Y(:,index:2*mu+index)*COEF(mu,epsilon,rbf);
    B(:,index+mu) = B(:,index+mu)/2;

    % Truncation criterion
    if mu > mu_n-1
        orderDifference = log10(norm(B(:,[mu_n^2+1:(mu_n+1)^2]),inf)/...
            norm(B(:,(mu+1)^2),inf)*epsilon^(2*(mu_n-mu)));
    end

    index = index+2*mu+1; mu = mu+1;   % Calculate column index of next block
end

[Q,R] = qr(B);                         % QR-factorization to find the RBF_QR basis

E = epsilon.^(2*(repmat(ceil(sqrt(n+1:mu^2))-1,n,1) - ...  % Introduce the
    repmat(ceil(sqrt(1:n))-1,mu^2-n,1)'));                 % powers of epsilon
```

```
%Solve the interpolation linear system
R_new = [eye(n),E.*(R(1:n,1:n)\R(1:n,n+1:end))]'; beta = Y*R_new\f';


% ============================== FUNCTION COEF ===================================
function c_mu = COEF(mu,epsilon,rbf)
% Returns the expansion coefficients in the cases of MQ, IMQ and GA radial functions.
switch rbf
    case 'MQ'
        c_mu = -2*pi*(2*epsilon^2+1+(mu+1/2)*sqrt(1+4*epsilon^2))/...
               (mu+1/2)/(mu+3/2)/(mu-1/2)*(2/(1+sqrt(4*epsilon^2+1)))^(2*mu+1);
    case 'IMQ'
        c_mu = 4*pi/(mu+1/2)*(2/(1+sqrt(4*epsilon^2+1)))^(2*mu+1);
    case 'IQ'
        c_mu = 4*pi^(3/2)*factorial(mu)/gamma(mu+3/2)/(1+4*epsilon^2)^(mu+1)*...
               hypergeom([mu+1,mu+1],2*mu+2,4*epsilon^2/(1+4*epsilon^2));
    case 'GA'
        c_mu = 4*pi^(3/2)*exp(-2*epsilon^2)*besseli(mu+1/2,2*epsilon^2)/...
               epsilon^(2*mu+1);
end


% ============================= FUNCTION SPH ==================================
function SPHBlockMu = SPH(mu,theta,phi)
% Returns a matrix containing the spherical harmonics of order mu, evaluated at the
% (theta,phi) node points.
n = length(theta);L_mu_nu(:,1:mu+1) = legendre(mu,cos(phi))'; a = 0:mu;
t = repmat(sqrt(factorial(1+mu-a-1)./factorial(1+mu+a-1)),n,1) ...
    .*L_mu_nu(:,a+1).*exp(i*repmat(a,n,1).*repmat(theta',1,mu+1));
SPHBlockMu = sqrt((2*mu+1)/(4*pi))*[imag(t(:,end:-1:2)),real(t)];
```

Acknowledgments. Discussions with Natasha Flyer, Grady Wright, and Elisabeth Larsson are gratefully acknowledged. Comments from two referees have been very helpful in making our presentation more clear.

REFERENCES

[1] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, 2nd ed., Dover, New York, 2001.

[2] G. L. Browning, P. N. Hack, and A. Swarztrauber, *A comparison of three different numerical methods for solving differential equations on the sphere*, Monthly Weather Review, 117 (1989), pp. 1058–1075.

[3] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations*, Cambridge University Press, London, 2003.

[4] R. Carlson and T. A. Foley, *The parameter R2 in multiquadric interpolation*, Comput. Math. Appl., 21 (1991), pp. 29–42.

[5] J. R. Driscoll and D. M. Healy, *Computing Fourier transforms and convolutions on the 2-sphere*, Adv. Appl. Math., 15 (1994), pp. 202–250.

[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 and G. Wright, *Transport schemes on a sphere using radial basis functions*, J. Comput. Phys., 226 (2007), pp. 1059–1084.

[8] N. Flyer and G. Wright, *Solving the Nonlinear Shallow Water Wave Equations Using Radial Basis Functions*, manuscript.

[9] T. A. Foley, *Near optimal parameter selection for multiquadric interpolation*, J. Appl. Sci. Comput., 1 (1994), pp. 54–69.

[10] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, Cambridge, 1996.

[11] B. Fornberg and N. Flyer, *Accuracy of radial basis function interpolation and derivative approximations on 1-D infinite grids*, Adv. Comput. Math., 23 (2005), pp. 5–20.

[12] B. Fornberg and D. Merrill, *Comparison of finite difference and pseudospectral methods for convective flow over a sphere*, Geophys. Res. Lett., 24 (1997), pp. 3245–3248.

[13] 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., to appear.

[14] 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.

[15] 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.

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

[17] F. X. Giraldo and T. E. Rosmond, *A scalable spectral element Eulerian atmospheric model (SEE-AM) for NWP: Dynamical core tests*, Monthly Weather Review, 132 (2004), pp. 133–153.

[18] J. J. Hack and R. Jakob, *Description of a Global Shallow Water Model Based on the Spectral Transform Method*, Technical note TN 343 STR NCAR, 1992.

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

[20] S. Hubbert and B. Baxter, *Radial basis functions for the sphere*, in Progress in Multivariate Approximation, Internat. Ser. Numer. Math. 137, Birkhauser, Boston, 2001, pp. 33–47.

[21] E. J. Kansa, *Multiquadrics - A scattered data approximation scheme with applications to computational fluid-dynamics.* I. *Surface approximations and partial derivative estimates*, Comput. Math. Appl., 19 (1990), pp. 127–145.

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

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

[24] E. Larsson and B. Fornberg, *A numerical study of radial basis function based solution methods for elliptic PDEs*, Comput. Math. Appl., 46 (2003), pp. 891–902.

[25] E. Larsson and B. Fornberg, *A Stable Algorithm for Flat Radial Basis Functions*, manuscript.

[26] 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.

[27] 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.

[28] J. Y. McLeod and M. L. Baart, *Geometry and Interpolation of Curves and Surfaces*, Cambridge University Press, Cambridge, 1998.

[29] P. E. Merilees, *The pseudospectral approximation applied to the shallow water equation on a sphere*, Atmosphere, 11 (1973), pp. 13–20.

[30] P. E. Merilees, *Numerical experiments with the pseudospectral method in spherical coordinates*, Atmosphere, 12 (1974), pp. 77–96.

[31] M. Mohlenkamp, *A fast transform for spherical harmonics*, J. Fourier Anal. Appl., 5 (1999), pp. 159–184.

[32] D. Potts, G. Steidl, and M. Tasche, *Fast and stable algorithms for discrete spherical Fourier transforms*, Linear Algebra Appl., 275–276 (1998), pp. 433–450.

[33] M. J. D. Powell, *The theory of radial basis function approximation in* 1990, in Advances in Numerical Analysis, Vol. II: Wavelets, Subdivision Algorithms and Radial Functions, W. Light, ed., Oxford University Press, Oxford, 1990, pp. 105–210.

[34] V. Rokhlin and M. Tygert, *Fast algorithms for spherical harmonic expansions*, SIAM J. Sci. Comput., 27 (2005), pp. 1903–1928.

[35] 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.

[36] 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.

[37] R. Schaback, *Comparisons of radial basis function interpolants*, in Multivariate Approximation: From CAGD to Wavelets, K. Jetter and F. I. Utreras, eds., World Scientific, Singapore, 1993, pp. 293–305.

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

[39] W. F. Spotz, M. A. Taylor, and P. N. Swarztrauber, *Fast shallow water equation solvers in latitude-longitude coordinates*, J. Comput. Phys., 145 (1998), pp. 432–444.

[40] P. N. Swarztrauber, *Spectral transform methods for solving the shallow-water equations on the sphere*, Monthly Weather Review, 124 (1996), pp. 730–744.

[41] M. Taylor, J. Tribbia, and M. Iskandarani, *The spectral element method for the shallow water equations on the sphere*, J. Comput. Phys., 130 (1997), pp. 92–108.

[42] C. Temperton, *On scalar and vector transform methods for global spectral models*, Monthly Weather Review, 119 (1991), pp. 1303–1307.

[43] S. J. Thomas and R. D. Loft, *The NCAR spectral element climate dynamical core: Semi-implicit, Eulerian formulation*, J. Sci. Comput., 25 (2005), pp. 307–322.

[44] L. N. Trefethen, *Spectral Methods in Matlab*, SIAM, Philadelphia, 2000.

[45] R. S. Womersley and I. Sloan, *Interpolation and Cubature on the Sphere*, http://web.maths.unsw.edu.au/rsw/Sphere/.

[46] G. Wright and B. Fornberg, *Scattered node compact finite difference-type formulas generated from radial basis functions*, J. Comput. Phys., 212 (2006), pp. 99–123.

[47] J. Yoon, *Spectral approximation orders of radial basis function interpolation on the Sobolev space*, SIAM J. Math. Anal., 33 (2001), pp. 946–958.