STABLE COMPUTATION OF DIFFERENTIATION MATRICES AND SCATTERED NODE STENCILS BASED ON GAUSSIAN RADIAL BASIS FUNCTIONS

ELISABETH LARSSON*, ERIK LEHTO*, ALFA HERYUDONO†, AND BENGT FORNBERG‡

Abstract. Radial basis function (RBF) approximation has the potential to provide spectrally accurate function approximations for data given at scattered node locations. For smooth solutions, the best accuracy for a given number of node points is typically achieved when the basis functions are scaled to be nearly flat. This also results in nearly linearly dependent basis functions and severe ill-conditioning of the interpolation matrices. Fornberg, Larsson, and Flyer recently generalized the RBF-QR method to provide a numerically stable approach to interpolation with flat and nearly flat Gaussian RBFs for arbitrary node sets in up to three dimensions. In this work, we consider how to extend this method to the task of computing differentiation matrices and stencil weights in order to solve partial differential equations. The expressions for first and second order derivative operators as well as hyperviscosity operators are established, numerical issues such as how to deal with non-unisolvency are resolved, and the accuracy and computational efficiency of the method are tested numerically. The results indicate that using the RBF-QR approach for solving PDE problems can be very competitive compared with using the ill-conditioned direct solution approach or using variable precision arithmetic to overcome the conditioning issue.

Key words. radial basis function, flat limit, ill-conditioning, differentiation matrix, stencil weight, RBF, RBF-QR, RBF-FD

AMS subject classifications. 65D15, 65D25

1. Introduction. Radial basis function (RBF) approximation [1, 31, 4] is emerging as an important method class for interpolation, approximation, and solution of partial differential equations (PDEs) for data given at scattered node locations, with non-trivial geometry, or with computational domains in higher dimensions. The main advantages are the spectral convergence rates that can be achieved using infinitely smooth basis functions, the geometrical flexibility, and the ease of implementation. However, in practical cases, convergence has often been hampered by ill-conditioning as the shape of the basis functions become flatter. The best accuracy for smooth and well resolved solutions is often found in this regime [18, 13, 19]. Therefore, moving to larger shape parameter values (less flat RBFs) is not a desirable solution to the conditioning problem.

The first method that allowed stable computations in the flat RBF regime was the Contour-Padé method derived by Fornberg and Wright [13]. The method works in any number of dimensions, but for relatively low numbers of nodes. Except for the approach in [24], limited to Gaussian RBFs on an equispaced grid in one dimension, the next method that was developed was the RBF-QR method, which was first derived for nodes on the surface of the sphere by Fornberg and Piret [12] and then for general node distributions in up to three dimensions [9]. The RBF-QR methods can be employed for approximations over thousands of nodes.

In [9], we gave examples of convergence and performance results in the case of interpolation. The next step is to use the RBF-QR method also for computing differentiation matrices, stencil weights and hyperviscosity operator matrices to

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

†University of Massachusetts Dartmouth, Department of Mathematics, 285 Old Westport Road, Dartmouth, MA 02747, USA (aheryudono@umassd.edu). The work was supported by the European Commission CORDIS Marie Curie FP7 program Grant #235730.

‡University of Colorado, Department of Applied Mathematics, 526 UCB, Boulder, CO 80309, USA (fornberg@colorado.edu). The work was supported by the NSF Grant DMS-0914647.

be used for example when solving PDE problems. By differentiation matrix we typically mean the matrix that connects a certain derivative of a function evaluated at the node points to the function values at the node points, for example

$$\frac{\partial}{\partial x}\underline{u} = D_x\underline{u},$$

where $\underline{u}$ is the vector of function values and D_x is the differentiation matrix. Equality holds for functions in the generating space, which here could be the space spanned by RBFs centered at the node points. For other functions, the relation is an approximation. Differentiation matrices are needed when solving PDEs using global RBF approximations [24, 8, 5, 23, 6] or solving PDEs using partition of unity based RBF methods [20]. Stencil weights are a special case of differentiation matrix, where the derivative value only at one (often central) node point of the stencil is considered. This is the approach used in RBF generated finite difference methods (RBF-FD), which generalize finite difference methods to scattered nodes [29, 28, 30, 2, 32, 26]. It is also in the RBF-FD case that hyperviscosity operators are of interest. A hyperviscosity operator matrix corresponds to the differentiation matrix for a (high-order) power of the Laplacian. These operators are employed for stabilization purposes, especially in the case of purely convective PDEs [10, 7]. Hyperviscosity operators can also be used together with global or partition of unity RBF approximations. However, in our experience, the approach employing the inverse of the interpolation operator, also introduced in [10] is more efficient in these cases.

The recent study [11] describes a somewhat related approach for the stable calculation of Gaussian RBF-based stencils. Both approaches involve a change from the original RBF basis to a more stable basis. In the RBF-QR method this involves truncation of an infinite expansion, whereas in [11] the change of basis is performed by using an identity involving the incomplete gamma function without any truncation. The present RBF-QR version is computationally slightly slower, but provides higher accuracy, especially for large stencil sizes. Computing Gaussian RBF-based stencils for the Laplacian using the RBF-QR method was discussed by Davydov and Oanh in [3].

The main contributions in this paper are that we derive expressions for evaluation of first order, second order, and hyperviscosity operators within the RBF-QR framework; that we design accurate and robust numerical implementations of the evaluation, including strategies to deal with non-unisolvent node sets; that we provide transparent and practical user interfaces for typical use cases to these implementations; and that we demonstrate relevant properties of the derivative approximations through numerical experiments.

The outline of the paper is as follows. In Section 2 we briefly review the RBF-QR method. Section 3 describes how to compute the differentiation matrices. Section 4 contains a discussion of the limit case when the RBFs become flat for non-unisolvent node sets, which requires some special attention, together with other implementation issues. We then present numerical results and the paper ends with a short summary. The exact formulas for the necessary derivatives are given in Appendix A.

2. The RBF-QR method. A standard radial basis function interpolant to data f_j given at the scattered nodes $\underline{x}_j$, $j = 1, \ldots, N$ has the form

$$s_\varepsilon(\underline{x}) = \sum_{j=1}^{N} \lambda_j \phi(\varepsilon\|\underline{x} - \underline{x}_j\|) \equiv \sum_{j=1}^{N} \lambda_j \phi_j(\underline{x}), \tag{2.1}$$

where $\phi(r)$ is a radial basis function, and ε is a shape parameter. Decreasing ε leads to flatter basis functions. In this paper, we consider Gaussian radial basis

functions, i.e., $\phi(r) = e^{-r^2}$. The unknown coefficients λ_j are determined through the interpolation conditions $s_\varepsilon(\underline{x}_i) = f_i$, $i = 1, \ldots, N$, leading to a linear system of equations

$$A_\phi \underline{\lambda} = \underline{f}, \tag{2.2}$$

where the symmetric matrix A_ϕ has elements $a_{ij} = \phi_j(\underline{x}_i) = \phi(\varepsilon \|\underline{x}_i - \underline{x}_j\|)$, $\underline{\lambda} = (\lambda_1, \ldots, \lambda_N)^T$, and $\underline{f} = (f_1, \ldots, f_N)^T$. If the data f_j is sampled from a smooth function, highly accurate interpolation results are typically achieved for small values of ε. However, as $\varepsilon \to 0$, the basis functions become nearly flat and the matrix A_ϕ is nearly singular.

The purpose of stable RBF algorithms is to evaluate $s_\varepsilon(\underline{x})$ by steps that all remain numerically well-conditioned however small ε becomes. The main idea behind the RBF-QR algorithm is to recognize that the RBFs themselves constitute an ill-conditioned basis in a good approximation space. We proceed to make a change of basis by first expanding the RBFs in terms of other expansion functions V_k, $k = 1, \ldots$. The particular functions that we use are combinations of polynomial powers, Chebyshev polynomials, and trigonometric functions, see Appendix A. The expansions are truncated at $k = M \geq N$ based on the size of the contributions, for details see [9]. This yields the relations

$$\begin{pmatrix} \phi_1(\underline{x}) \\ \vdots \\ \phi_N(\underline{x}) \end{pmatrix} \approx \begin{pmatrix} c_1(\underline{x}_1) & \cdots & c_M(\underline{x}_1) \\ \vdots & & \vdots \\ c_1(\underline{x}_N) & \cdots & c_M(\underline{x}_N) \end{pmatrix} \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_M \end{pmatrix} \begin{pmatrix} V_1(\underline{x}) \\ \vdots \\ V_M(\underline{x}) \end{pmatrix}, \tag{2.3}$$

where the elements of the coefficient matrix C are $\mathcal{O}(1)$ and the element d_k of the scaling matrix D is proportional to ε^{2m_k}, where $m_{k+1} \geq m_k$.

Then, we QR-factorize the coefficient matrix to get $C = Q\,(R_1\; R_2)$, where R_1 is upper triangular and contains the N first columns of R. The scaling matrix D is partitioned correspondingly with the diagonal blocks D_1 of size $N \times N$ and D_2 of size $(M - N) \times (M - N)$. The new basis is then obtained as

$$\begin{pmatrix} \psi_1(\underline{x}) \\ \vdots \\ \psi_N(\underline{x}) \end{pmatrix} = D_1^{-1} R_1^{-1} Q^T \begin{pmatrix} \phi_1(\underline{x}) \\ \vdots \\ \phi_N(\underline{x}) \end{pmatrix} \approx \begin{pmatrix} I_N & D_1^{-1} R_1^{-1} R_2 D_2 \end{pmatrix} \begin{pmatrix} V_1(\underline{x}) \\ \vdots \\ V_M(\underline{x}) \end{pmatrix}, \tag{2.4}$$

where I_N is the unit matrix of size $N \times N$ and the correction matrix $\tilde{R} = D_1^{-1} R_1^{-1} R_2 D_2$ only contains non-negative powers of ε due to the ordering of the scaling coefficients. Since the new basis functions $\Psi_j(\underline{x})$ depend linearly on the expansion functions, we can easily calculate the action of a linear differential operator $\mathcal{L}$ on the basis functions through

$$\begin{pmatrix} \mathcal{L}\psi_1(\underline{x}) \\ \vdots \\ \mathcal{L}\psi_N(\underline{x}) \end{pmatrix} = \begin{pmatrix} I_N & \tilde{R} \end{pmatrix} \begin{pmatrix} \mathcal{L}V_1(\underline{x}) \\ \vdots \\ \mathcal{L}V_M(\underline{x}) \end{pmatrix}. \tag{2.5}$$

To compute the (unsymmetric) interpolation matrix A_Ψ with elements $a_{ij} = \psi_j(\underline{x}_i)$, we apply the transpose of relation (2.4) at each evaluation point $\underline{x}_i$, $i = 1, \ldots, N$ to get

$$A_\Psi = V \begin{pmatrix} I_N \\ \tilde{R}^T \end{pmatrix}, \tag{2.6}$$

where the matrix V has elements $v_{ij} = V_j(\underline{x}_i)$, $i = 1, \ldots, N$, $j = 1, \ldots, M$. Similarly, we can compute matrices $B_{\mathcal{L}\Psi}$, where an operator $\mathcal{L}$ is applied to the basis functions and where the evaluation points $\underline{x}_i$ may differ from the node points both in location and numbers.

3. Computing differentiation matrices. Whether we are using global RBF approximations, partitioned RBF approximations [20], or RBF-FD [32, 10], there are many situations where we need to compute derivative approximations based on the function values at the node points.

Assume that we want to apply a differential operator $\mathcal{L}$ to a solution function $u(\underline{x})$ at a set of evaluation points $Y = \{\underline{y}_i\}_{i=1}^M$, given the solution values at the set of node points $X = \{x_j\}_{j=1}^N$. The differentiation matrix D relates the two through

$$\mathcal{L}\underline{u}_Y \approx D\underline{u}_X, \tag{3.1}$$

where $\mathcal{L}\underline{u}_Y = (\mathcal{L}u(\underline{y}_1), \ldots, \mathcal{L}u(\underline{y}_M))^T$, and $\underline{u}_X = (u(\underline{x}_1), \ldots, u(x_N))^T$. In the following subsections, two equivalent ways of constructing and understanding RBF based differentiation matrices are described.

3.1. Differentiating RBF approximants. Applying the interpolation relation (2.2), using basis functions $\psi_j(\underline{x})$, $j = 1, \ldots, N$, for data $\underline{u}_X$ yields

$$A_\Psi \underline{\mu} = \underline{u}_X. \tag{3.2}$$

We can approximate any derivative or linear combination of derivatives of $u(\underline{x})$ by differentiating the RBF interpolant. That is,

$$\mathcal{L}\underline{u}_Y \approx B_{\mathcal{L}\Psi}\underline{\mu} = B_{\mathcal{L}\Psi}A_\Psi^{-1}\underline{u}_X,$$

where $B_{\mathcal{L}\Psi}$ has elements $b_{ij} = \mathcal{L}\psi_j(\underline{y}_i)$, $i = 1, \ldots, M$, $j = 1, \ldots, N$. The differentiation matrix D for the operator $\mathcal{L}$ is hence given by

$$D = B_{\mathcal{L}\Psi}A_\Psi^{-1}. \tag{3.3}$$

For positive definite RBFs such as Gaussians, the matrix A_Φ in (2.2) is guaranteed to be non-singular for distinct node points and $\varepsilon > 0$. The corresponding A_Ψ is non-singular if the change of basis is well-defined, which will be further discussed in section 4. For conditionally positive definite RBFs, the interpolant may need to be augmented with a polynomial term in order to guarantee non-singularity. However, even in the positive definite case, there are situations where it is beneficial with respect to accuracy and convergence to add a low order polynomial term [10]. We let the RBF interpolant expressed in the original basis functions take the form

$$s(\underline{x}) = \sum_{j=1}^N \lambda_j \phi_j(\underline{x}) + \sum_{j=1}^m \alpha_j p_j(\underline{x}), \tag{3.4}$$

with the additional constraints on the coefficients

$$\sum_{j=1}^N \lambda_j p_k(\underline{x}_j) = 0, \quad k = 1, \ldots, m. \tag{3.5}$$

The modified system of equations corresponding to (2.2) becomes

$$\begin{pmatrix} A_\Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \underline{\lambda} \\ \underline{\alpha} \end{pmatrix} = \begin{pmatrix} \underline{u}_X \\ 0 \end{pmatrix}, \tag{3.6}$$

where the matrix P has elements $p_{ij} = p_j(\underline{x}_i)$, $i = 1, \ldots, N$, $j = 1, \ldots, m$. To find the corresponding system for the new basis, we use (2.4) for the relation $A_\Psi = A_\Phi G$, where $G^T = D_1^{-1}R_1^{-1}Q^T$. We also introduce the change of variables $\underline{\lambda} = G\underline{\mu}$ to get

$$\begin{pmatrix} A_\Psi G^{-1} & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \underline{\lambda} \\ \underline{\alpha} \end{pmatrix} = \begin{pmatrix} A_\Psi & P \\ \sigma P^T G & 0 \end{pmatrix} \begin{pmatrix} \underline{\mu} \\ \underline{\alpha} \end{pmatrix} = \begin{pmatrix} \underline{u}_X \\ 0 \end{pmatrix}, \tag{3.7}$$

where the σ is an arbitrary scaling constant that can be added because this part of the system of equations is homogeneous. Evaluating the approximation of the differential operator $\mathcal{L}$ then leads to

$$\mathcal{L}\underline{u}_Y \approx \begin{pmatrix} B_{\mathcal{L}\Psi} & P_{\mathcal{L}} \end{pmatrix} \begin{pmatrix} \underline{\mu} \\ \underline{\alpha} \end{pmatrix} = \begin{pmatrix} B_{\mathcal{L}\Psi} & P_{\mathcal{L}} \end{pmatrix} \begin{pmatrix} A_\Psi & P \\ \sigma P^T G & 0 \end{pmatrix}^{-1} \begin{pmatrix} \underline{u}_X \\ 0 \end{pmatrix}, \quad (3.8)$$

where $P_{\mathcal{L}}$ has elements $p_{ij} = \mathcal{L}p_j(\underline{x}_i)$. The differentiation matrix in this case consists of the first N columns of the resulting matrix. That is,

$$D = \left[\begin{pmatrix} B_{\mathcal{L}\Psi} & P_{\mathcal{L}} \end{pmatrix} \begin{pmatrix} A_\Psi & P \\ \sigma P^T G & 0 \end{pmatrix}^{-1} \right]_{(1:M,1:N)}. \quad (3.9)$$

If we expand the matrix of the constraint equations, we get

$$\sigma P^T G = \sigma (D_1^{-1} R_1^{-1} Q^T P)^T. \quad (3.10)$$

The inverse of the scaling matrix D_1 contains negative powers of ε, and can hence not be evaluated in a numerically stable way for small ε, see also [3]. We can choose σ to cancel all negative powers of ε (this step should be performed analytically) and further to normalize the largest element to one. However, numerical experiments for the case where a constant term is added indicate that the condition number κ of the augmented interpolation matrix depends on the smallest matrix element (after the scaling with σ), $z_{\min}$, in the constraint equation as $\kappa \approx \frac{C}{z_{\min}}$, where C is a constant that depends on the size of the stencil and the node layout. A truncation procedure to improve conditioning is discussed in [3]. We make the following observations

- If the shape parameter is large enough, so that the elements in the constraint equations are not too small, using (3.9) works perfectly fine. The matrices needed for generating G are provided by the RBF-QR algorithm.
- If the shape parameter is small, the RBF interpolant is approaching the flat RBF limit interpolant, which is polynomial [19]. Hence, there is no particular need for adding polynomial terms, since the differentiation matrix will be highly accurate for polynomials due to the limit property.
- Using the same constraint equations for $\underline{\mu}$ as for $\underline{\lambda}$, i.e., replacing λ_j by μ_j in (3.5), leads to a well conditioned interpolation matrix. However, there is no theoretical justification for this choice.

For the differentiation matrices derived above in (3.3) and (3.9), we assumed an exact interpolant to the values $\underline{u}_X$. If we instead assume a least squares fit to data $\underline{u}_{\tilde{X}}$ given at $N_d > N$ locations, we get a rectangular $(N_d \times N)$ matrix A_Ψ in (3.2), which we can QR-factorize to get the differentiation matrix

$$D = B_{\mathcal{L}\Psi} R^{-1} Q^T,$$

in place of (3.3). A similar result can, if needed, be derived for the augmented case while taking care to assure the exact fulfillment of the constraints.

3.2. RBF-generated finite difference stencils. When computing an RBF-FD stencil for the N node points in the set X, we are looking for weights $\underline{w} = (w_1, \ldots, w_N)^T$ such that

$$\mathcal{L}u(\underline{x}_c) \approx \underline{w}^T \underline{u}_X, \quad (3.11)$$

where $\underline{x}_c$ is the central point of the stencil, i.e, the point where the stencil is applied. Comparing with (3.1), we can conclude that the weights correspond to a special case of a differentiation matrix with just one evaluation point, $\underline{x}_c$. However, the construction of stencils is often approached from a different perspective. For an

RBF-generated stencil, we require the stencil to produce a correct result for all functions spanned by the involved RBFs. That is, working with the basis functions $\psi_i(\underline{x})$, $i = 1, \ldots, N$ we have

$$\sum_{j=1}^{N} w_j \psi_i(\underline{x}_j) = \mathcal{L}\psi_i(\underline{x}_c), \quad i = 1, \ldots, N. \tag{3.12}$$

As a linear system of equations for the weights this becomes

$$A_\Psi^T \underline{w} = B_{\mathcal{L}\Psi}^T \quad \text{or} \quad \underline{w}^T = B_{\mathcal{L}\Psi} A_\Psi^{-1}.$$

The second form is identical to (3.3), so in fact, this is just another way to formulate the same problem.

For stencil computations, it can be especially beneficial to augment the stencil with polynomial terms. As an example, including a constant and linear term assures that the convergence will always be at least second order for a smooth enough solution function. In the stencil frame of thought this corresponds to requiring the weights to yield the correct result for constant and linear polynomials. Consider the linear system

$$\left(\begin{array}{ccc|ccc} \phi_1(\underline{x}_1) & \cdots & \phi_1(\underline{x}_N) & p_1(\underline{x}_1) & \cdots & p_m(\underline{x}_1) \\ \vdots & & \vdots & \vdots & & \vdots \\ \phi_N(\underline{x}_1) & \cdots & \phi_N(\underline{x}_N) & p_1(\underline{x}_N) & \cdots & p_m(\underline{x}_N) \\ \hline p_1(x_1) & \cdots & p_1(x_N) & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ p_m(\underline{x}_1) & \cdots & p_m(\underline{x}_N) & 0 & \cdots & 0 \end{array}\right) \left(\begin{array}{c} w_1 \\ \vdots \\ w_N \\ \hline \nu_1 \\ \vdots \\ \nu_m \end{array}\right) = \left(\begin{array}{c} \mathcal{L}\phi_1(\underline{x}_c) \\ \vdots \\ \mathcal{L}\phi_N(\underline{x}_c) \\ \hline \mathcal{L}p_1(\underline{x}_c) \\ \vdots \\ \mathcal{L}p_m(\underline{x}_c) \end{array}\right).$$

The last m rows correspond to making the stencil approximation exact for polynomials spanned by $p_1, \ldots, p_m$. The first N rows correspond to making the approximation good for the part represented by the basis functions $\phi_1, \ldots, \phi_N$. However, because we now have more conditions than we have weights, we cannot fulfill all of them exactly. The part represented by the $\nu_1, \ldots, \nu_m$ coefficients can be viewed as an error. The simplest case is when $m = 1$ and $p_1(\underline{x}) = 1$. The effect then is that we make the same error, ν_1, for each basis function. However, this system, being the equivalent of (3.8) is based on the assumption that the constraints (3.5) are fulfilled by the coefficients $\underline{\lambda}$. Hence, when we take a linear combination $u = \sum_{j=1}^{N} \lambda_j \phi_j(\underline{x})$ and apply the stencil weights, we end up with $\underline{w}^T \underline{u}_X = \sum_{j=1}^{N} \lambda_j \mathcal{L}\phi_j(\underline{x}_c) - \nu_1 \sum_{j=1}^{N} \lambda_j = \sum_{j=1}^{N} \lambda_j \mathcal{L}\phi_j(\underline{x}_c)$ due to the first constraint. For the higher order error terms the size of the error in the individual approximation depends on location, but the same argument in relation to the constraints holds. Therefore, for any approximant of the form (3.4)–(3.5), the errors in the approximations for individual basis functions are cancelled out and the stencil weights compute the correct derivative. We conclude that we can safely ignore the values of ν_j just as we can discard the last columns in (3.9).

To compute stencil weights with added polynomial precision using the RBF-QR basis functions ψ_j we need to modify the part of the matrix corresponding to the constraints, just as for the differentiation matrix (3.9). This means that here the upper right matrix block, which is now P, is replaced by $G^T P$.

3.3. An alternative approach for higher order differentiation. In the papers [17, 16], Fuselier and Wright, demonstrate how RBF differentiation matrices for higher order operators can be computed by combining lower order operators, resulting in considerable reductions of the complexity of the analytic expressions involved, at the price of a somewhat higher computational cost. A less intuitive

result was that the accuracy of the approximations for the composite operators proved to be similar or in some cases even better than when applying the high order operator directly.

We describe how to implement this approach by an example. Consider the operator $\mathcal{L} = \frac{\partial^2}{\partial x^2} = \frac{\partial}{\partial x}\frac{\partial}{\partial x} = \mathcal{L}_1\mathcal{L}_2$. Given data at the node set X, we form an RBF interpolant, we differentiate the interpolant, and evaluate the first derivative at the node points of X. This gives us new data (representing the first derivative) at the node points, which we interpolate, differentiate, and evaluate at the node set Y. The composite differentiation matrix corresponding to the direct version in (3.3) then becomes

$$D = (B^Y_{\mathcal{L}_1\Psi} A^{-1}_\Psi)(B^X_{\mathcal{L}_2\Psi} A^{-1}_\Psi). \tag{3.13}$$

This can be done analogously for stencils or in combination with polynomial terms.

4. Implementation details. The algorithms described here have been implemented in MATLAB for nodes in one, two, and three space dimensions. The MATLAB codes are freely available for downloading from the first author's website. Several different interfaces depending on the intended use are currently provided. For example, a differentiation matrix or a set of stencil weights can be obtained by the call

```
D=RBF_QR_diffmat_2D(op,xe,xk,ep,tol);
```

where `op` indicates the operator under consideration such as `'L'` for the Laplacian or `'x'` for the first derivative in the x-direction, `xe` contains the evaluation point(s), `xk` the node points, `ep` the shape parameter, and `tol` is a tolerance for the pivoting described below.

It is also possible to compute matrices such as A_Ψ and $B_{\mathcal{L}\Psi}$ separately by subsequently calling

```
Psi=InitPsi_2D(ep,xk,tol);
```

which constructs the RBF-QR basis functions, and

```
A=RBF_QR_mat_2D(Psi,op,xe);
```

which builds an RBF-QR matrix according to the specified operator and evaluation points. However, in this case, the points are assumed to be given in polar form and scaled to fall within the unit disc.

In a general case, the implementation of the algorithms for computing differentiation matrices as described in the previous sections is quite straightforward. However, in the following subsections we cover the practical aspects of how to deal with non-unisolvent node sets and how to reduce the computational cost for high order hyperviscosity operators.

4.1. Pivoting for non-unisolvent node sets. For Gaussian RBFs, the interpolation matrix is always non-singular for distinct node points. In the flat limit, when $\varepsilon \to 0$, most smooth RBFs can diverge for non-unisolvent point sets [14, 19, 27, 21]. However, Gaussian RBFs never diverge [27]. This property does not automatically carry over to the RBF-QR basis. For the change of basis to be valid, we need R_1 in (2.4) to be non-singular. This in turn requires the N first columns of the coefficient matrix C to be linearly independent.

We illustrate what happens for a non-unisolvent point set with an example. Let all node point $\underline{x}_k = (x_k, y_k)$ lie on the line $x = y$. Then in polar coordinates $r_k = \sqrt{2}|x_k|$ and $\theta_k = \frac{\pi}{4}$ or $\frac{\pi}{4} + \pi$. The kth row in the matrix C is given by

$$[c_{0,0}(\underline{x}_k)\ c_{1,0}(\underline{x}_k)\ s_{1,0}(\underline{x}_k)\ c_{2,0}(\underline{x}_k)\ c_{2,1}(\underline{x}_k)\ s_{2,1}(\underline{x}_k)\ c_{3,0}(\underline{x}_k)\ \cdots],$$

where

$$c_{j,m}(\underline{x}_k) = F_{j,m}(\varepsilon, r_k)\cos((2m+p)\theta_k),$$
$$s_{j,m}(\underline{x}_k) = F_{j,m}(\varepsilon, r_k)\sin((2m+p)\theta_k),$$

where $p = j \mod 2$. For our particular node set, this means for example that $c_{j,0} = s_{j,0}$ for all odd j, since $\sin\theta_k = \cos\theta_k$, leading to pairwise equal columns in C. We also have cases like, $c_{j,1} = s_{j,2} = 0$ for all even j, since $\cos 2\theta_k = \sin 4\theta_k = 0$, leading to a number of zero columns in C.

The problem of linearly dependent or zero columns can be solved by using pivoting in the QR-factorization. However, we cannot allow unconstrained pivoting because we need to keep the columns sorted according to the power of ε in the scaling coefficient as described in section 2. The scaling coefficients appear in groups with common powers of ε [15, 9]. Table 4.1 indicates the pattern in different numbers of dimensions.

TABLE 4.1
The number of scaling coefficients with a certain power of ε in different numbers of dimensions.

Power	0	2	4	6	8	2j
1-D	1	1	1	1	1	1
2-D	1	2	3	4	5	(j+1)/1!
3-D	1	3	6	10	15	(j+1)(j+2)/2!

We implement a selective pivoting strategy within the QR-algorithm, where each group of columns with similar scaling is examined separately until we have found N linearly independent columns. Each group contributes at least one column to Q, since the groups correspond to different polynomial orders. After computing the new tentative contribution to Q and R using pivoting within the group, we check if there are any significant drops in magnitude of the pivot elements (the diagonal elements in R). Due to the pivoting, these are ordered from largest in magnitude to smallest. A detected drop in magnitude is handled with the following strategies:

(i) If the pivot is numerically zero, this column is moved into R_2. This does not cause scaling problems because all the elements from the pivot and down (that would be multiplied by a negative power of ε) are zero.

(ii) Otherwise, the size of the pivot is computed in terms of powers of ε. If the magnitude is $\mathcal{O}(\varepsilon^{2q})$, then the elements from the pivot and down can absorb a multiplication with a negative power ε^{-2q}. This tells us how many blocks further down the line the column can be moved. When we eventually reach that block, the column is reexamined.

Figure 4.1 shows the result of the selection process for points on the line $x = y$ and for points on a grid. For the line, exactly one column per group is chosen as is expected since the effective dimension of the node set is one. The selection for the nodes on a grid also follows the expected pattern, which is $1, \ldots, 10, 11, 10, \ldots, 1$. This is related to the size of the polynomial null spaces over the nodes, see [19]. The described pivoting strategy leads to a stable algorithm, and in some cases we will end up with the true limit for a Gaussian RBF interpolant as $\varepsilon \to 0$. However, as shown in [19] for non-unisolvent node sets, the limit interpolant is a very specific RBF-dependent combination of polynomial powers. The pivoting strategy cannot in general determine which basis functions of a certain degree should be selected in order to produce a Gaussian limit. Any linearly independent combination will work. This means that as ε goes to zero, there will be some point where the RBF-QR algorithm switches from the true Gaussian path to the limit given by the pivoting choice. Note that even though this limit is different it does not imply that it is a worse choice than the Gaussian limit. Numerical experiments on this are included in Section 5.3.

4.2. Reducing the computational cost for hyperviscosity stencils. For each expansion function V_j, see (2.4), the computation of a hyperviscosity operator requires the evaluation of a sum of polynomials in r, as described in Appendix A. The number of polynomial terms increases with the order of the operator, and the

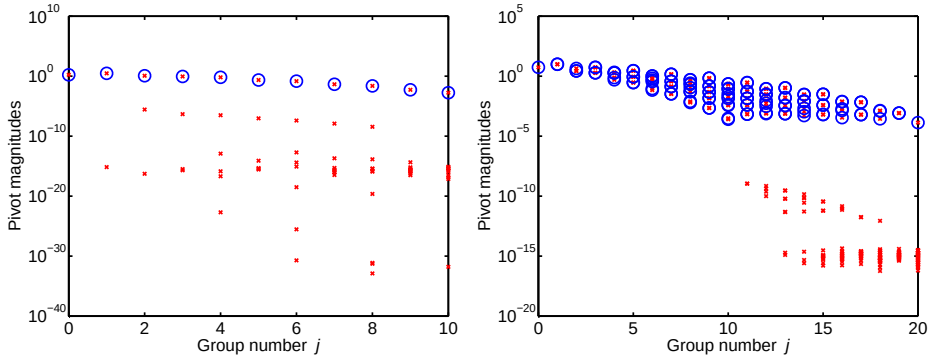

FIG. 4.1. *The magnitude of tentative pivot elements* $\times$ *and selected pivot elements* $\bigcirc$ *for 11 points uniformly distributed over the line* $x = y$ *(left) and* 11×11 *points on a uniform grid (right). The shape parameter value here is* $\varepsilon = 0.1$ *For smaller values, the distance between the magnitudes of the tentative and the selected pivots grows.*

evaluation may constitute a dominant part of the computational cost for high order hyper-viscosity. A significant simplification is possible for stencil computations, where there is only one evaluation point. We are then free to translate the node set such that this point coincides with the origin. By this shift, most terms become identically zero and only a small fraction of them needs to be computed. The simplified expression for evaluation of hyperviscosity at $r = 0$ is given in Appendix A.

Another aspect of computing high order hyperviscosity operators that needs to be taken into account is that we need to increase the number of terms in the expansion of the RBFs compared with the interpolation case in order to retain the accuracy in the computation of the differentiation matrices. Figure 4.2 shows an example of how many extra blocks of expansion functions are needed for different values of the shape parameter.

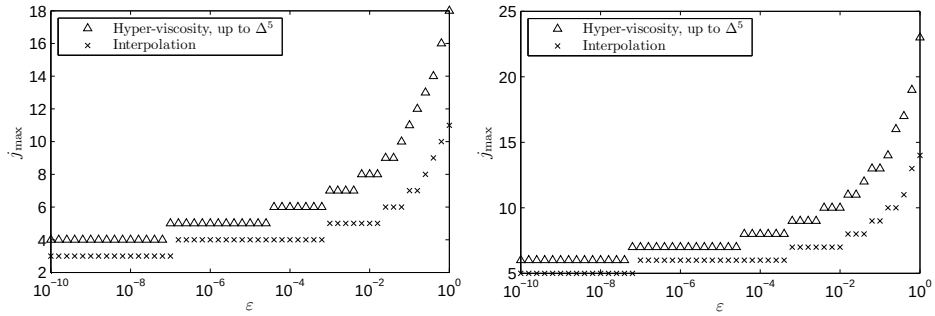

FIG. 4.2. *The parameter* $j_{\max}$ *indicates to which degree expansion functions are included before truncation. The different markers denote results for interpolation (marked* $\times$*) and tenth order hyper-viscosity (marked* Δ*). The results to the left are for stencils with 10 node points and to the right for stencils with 35 node points.*

5. Numerical results. All numerical experiments presented here are performed in MATLAB. In the RBF-FD experiments, the parameter h represents the size of the box where nodes are generated, and is used as a measure of the node density. The results computed with the RBF-QR method are compared in different ways with the results obtained when using equations (2.2) and (2.1) directly. We will denote the direct approach by RBF-Direct. Experiments have been performed for 2-D and 3-D problems, but we do not display all combinations. In most cases the results are similar.

5.1. Accuracy and conditioning. The condition number of the RBF-QR basis for different node sets is shown in Figure 5.1. In agreement with the results in [9], the condition number of the RBF-QR basis shows no significant dependence on the value of the shape parameter in our experiments. The logarithm of the condition number grows proportionally to $\sqrt{N}$ in 2-D and $\sqrt[3]{N}$ in 3-D, which coincides with the growth of the polynomial order of the interpolant in the limit as $\varepsilon \to 0$. Note that for the N-values that are most likely to be of practical interest, the condition numbers stay below 10^5. Figure 5.2 shows the error in the computed stencil

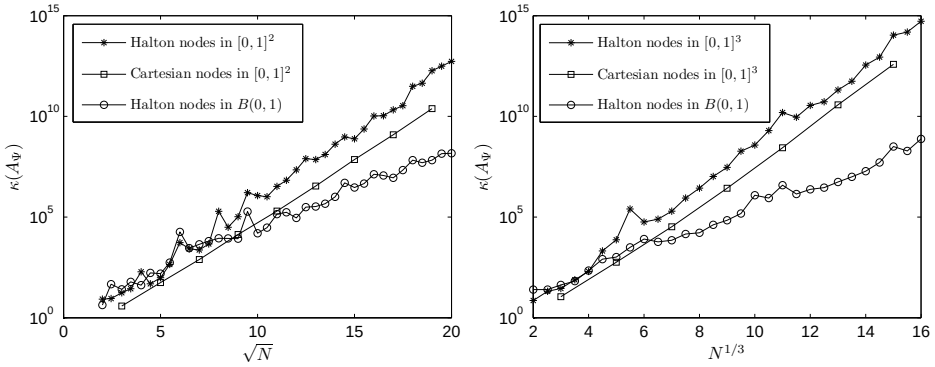

FIG. 5.1. *The condition numbers of the RBF-QR interpolation matrices A_Ψ for up to 400 node points in 2-D (left) and up to 4096 node points in 3-D (right) for different types of node distributions. The notation $B(0,1)$ is used for the unit ball in the respective dimensions*

weights. The exact counterparts are in this comparison obtained using variable precision arithmetic. Numerical accuracy deteriorates with increasing N according to the rate predicted by the condition numbers shown in Figure 5.1. Note however that the conditioning can be significantly improved by clustering of the node points towards the boundary of the computational domain [25, 9]. This is typically not practical in the RBF-FD case, since the stencils are computed based on a local selection of nodes from a global node set. Nor is it typically desired, since we are interested in the accuracy near the center of each stencil, and do not want to trade that against better accuracy near its edges. However, for global or partition differentiation matrices node clustering can be of interest.

When we have access to accurately computed differentiation matrices for small values of ε, we can also study the convergence properties of RBF approximation in this regime. Figure 5.3 illustrates the algebraic convergence of a fixed size RBF-FD stencil when the node density is increased. The test function used in this experiment was

$$f(x,y,z) = \frac{\cos(6z)1.25 + \cos(5.4y)}{6 + 6(3x-1)^2}.$$

The convergence orders agree with those that we would expect from a well behaved polynomial approximation with the same number of degrees of freedom. That is, if for example we have a second order polynomial approximation, we would expect errors of order h^3 for interpolation, h^2 for first order derivatives, and h for second order derivatives.

Figure 5.4 shows convergence results from different ways of employing RBF-FD. As we can see in the figure, stencils computed using a fixed shape parameter and RBF-QR for evaluation give the best results over the whole range of node densities. Using a scaled shape parameter and RBF-Direct works well for low node densities, but as the node density is increased we run into the saturation error associated with stationary interpolation [1, 4]. The effect of saturation can be reduced by

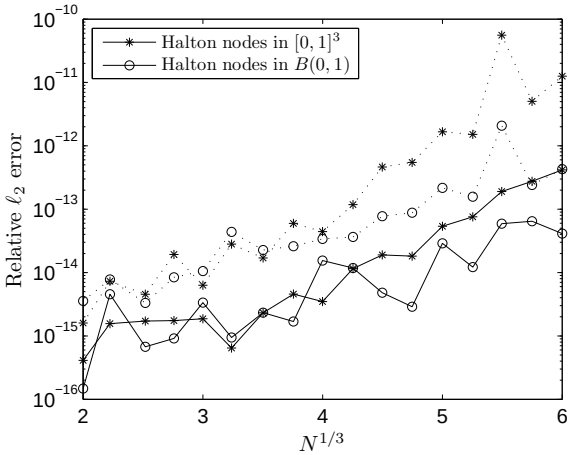

FIG. 5.2. *The errors in computing stencil weights (solid lines) when using the RBF-QR approach in double precision for nodes in the unit cube (∗) and the unit sphere (○). The reference values are computed using RBF-Direct and variable precision arithmetic. The dotted lines show the condition number of A_Ψ multiplied with machine epsilon.*

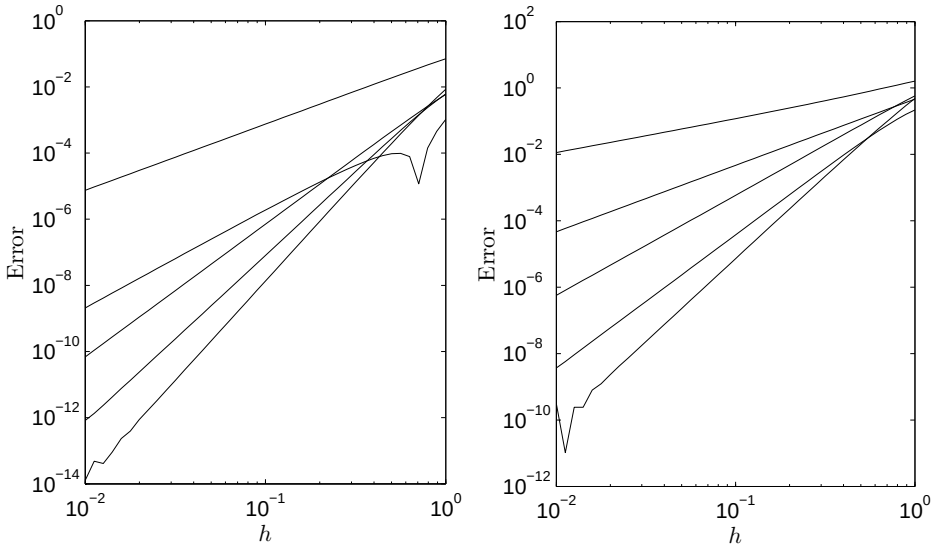

FIG. 5.3. *The error when applying RBF-FD stencils to a test function in 3-D for $\partial/\partial x$ (left) and Δ (right) as a function of the node density. The stencil sizes are $N = 10, 20, 35, 56, 84$ corresponding to from second to sixth order polynomial approximation in the small ε limit. The resulting convergence rates coincide with the expected second to sixth order convergence for the first derivative and first to fifth order convergence for the Laplacian.*

augmenting the stencil with polynomial terms. When adding polynomial terms up to degree three, we recover second order convergence for the Laplacian operator.

Using pure polynomials to generate stencils performs approximately one order of magnitude worse than RBF-FD until we approach the flat RBF limit (note that increasing the node density for a stencil with a fixed number of nodes is tantamount to decreasing the shape parameter for a fixed node density, computationally). However, it should be noted that with RBF-FD, the linear system for the stencil weights is guaranteed to be non-singular for any (pairwise different) node configuration, whereas polynomial approximations may fail.

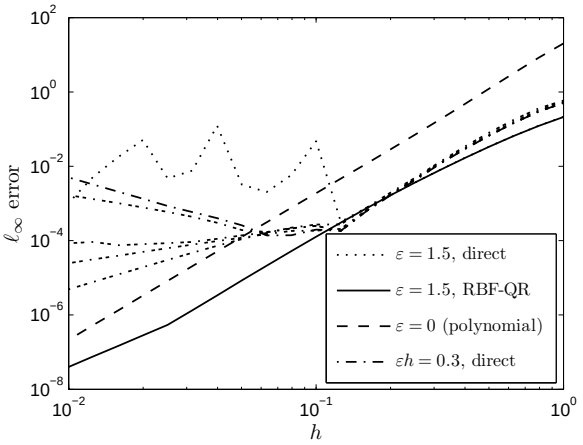

FIG. 5.4. *A comparison of the errors when using RBF-FD stencils with $N = 56$ node points to approximate the Laplacian for a test function. For the RBF-QR method a constant shape parameter $\varepsilon = 1.5$ is used. For RBF-Direct the shape parameter is scaled with the node density and results for stencils augmented by polynomial terms of orders 0 to 3 (top to bottom with a modified linestyle) are also included. RBF-Direct with a fixed shape parameter is included for reference.*

5.2. Computational cost. Figure 5.5 shows the run-time of the RBF-QR code for computing a stencil in 3-D. Included for comparison are RBF-Direct and RBF-Direct with variable precision arithmetic (VPA), here set to 100 digits of accuracy. The computational cost of the RBF-QR algorithm is about 10-20 times higher than that for the direct method. This comparison is merely included to give an idea of the computational cost, as the RBF-Direct method is not numerically accurate for small values of the shape parameter. Variable precision arithmetic, as provided by the Symbolic Math Toolbox, is more than 100 times slower than RBF-QR and does not constitute a computationally feasible solution in a typical application. However, by instead using the Advanpix VPA tools in MATLAB, the time for these computations would be reduced by a factor of about five as seen in [11]. The RBF-QR algorithm is also considerably more efficient in 2-D, where the computational cost for a given N is typically 50–60 percent of the corresponding one in 3-D.

5.3. Pivoting for non-unisolvent node sets. To test the effect of introducing pivoting in the QR-factorization, we consider both the errors in the computed stencil weights and the approximation errors when using the computed stencils. As test function we use

$$u(x, y) = \sin(0.18\pi(x^2 + 2y^2)) - \sin(0.18\pi(2x^2 + (y - 5/3)^2)).$$

The tolerance used to determine when a pivot is small in the RBF-QR algorithm was set to 2 in all experiments. This indicates that we deselect all columns that drop in magnitude by a factor of $100 = 10^2$ or more. When testing the accuracy of the weights we compare with the result from using RBF-Direct with MATLAB's variable precision arithmetic with up to 400 digits. In the first example, we place n points uniformly on the line $x = y$ between $\pm(1/\sqrt{2}, 1/\sqrt{2})$. The results are shown in Figure 5.6. The relative errors in the stencil weights for the first derivative in x grow slowly with n, but are small as long as $\varepsilon < 1$. We do get the true Gaussian limit in all cases. For larger ε the errors in the weights grow faster. There are no significant differences in the approximation errors. Note however, that if it is known that node points are located on a line, it is always better to use the one-dimensional

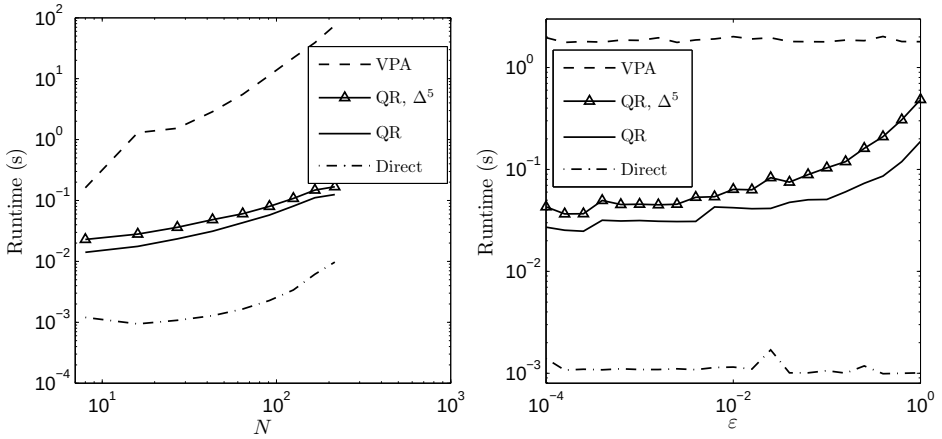

FIG. 5.5. *The run-time of the RBF-QR algorithm as a function of N at $\varepsilon = 10^{-4}$ (left), and as a function of ε at $N = 30$ (right), for computing a $\partial/\partial x$-stencil in 3-D. RBF-Direct and RBF-Direct with variable precision arithmetic (VPA), as provided by the Symbolic Math Toolbox, are shown for comparison. Here, the number of digits for VPA is set to* 100. *The solid line with markers (Δ) correspond to instead computing a tenth order hyper-viscosity stencil. Random nodes were used in the experiment.*

RBF-QR method than to force a higher dimensional version to search for the correct answer by pivoting.

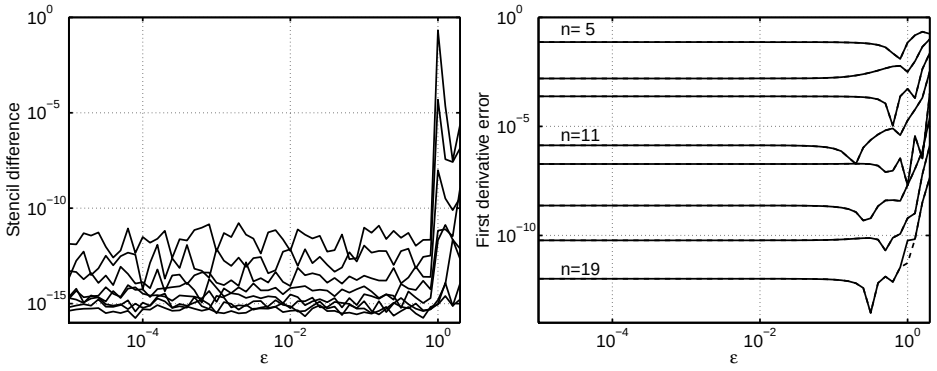

FIG. 5.6. *The relative error in the stencil weights for the first derivative in x (left) and the approximation error for the test function u (right) for the RBF-QR method (solid) and RBF-Direct with variable precision arithmetic (dashed) when using $n = 5, 7, \ldots, 19$ node points uniformly distributed on the line $x = y$ within the unit circle.*

In the second example, $n \times n$ node points are placed on a uniform grid, again scaled to fall within the unit disc. Here, we have computed the weights for the Laplacian. In Figure 5.7, we can see that except for in the $n = 3$ case, the stencil weights gradually approach a limit different from the reference stencil. However, the differences in approximation errors are very small and are sometimes in favor of the RBF-QR limit. To see in what way the limits differ, we display the RBF-QR limit stencil and the reference limit stencil for $n = 5$ side by side below.

$$\frac{1}{36}\begin{pmatrix} -1 & 4 & -30 & 4 & -1 \\ 4 & -16 & 408 & -16 & 4 \\ -30 & 408 & -1476 & 408 & -30 \\ 4 & -16 & 408 & -16 & 4 \\ -1 & 4 & -30 & 4 & -1 \end{pmatrix} \quad \frac{1}{36}\begin{pmatrix} 0 & 0 & -24 & 0 & 0 \\ 0 & 0 & 384 & 0 & 0 \\ -24 & 384 & -1440 & 384 & -24 \\ 0 & 0 & 384 & 0 & 0 \\ 0 & 0 & -24 & 0 & 0 \end{pmatrix}$$

The true Gaussian limit on a grid is always the tensor product stencil, with the

weights forming a cross. In the RBF-QR case we also get non-zero values for the corner nodes. However, the overall size of the stencil weights is very similar.

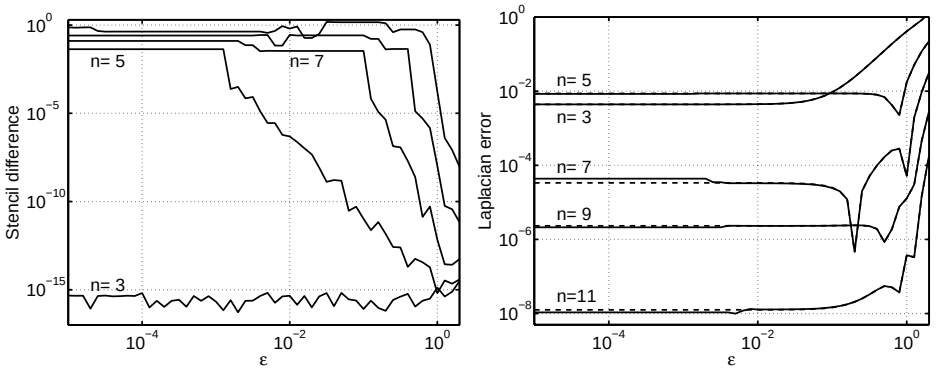

FIG. 5.7. *The relative error in the stencil weights for the Laplacian (left) and the approximation error for the test function u (right) for the RBF-QR method (solid) and RBF-Direct with variable precision arithmetic (dashed) when using $n \times n$ node points on a uniform grid with the corner points falling on the unit circle.*

For comparison, we also tested the RBF-QR algorithm with pivoting on stencils with Halton nodes (unstructured). Also in this case, the accuracy of the weights decrease slowly with n from $1.8 \cdot 10^{-15}$ at 3×3 points to $2.9 \cdot 10^{-11}$ at 11×11 points. However, there is no apparent dependence on the shape parameter and the answer agrees with the reference in all cases. Regarding the computational cost, it is expected that pivoting generates additional work. Comparing the cost for computing stencils for $n \times n$ uniform nodes (pivoting needed) and the same number of Halton nodes (no pivoting) resulted in approximately 1.5 times higher cost for uniform nodes, for the stencil sizes used here.

5.4. Application example. The RBF-FD differentiation matrices described in Section 3.2 can easily be utilized for solving boundary value problems. As a basic test problem, we choose the 2D Poisson equation with Dirichlet boundary conditions

$$-\Delta u = f \quad \text{in} \quad \Omega, \tag{5.1}$$

$$u = g \quad \text{on} \quad \partial\Omega, \tag{5.2}$$

where $\partial\Omega$ (see the left part of Figure 5.8) is a starfish like shape with parametric equation

$$r_b(\theta) = 0.8 + 0.1(\sin(6\theta) + \sin(3\theta)), \quad \theta \in [0, 2\pi). \tag{5.3}$$

The collocation process involves three steps. First, we discretize the domain with $N = N_i + N_b$ points, where N_i is the number of interior points and N_b is the number of boundary points. Then, the Poisson equation is collocated at the interior points using RBF-FD stencils for approximating the Laplacian. This results in an under-determined system of size $N_i \times N$. By augmenting the system with N_b equations from the boundary condition we end up with an $N \times N$ linear system of equations. Finally, we solve the linear system to obtain the nodal solution values. As in the finite difference case, the system matrix is sparse. The right part of Figure 5.8 shows an example of the sparsity distribution of a system matrix with $N = 363$ and stencil size $n_{\text{loc}} = 21$ after a minimal degree reordering.

The process of distributing RBF points uniformly can for example be carried out by treating nodes as being connected by springs that repel and attract one another

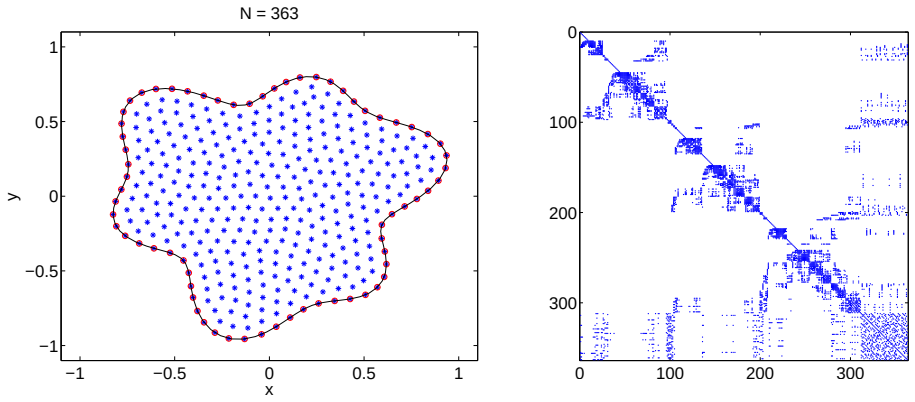

FIG. 5.8. *An example of starfish like domain discretized with $N = 363$ uniformly distributed nodes (left) and the sparsity distribution of the system matrix with stencil size $n_{loc} = 21$ (right).*

until an equilibrium state is achieved [22]. The forcing function and boundary condition of (5.1) are chosen such that the exact solutions are the following smooth functions

$$u_1(x, y) = \sin(\pi x)\sin(\pi y), \qquad \text{(Test case 1)}$$
$$u_2(x, y) = (x^2 + y^2 - 0.25)^2, \qquad \text{(Test case 2)}$$

Figure 5.9 shows numerical experiments for the two test cases using the nodes displayed in Figure 5.8, with the total number of points $N = 363$ kept fixed. The accuracy of the numerical solutions of (5.1) with respect to the stencil sizes (n_{loc}) for $\varepsilon = 0.1$ can be seen in the two leftmost subfigures.

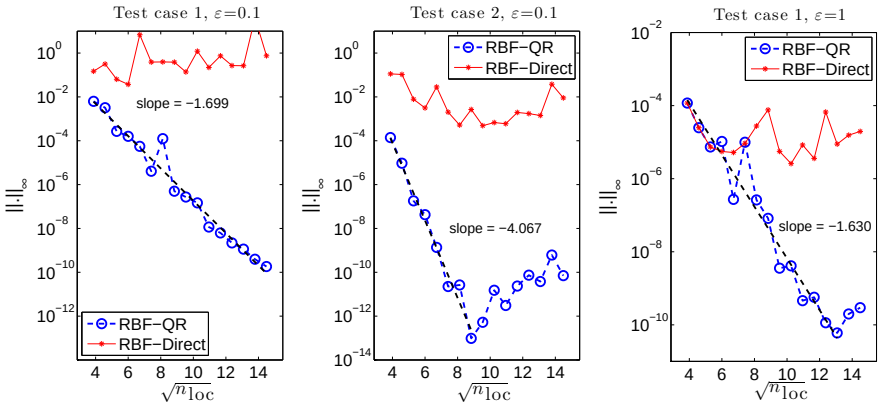

FIG. 5.9. *Comparisons of convergence trends for RBF-QR and RBF-Direct with respect to the square root of the stencil sizes $\sqrt{n_{\text{loc}}}$ when solving the Poisson equation.*

For RBF-QR the convergence trends in both cases are spectral of the form

$$\|\cdot\|_\infty \propto \exp(-s\sqrt{n_{\text{loc}}})$$

with slopes s around 1.7 for Test case 1 and 4.1 for Test case 2. The results are compared with using RBF-Direct (in double precision). As can be seen in the figures, RBF-Direct does not converge due to severe ill-conditioning. The rightmost subfigure shows the result for Test case 1 with the larger shape parameter value $\varepsilon = 1$. In that case, RBF-Direct agrees with the results obtained with RBF-QR for smaller stencil sizes before leveling off again due to the conditioning issue.

6. Summary. In this work, we have derived and implemented the necessary expressions for computing differentiation matrices and stencil weights based on the numerically stable RBF-QR formulation of RBF approximation, which allows computation of interpolants and derivative approximations for small values of the shape parameter. We have also considered practical implementation aspects of the algorithm, such as introducing a pivoting strategy to handle non-unisolvent node sets and how to implement the algorithm efficiently to render the computational cost competitive. Special care has been taken to implement hyperviscosity operators in an accurate and efficient way, since these are a necessary tool when solving convective PDEs with explicit time-stepping using RBF-FD.

We have showed in numerical experiments that differentiation matrices and stencil weights are accurately computed by the RBF-QR algorithm. The implemented pivoting strategy provides stable results also for non-unisolvent node sets, but does not always reproduce the exact Gaussian limit. However, from the approximation point of view, the results are equally good.

We also looked at the accuracy when RBF-FD stencils are applied to test functions. With RBF-QR, we get the best possible accuracy for all tested node densities without the need to augment the approximation with polynomial terms. A stationary approach combined with RBF-Direct performs reasonably well, but the convergence rate for high node densities is determined by the degree of the added polynomial terms.

The computational cost for the RBF-QR method is around 10–20 times higher than it would be for the RBF-Direct method if that was possible to use. This is a reasonable penalty for the increased range of problems that can be solved (accurately). It should be noted that the ill-conditioning of RBF-Direct grows both with decreasing shape parameters and increasing node numbers. Hence, we can also solve larger problems with RBF-QR.

The RBF-QR method or another stable evaluation method is required in order to achieve the high order convergence rates the RBF approximations are capable of. In the elliptic PDE example we can see how the RBF-QR results converge almost down to machine precision while RBF-Direct fails to produce accurate results altogether in the small shape parameter regime and levels off quite early also for larger shape parameter values.

Acknowledgements. The authors would like to thank Cécile Piret for sharing an implementation of derivatives of spherical harmonics, and Grady Wright and Edward Fuselier for interesting discussions on RBF differentiation matrices. Furthermore, we thank an anonymous reviewer for alerting us to the complexities of the issue of combining the RBF-QR method with RBF interpolants augmented by polynomials.

Appendix A. Derivative formulas. Consider a generic expansion function of the type

$$V(\underline{x}) = f(r) \cdot \begin{cases} g(\theta) & \text{in 2D} \\ Y(\theta,\varphi) & \text{in 3D} \end{cases}, \tag{A.1}$$

where in both cases $f(r) = T_{j,m}(r) = e^{-\varepsilon^2 r^2} r^{2m} T_{j-2m}(r)$ and $T_n(r)$ is the n:th Chebyshev polynomial of the first kind. The angular part is given by

$$g(\theta) = \begin{cases} \cos((2m+p)\theta), \\ \sin((2m+p)\theta), \quad 2m+p \neq 0, \end{cases} \tag{A.2}$$

$$Y(\theta,\varphi) = Y_\mu^\nu(\theta,\varphi), \qquad \mu = 2m+p,\ \nu = -\mu,\dots,\mu, \tag{A.3}$$

where Y_μ^ν denote the real spherical harmonics and p is the parity of j. Furthermore, the Chebyshev polynomials and their derivatives can be computed by

$$T_n(r) = \cos(n \arccos(r)), \tag{A.4}$$

$$T_n'(r) = nU_{n-1}(r) = n\frac{\sin(n \arccos(r))}{\sin(\arccos(r))}, \tag{A.5}$$

$$T_n''(r) = \frac{n^2 T_n(r) - rnU_{n-1}(r)}{r^2 - 1}, \tag{A.6}$$

where

$$\lim_{r\to 1} T_n'(r) = n^2, \quad \lim_{r\to 1} T_n''(r) = \frac{1}{3}n^2(n^2-1).$$

A.1. Low order derivatives.

A.1.1. 2D. The polar coordinate system is in 2D given by $r \in [0,\infty)$, $\theta \in [0,\pi)$, and the first derivatives in Cartesian coordinates $\underline{x} = (x, y)$ are

$$\frac{\partial}{\partial x} = \cos\theta\frac{\partial}{\partial r} - \sin\theta\frac{1}{r}\frac{\partial}{\partial\theta}, \tag{A.7}$$

$$\frac{\partial}{\partial y} = \sin\theta\frac{\partial}{\partial r} + \cos\theta\frac{1}{r}\frac{\partial}{\partial\theta}. \tag{A.8}$$

For the trigonometric functions $g(\theta)$ we have

$$g'(\theta) = (2m+p)h(\theta), \tag{A.9}$$

$$g''(\theta) = -(2m+p)^2 g(\theta), \tag{A.10}$$

where $h(\theta)$ corresponds to $g(\theta)$ with the two functions in reversed order. Thus for a generic expansion function we arrive at

$$\frac{\partial V}{\partial x} = f'(r)g(\theta)\cos\theta - (2m+p)\frac{f(r)}{r}h(\theta)\sin\theta, \tag{A.11}$$

$$\frac{\partial V}{\partial y} = f'(r)g(\theta)\sin\theta + (2m+p)\frac{f(r)}{r}h(\theta)\cos\theta. \tag{A.12}$$

We proceed in a similar way with the second order derivatives. By applying the chain rule and collecting terms with the same angular dependence, we end up with

$$\begin{aligned}\frac{\partial^2 V}{\partial x^2} &= f''(r)g(\theta)\cos^2\theta + 2(2m+p)\left(\frac{f(r)}{r^2} - \frac{f'(r)}{r}\right)h(\theta)\sin\theta\cos\theta \\ &\quad + \left(\frac{f'(r)}{r} - (2m+p)^2\frac{f(r)}{r^2}\right)g(\theta)\sin^2\theta,\end{aligned} \tag{A.13}$$

$$\begin{aligned}\frac{\partial^2 V}{\partial y^2} &= f''(r)g(\theta)\sin^2\theta - 2(2m+p)\left(\frac{f(r)}{r^2} - \frac{f'(r)}{r}\right)h(\theta)\sin\theta\cos\theta \\ &\quad + \left(\frac{f'(r)}{r} - (2m+p)^2\frac{f(r)}{r^2}\right)g(\theta)\cos^2\theta,\end{aligned} \tag{A.14}$$

$$\begin{aligned}\frac{\partial^2 V}{\partial x\partial y} &= \left(f''(r) - \frac{f'(r)}{r} + (2m+p)^2\frac{f(r)}{r^2}\right)g(\theta)\sin\theta\cos\theta \\ &\quad - (2m+p)\left(\frac{f(r)}{r^2} - \frac{f'(r)}{r}\right)\left(\cos^2\theta - \sin^2\theta\right)h(\theta).\end{aligned} \tag{A.15}$$

The trigonometric components of the derivatives can be evaluated directly, but each radial function needs to be handled with care. We need to expand and investigate

the following five radial functions

$$F(r) = (2m+p)\frac{f(r)}{r}, \tag{A.16}$$

$$G(r) = f'(r), \tag{A.17}$$

$$Q(r) = f''(r), \tag{A.18}$$

$$R(r) = (2m+p)\left(\frac{f(r)}{r^2} - \frac{f'(r)}{r}\right), \tag{A.19}$$

$$S(r) = \left(\frac{f'(r)}{r} - (2m+p)^2\frac{f(r)}{r^2}\right). \tag{A.20}$$

We will divide the results into three different cases.

The case $m > 0$:.

$$F(r) = (2m+p)e^{-\varepsilon^2 r^2} r^{2m-1} T_{j-2m}(r), \tag{A.21}$$

$$G(r) = e^{-\varepsilon^2 r^2}\left\{2(-\varepsilon^2 r^{2m+1} + mr^{2m-1})T_{j-2m}(r) + r^{2m}T'_{j-2m}(r)\right\}, \tag{A.22}$$

$$\begin{aligned} Q(r) = e^{-\varepsilon^2 r^2}\big\{&\left(2\varepsilon^2(2\varepsilon^2 r^2 - 1 - 4m)r^{2m} + 2m(2m-1)r^{2m-2}\right)T_{j-2m}(r) \\ &- 4(\varepsilon^2 r^{2m+1} - mr^{2m-1})T'_{j-2m}(r) + r^{2m}T''_{j-2m}(r)\big\}, \end{aligned} \tag{A.23}$$

$$\begin{aligned} R(r) = (2m+p)e^{-\varepsilon^2 r^2}\big\{&\left(r^{2m-2}(1-2m) + 2\varepsilon^2 r^{2m}\right)T_{j-2m}(r) \\ &- r^{2m-1}T'_{j-2m}(r)\big\}, \end{aligned} \tag{A.24}$$

$$\begin{aligned} S(r) = e^{-\varepsilon^2 r^2}\big\{&(-2\varepsilon^2 r^{2m} + (2m - (2m+p)^2)r^{2m-2})T_{j-2m}(r) \\ &+ r^{2m-1}T'_{j-2m}(r)\big\}. \end{aligned} \tag{A.25}$$

The case $m = p = 0$:.

$$F(r) \equiv 0, \tag{A.26}$$

$$G(r) = e^{-\varepsilon^2 r^2}\left\{-2\varepsilon^2 r T_j(r) + T'_j(r)\right\}, \tag{A.27}$$

$$Q(r) = e^{-\varepsilon^2 r^2}\left\{2\varepsilon^2(2\varepsilon^2 r^2 - 1)T_j(r) - 4\varepsilon^2 r T'_j(r) + T''_j(r)\right\}, \tag{A.28}$$

$$R(r) \equiv 0, \tag{A.29}$$

$$S(r) = e^{-\varepsilon^2 r^2}\left\{-2\varepsilon^2 T_j(r) + \frac{T'_j(r)}{r}\right\}, \tag{A.30}$$

The function $S(r)$ cannot be evaluated at $r = 0$, where we have the limit

$$\lim_{r\to 0} S(r) = (2\varepsilon^2 + j^2)(-1)^{j/2-1}.$$

The case $m = 0$, $p = 1$:.

$$F(r) = e^{-\varepsilon^2 r^2}\frac{T_j(r)}{r}, \tag{A.31}$$

$$G(r) = e^{-\varepsilon^2 r^2}\left\{-2\varepsilon^2 r T_j(r) + T'_j(r)\right\}, \tag{A.32}$$

$$Q(r) = e^{-\varepsilon^2 r^2}\left\{2\varepsilon^2(2\varepsilon^2 r^2 - 1)T_j(r) - 4\varepsilon^2 r T'_j(r) + T''_j(r)\right\}, \tag{A.33}$$

$$R(r) = e^{-\varepsilon^2 r^2}\left\{2\varepsilon^2 T_j(r) + \frac{T_j(r)}{r^2} - \frac{T'_j(r)}{r}\right\}, \tag{A.34}$$

$$S(r) = -R(r). \tag{A.35}$$

The limits are in this case

$$\lim_{r\to 0} F(r) = j(-1)^{(j-1)/2}, \quad \lim_{r\to 0} R(r) = \lim_{r\to 0} S(r) = 0.$$

A.1.2. 3D. The spherical coordinate system in 3D is given by $r \in [0, \infty)$, $\theta \in [0, \pi]$ (colatitude) and $\varphi \in [-\pi, \pi]$, and the first derivatives in Cartesian coordinates $\underline{x} = (x, y, z)$ are

$$\frac{\partial}{\partial x} = \cos\varphi \sin\theta \frac{\partial}{\partial r} - \frac{\sin\varphi}{r\sin\theta}\frac{\partial}{\partial \varphi} + \frac{\cos\varphi\cos\theta}{r}\frac{\partial}{\partial\theta}, \tag{A.36}$$

$$\frac{\partial}{\partial y} = \sin\varphi \sin\theta \frac{\partial}{\partial r} + \frac{\cos\varphi}{r\sin\theta}\frac{\partial}{\partial \varphi} + \frac{\sin\varphi\cos\theta}{r}\frac{\partial}{\partial\theta}, \tag{A.37}$$

$$\frac{\partial}{\partial z} = \cos\theta \frac{\partial}{\partial r} - \frac{\sin\theta}{r}\frac{\partial}{\partial\theta}. \tag{A.38}$$

The real spherical harmonics are given by

$$Y_\mu^\nu(\theta,\varphi) = P_\mu^\nu(\cos\theta)g(\varphi) = \begin{cases} P_\mu^\nu(\cos\theta)\cos(\nu\varphi), & \nu = 0, \cdots, \mu \\ P_\mu^{-\nu}(\cos\theta)\sin(-\nu\varphi), & \nu = -\mu, \cdots, -1 \end{cases} \tag{A.39}$$

where $P_\mu^\nu(x)$ are the normalized associated Legendre polynomials. Unless otherwise noted, P_μ^ν will denote $P_\mu^\nu(\cos\theta)$. The angular derivatives of Y_μ^ν are

$$\frac{\partial Y_\mu^\nu}{\partial\varphi} = -\nu Y_\mu^{-\nu}, \tag{A.40}$$

$$\frac{\partial Y_\mu^\nu}{\partial\theta} = \frac{1}{2}g(\varphi)\left(\sqrt{\mu^2+\mu+\nu^2-|\nu|}P_\mu^{\nu+1} - \sqrt{\mu^2+\mu+\nu^2+|\nu|}P_\mu^{\nu-1}\right), \tag{A.41}$$

where we have two special cases given by

$$P_\mu^\nu = \begin{cases} 0, & |\nu| > \mu \\ -P_\mu^{-\nu}, & \nu = -1 \end{cases}. \tag{A.42}$$

Differentiation of a generic expansion function thus yields

$$\frac{\partial V}{\partial x} = f'\cos\varphi\sin\theta Y_\mu^\nu - \frac{f}{r}\frac{\sin\varphi}{\sin\theta}\frac{\partial Y_\mu^\nu}{\partial\varphi} + \frac{f}{r}\cos\varphi\cos\theta\frac{\partial Y_\mu^\nu}{\partial\theta}, \tag{A.43}$$

$$\frac{\partial V}{\partial y} = f'\sin\varphi\sin\theta Y_\mu^\nu + \frac{f}{r}\frac{\cos\varphi}{\sin\theta}\frac{\partial Y_\mu^\nu}{\partial\varphi} + \frac{f}{r}\sin\varphi\cos\theta\frac{\partial Y_\mu^\nu}{\partial\theta}, \tag{A.44}$$

$$\frac{\partial V}{\partial z} = f'\cos\theta Y_\mu^\nu - \frac{f}{r}\sin\theta\frac{\partial Y_\mu^\nu}{\partial\theta}. \tag{A.45}$$

For $\nu > 0$, the factor $\frac{P_\mu^\nu(\cos\theta)}{\sin\theta}$ appearing in the x- and y-derivatives cannot be evaluated directly at $\theta = 0$. It is harmless however, which is easily seen if we let

$$M_\mu^\nu(x) = (-1)^\nu\frac{\partial^\nu}{\partial x^\nu}P_\mu(x), \tag{A.46}$$

where $P_\mu(x)$ denotes the ordinary Legendre polynomials. Then from the definition of the associated Legendre polynomials, we have

$$P_\mu^\nu(x) = (1-x^2)^{\frac{\nu}{2}}M_\mu^\nu(x) \tag{A.47}$$

and thus

$$\frac{P_\mu^\nu(\cos\theta)}{\sin\theta} = (\sin\theta)^{\nu-1}M_\mu^\nu(\cos\theta). \tag{A.48}$$

We again look at three cases for the radial part. The case $m > 0$ is the same as for 2D, and thus

$$\frac{f(r)}{r} = e^{-\varepsilon^2r^2}r^{2m-1}T_{j-2m}(r), \tag{A.49}$$

$$f'(r) = e^{-\varepsilon^2r^2}\left\{2(-\varepsilon^2r^{2m+1} + mr^{2m-1})T_{j-2m}(r) + r^{2m}T'_{j-2m}(r)\right\}. \tag{A.50}$$

For $m = p = 0$ and thus $\mu = \nu = 0$, both partial derivatives of Y_μ^ν are identically zero. With odd j, we instead have $p = 1$ and $\mu = 1$. Here,

$$\frac{f}{r} = e^{-\varepsilon^2 r^2} \frac{T_j(r)}{r}, \tag{A.51}$$

$$f' = e^{-\varepsilon^2 r^2} \left\{ T_j'(r) - 2\varepsilon^2 r T_j(r) \right\}, \tag{A.52}$$

and we have the limits

$$\lim_{r\to 0} \frac{f}{r} = \lim_{r\to 0} f' = j(-1)^{(j-1)/2}.$$

A.2. Powers of the Laplacian. The formulas for the Laplace operator are very neat in both 2D and 3D, since the angular parts are eigenfunctions of the operator. Application of the Laplace operator to a generic expansion function gives

$$\Delta V = \begin{cases} \left(f'' + \frac{f'}{r} - (2m+p)^2 \frac{f}{r^2} \right) g(\theta), & \text{in 2D,} \\ \left(f'' + \frac{2f'}{r} - \mu(\mu+1) \frac{f}{r^2} \right) Y_\mu^\nu, & \text{in 3D.} \end{cases} \tag{A.53}$$

The radial part is non-singular, since it is given by $Q(r) + S(r)$ in 2D and we have

$$\frac{f}{r^2} = e^{-\varepsilon^2 r^2} r^{2m-2} T_{j-2m}(r), \tag{A.54}$$

$$\frac{f'}{r} = e^{-\varepsilon^2 r^2} \left\{ 2(m r^{2m-2} - \varepsilon^2 r^{2m}) T_{j-2m}(r) + r^{2m-1} T_{j-2m}'(r) \right\}, \tag{A.55}$$

$$\begin{aligned} f'' = e^{-\varepsilon^2 r^2} \big\{ & 2 \left(2\varepsilon^4 r^4 - (4m+1)\varepsilon^2 r^2 + m(2m-1) \right) r^{2m-2} T_{j-2m}(r) \\ & + 4 \left(m - \varepsilon^2 r^2 \right) r^{2m-1} T_{j-2m}'(r) + r^{2m} T_{j-2m}''(r) \big\}, \end{aligned} \tag{A.56}$$

and the limits

$$\lim_{r\to 0} \frac{f'}{r} = (2\varepsilon^2 + j^2)(-1)^{j/2+1}, \qquad m = p = 0,$$

$$\lim_{r\to 0} \left(\frac{f'}{r} - \frac{f}{r^2} \right) = 0, \qquad m = 0,\ p = 1.$$

in 3D. To calculate the hyper-viscosity operator Δ^k, we write the radial part as

$$f^k(r) = \sum_{l=0}^{2k+1} \varepsilon^{2l} p_{l,k}(r) e^{-\varepsilon^2 r^2}, \tag{A.57}$$

where $p_{l,k}$ is a polynomial of order at most $j + 2k$, and starting from $f^0(r) = f(r)$, we apply the Laplace operator recursively. In the 3D case, we obtain

$$\begin{aligned} f^{k+1} = \Delta f^k = \sum_{l=0}^{2k+1} \varepsilon^{2l} \big\{ & p_{l,k}'' + \frac{2p_{l,k}'}{r} - \mu(\mu+1) \frac{p_{l,k}}{r^2} \\ & - 2\varepsilon^2 (2r p_{l,k}' + 3p_{l,k}) + 4\varepsilon^4 r^2 p_{l,k} \big\} e^{-\varepsilon^2 r^2} \end{aligned} \tag{A.58}$$

and thus

$$p_{l,k+1} = \begin{cases} q_{l,k}^1, & l = 0, \\ q_{l,k}^1 + q_{l,k}^2, & l = 1, \\ q_{l,k}^1 + q_{l,k}^2 + q_{l,k}^3, & l = 2, \dots, 2k+1, \\ q_{l,k}^2 + q_{l,k}^3, & l = 2k+2, \\ q_{l,k}^3, & l = 2k+3, \end{cases} \tag{A.59}$$

where

$$q^1_{l,k} = p''_{l,k} + \frac{2p'_{l,k}}{r} - \mu(\mu+1)\frac{p_{l,k}}{r^2}, \tag{A.60}$$

$$q^2_{l,k} = -2(2rp'_{l-1,k} + 3p_{l-1,k}), \tag{A.61}$$

$$q^3_{l,k} = 4r^2 p_{l-2,k}. \tag{A.62}$$

The derivation is similar in 2D, with the polynomials $q_{l,k}$ now given by

$$q^1_{l,k} = p''_{l,k} + \frac{p'_{l,k}}{r} - (2m+p)^2\frac{p_{l,k}}{r^2}, \tag{A.63}$$

$$q^2_{l,k} = -4(rp'_{l-1,k} + p_{l-1,k}), \tag{A.64}$$

$$q^3_{l,k} = 4r^2 p_{l-2,k}. \tag{A.65}$$

Note that the polynomial coefficients for given j, m, k can be precomputed and stored. Rewriting the polynomials as Chebyshev series and using Clenshaw's algorithm to evaluate them appears to give very high accuracy for $j < 60$.

When computing hyper-viscosity for stencils, it is possible to evaluate at $r = 0$ with no loss of generality, which simplifies the expressions substantially. Only basis functions with even j and $m = 0$ are non-zero at $r = 0$ and with $\Delta^k f(r) = f^k(r)$ as before, we obtain in 3-D

$$f^k(0) = (-1)^k(2k+1)\sum_{l=0}^{k}\frac{(2k)!}{(2l)!(k-l)!}\varepsilon^{2(k-l)}\prod_{i=1}^{l}(-1)^{j/2}(j^2 - 4(i-1)^2) \tag{A.66}$$

for these basis functions. For $m = 0$, we have $\mu = \nu = 0$ and thus

$$\Delta^k V|_{r=0} = \begin{cases} f^k(0)Y^0_0, & m = 0, \quad j = 0, 2, 4, 6, \ldots \\ 0, & \text{otherwise.} \end{cases} \tag{A.67}$$

REFERENCES

[1] M. D. Buhmann, *Radial basis functions: theory and implementations*, vol. 12 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2003.

[2] T. Cecil, J. Qian, and S. Osher, *Numerical methods for high dimensional Hamilton-Jacobi equations using radial basis functions*, J. Comput. Phys., 196 (2004), pp. 327–347.

[3] O. Davydov and D. T. Oanh, *Adaptive meshless centres and RBF stencils for Poisson equation*, J. Comput. Phys., 230 (2011), pp. 287–304.

[4] G. E. Fasshauer, *Meshfree approximation methods with MATLAB*, vol. 6 of Interdisciplinary Mathematical Sciences, World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2007.

[5] A. J. M. Ferreira, G. E. Fasshauer, R. C. Batra, and J. D. Rodrigues, *Static deformations and vibration analysis of composite and sandwich plates using a layerwise theory and RBF–PS discretizations with optimal shape parameter*, Composite structures, 86 (2008), pp. 328–343.

[6] N. Flyer and E. Lehto, *Rotational transport on a sphere: local node refinement with radial basis functions*, J. Comput. Phys., 229 (2010), pp. 1954–1969.

[7] N. Flyer, E. Lehto, S. Blaise, G. B. Wright, and A. St-Cyr, *A guide to RBF-generated finite differences for nonlinear transport: Shallow water simulations on a sphere*, J. Comput. Phys., 231 (2012), pp. 4078–4095.

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

[9] B. Fornberg, E. Larsson, and N. Flyer, *Stable computations with Gaussian radial basis functions*, SIAM J. Sci. Comput., 33 (2011), pp. 869–892.

[10] B. Fornberg and E. Lehto, *Stabilization of RBF-generated finite difference methods for convective PDEs*, J. Comput. Phys., 230 (2011), pp. 2270–2285.

[11] B. Fornberg, E. Lehto, and C. Powell, *Stable calculation of Gaussian-based RBF-FD stencils*, Comput. Math. Appl., 65 (2013), pp. 627–637.

[12] 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.
[13] B. Fornberg and G. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter*, Comput. Math. Appl., 48 (2004), pp. 853–867.
[14] B. Fornberg, G. Wright, and E. Larsson, *Some observations regarding interpolants in the limit of flat radial basis functions*, Comput. Math. Appl., 47 (2004), pp. 37–55.
[15] B. Fornberg and J. Zuev, *The Runge phenomenon and spatially variable shape parameters in RBF interpolation*, Comput. Math. Appl., 54 (2007), pp. 379–398.
[16] E. J. Fuselier and G. B. Wright, *Order-preserving approximations of derivatives with radial basis functions*. To be submitted (2012).
[17] ———, *A high-order kernel method for diffusion and reaction-diffusion equations on surfaces*, J. Sci. Comput., (2013). In press.
[18] E. Larsson and B. Fornberg, *A numerical study of some radial basis function based solution methods for elliptic PDEs*, Comput. Math. Appl., 46 (2003), pp. 891–902.
[19] ———, *Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions*, Comput. Math. Appl., 49 (2005), pp. 103–130.
[20] E. Larsson and A. Heryudono, *A partition of unity radial basis function collocation method for partial differential equations*. Manuscript in preparation.
[21] Y. J. Lee, G. J. Yoon, and J. Yoon, *Convergence of increasingly flat radial basis interpolants to polynomial interpolants*, SIAM J. Math. Anal., 39 (2007), pp. 537–553.
[22] P.-O. Persson and G. Strang, *A simple mesh generator in Matlab*, SIAM Rev., 46 (2004), pp. 329–345 (electronic).
[23] U. Pettersson, E. Larsson, G. Marcusson, and J. Persson, *Improved radial basis function methods for multi-dimensional option pricing*, J. Comput. Appl. Math., 222 (2008), pp. 82–93.
[24] R. B. Platte and T. A. Driscoll, *Eigenvalue stability of radial basis function discretizations for time-dependent problems*, Comput. Math. Appl., 51 (2006), pp. 1251–1268.
[25] R. B. Platte, L. N. Trefethen, and A. B. J. Kuijlaars, *Impossibility of fast stable approximation of analytic functions from equispaced samples*, SIAM Rev., 53 (2011), pp. 308–318.
[26] B. Šarler and R. Vertnik, *Meshfree explicit local radial basis function collocation method for diffusion problems*, Comput. Math. Appl., 51 (2006), pp. 1269–1282.
[27] R. Schaback, *Multivariate interpolation by polynomials and radial basis functions*, Constr. Approx., 21 (2005), pp. 293–317.
[28] 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. Eng., 192 (2003), pp. 941–954.
[29] A. I. Tolstykh, *On using RBF-based differencing formulas for unstructured and mixed structured-unstructured grid calculations*, in Proceedings of the 16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Lausanne, Switzerland, 2000, p. 6 pp.
[30] A. I. Tolstykh and D. A. Shirobokov, *On using radial basis functions in a "finite difference mode" with applications to elasticity problems*, Comput. Mech., 33 (2003), pp. 68–79.
[31] H. Wendland, *Scattered data approximation*, vol. 17 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2005.
[32] 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.