Transport schemes in spherical geometries using spline-based RBF-FD with polynomials

David Gunderman[1], Natasha Flyer *[2], and Bengt Fornberg[1]

[2]National Center for Atmospheric Research, Boulder, Colorado, USA
[1]University of Colorado, Boulder, USA

August 29, 2019

Abstract

This work presents a numerical algorithm for using radial basis function-generated finite differences (RBF-FD) to solve partial differential equations (PDEs) on $\mathbb{S}^2$ using polyharmonic splines with added polynomials defined in a 2D plane (PHS+Poly). We introduce a novel method for calculating RBF-FD PHS+Poly differentiation weights on $\mathbb{S}^2$ using first a Householder reflection and then a projection onto the tangent plane. The new PHS+Poly RBF-FD method is implemented on two standard test cases: 1) solid body rotation on $\mathbb{S}^2$ and 2) 3D tracer transport within Earth's atmosphere. Compared to existing methods (including those in the RBF literature) at similar resolutions, this approach requires fewer degrees of freedom and is algorithmically much simpler. A MATLAB code to implement the method is included in the Appendix.

1 Introduction

A number of applications arising in fields ranging from geophysics to blood flow involve solving transport PDEs on $\mathbb{S}^2$ or in 3-dimensional spherical domains. Radial basis function-generated finite differences (RBF-FD) generalize regular finite difference approximations from grids to scattered layouts, allowing for geometric flexibility while maintaining the computational benefits of local differentiation weight calculation. As surveyed in [9, 10], RBF-FD has been successfully used for more than a decade to solve PDEs in spherical geometries [3, 7, 23]. However, recent developments in the RBF-FD framework provide advantages over previous implementations which improve the accuracy and flexibility of RBF-FD methods.

A series of recent papers has shown that the strategy of combining polyharmonic spline RBFs (PHS), defined as $\phi(r) = r^m$ for odd m and $\phi(r) = r^m log(r)$ for even m, with high degree polynomials (RBF-FD PHS+Poly) is particularly effective [1, 2, 5, 6]. This approach

*Corresponding author.

avoids the difficulty encountered with infinitely smooth RBFs (such as Gaussians) of selecting a shape parameter and provides an acceptable compromise between accuracy and numerical conditioning. Polynomial reproduction up to a user-specified degree can also be seamlessly guaranteed. Along with these advantages, it has also been shown that the use of RBF-FD PHS+Poly avoids the Runge phenomenon near boundaries associated with finite-difference type schemes [1]. All this is gained while still ensuring RBFs traditional advantage: a guarantee of the existence of a unique interpolant to scattered data [4]. However, application of RBF-FD PHS+Poly to PDEs on $\mathbb{S}^2$ requires that the method be modified to avoid linear dependence of the polynomials when expressed in Cartesian coordinates (e.g. on the sphere $x^2 + y^2 + z^2 = 1$).

The method presented here is inspired by a technique discussed in [20], which uses RBF-FD PHS+Poly to obtain high order accuracy in numerical quadrature. The presented method uses a Householder reflection to transform the RBF-FD stencil to a coordinate axis, allowing it to be considered in the tangent plane where the differentiation weights are calculated using PHS plus 2D polynomials, and then the weights are transformed back to the original stencil location.

An overview of the paper is as follows: Section 2 gives a general review of the RBF-FD PHS+Poly method; Section 3 discusses the new RBF-FD PHS+Poly method for spherical geometries; Section 4 discusses how to transform to spherical coordinates; Section 5 presents numerical results on a solid body rotation test case on $\mathbb{S}^2$ and a Hadley-like meridional circulation trace transport test case, comparing these results with other methods from the literature; Section 6 is a short summary.

2 Review of RBF-FD PHS+Poly

The use of RBFs to approximate differentiation operators discretely in multiple dimensions has a long history, as surveyed in [4, 9]. We provide a brief review of the recent history and developments here for background purposes. The earliest applications of the method used global RBF approximations, in which finding differentiation weights involves solving a dense $N \times N$ matrix problem, with each time step requiring a full $N \times N$ matrix-vector multiply, where N is the total number of nodes in the domain. In most application areas computational cost becomes prohibitive as N increases. RBF-FD is a local method leading to exceptionally sparse matrices. Each node in the set $\{\underline{x}_k\}_{k=1}^N$, serves as the "center", $\underline{x}_c$, of an RBF-FD stencil, $\{\underline{x}_i\}_{i=1}^n$ defined by the n nearest nodes to $\underline{x}_c$ (as well as including it), where $n \ll N$. The nodes in each stencil are used to approximate the derivative at $\underline{x}_c$. The computational cost for calculating the differentiation weights and evolution for one time step is reduced, respectively, from $O(N^3)$ and $O(N^2)$ in the global RBF case to being $O(n^3 N)$ and $O(nN)$ in the local RBF-FD case. The calculation of RBF-FD weights is a pre-processing step that only needs to be performed once for any fixed N.[1]

It long has been known that including polynomials to PHS RBF basis: 1) is necessary to ensure that the PHS RBF interpolation matrix is nonsingular for any scattered layout of

[1]The local RBF-FD case also requires as pre-processing a nearest-neighbor search, costing $\mathcal{O}(\mathcal{N} \log \mathcal{N})$ operations when using a *kd*-tree algorithm.

distinct nodes and 2) gives polynomial reproduction up to the degree of polynomial included [4]. However, it was not known until [5, 6] that for when using a PHS basis with RBF-FD the order of accuracy in approximating differentiation operators is dictated by the highest-degree polynomial used and not the PHS RBFs. In [1, 2], it was shown that such a formulation also ensures a variety of other benefits especially with regard to suppressing Runge phenomenon at boundaries.

2.1 Calculating RBF-FD PHS+Poly differentiation weights

Assume we want to approximate a linear operator $\mathcal{L}$, e.g. the components of the gradient $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, or $\frac{\partial}{\partial z}$, at $\underline{x}_c$ by a linear combination of weighted function values at the nearest n nodes which form the stencil, i.e.

$$\mathcal{L}u\big|_{\underline{x}_c} \approx \sum_{i=1}^{n} w_i u_i. \tag{1}$$

The RBF-FD PHS+Poly weights w_i above are obtained by solving the following system, where we have added polynomials only up to degree $d = 1$ for illustration; for a detailed explanation see Chapter 5 in [9]

$$\left[\begin{array}{ccc|ccc} \phi(||\underline{x}_1 - \underline{x}_1||) & \dots & \phi(||\underline{x}_1 - \underline{x}_n||) & 1 & x_1 & y_1 \\ \phi(||\underline{x}_2 - \underline{x}_2|| & \dots & \phi(||\underline{x}_2 - \underline{x}_n||) & 1 & x_2 & y_2 \\ \vdots & . & \vdots & \vdots & \vdots & \vdots \\ \phi(||\underline{x}_n - \underline{x}_1|| & \dots & \phi(||\underline{x}_n - \underline{x}_n||) & 1 & x_n & y_n \\ \hline 1 & \dots & 1 & - & - & - \\ x_1 & \dots & x_n & & 0 & \\ y_1 & \dots & y_n & & & \end{array}\right] \left[\begin{array}{c} w_1 \\ w_2 \\ \vdots \\ w_n \\ \hline \beta_1 \\ \beta_2 \\ \beta_3 \end{array}\right] = \left[\begin{array}{c} \mathcal{L}\phi(||\underline{x} - \underline{x}_1||)_{\underline{x}=\underline{x}_c} \\ \mathcal{L}\phi(||\underline{x} - \underline{x}_2||)_{\underline{x}=\underline{x}_c} \\ \vdots \\ \mathcal{L}\phi(||\underline{x} - \underline{x}_n||)_{\underline{x}=\underline{x}_c} \\ \hline - \\ \mathcal{L}1_{\underline{x}=\underline{x}_c} \\ \mathcal{L}x_{\underline{x}=\underline{x}_c} \\ \mathcal{L}y_{\underline{x}=\underline{x}_c} \end{array}\right] \tag{2}$$

Here $||\cdot||$ denotes the standard Euclidean norm. Notice that when adding polynomials we also need to add the constraints $\sum_{i=1}^{n} w_i p_j(x_i) = \mathcal{L}p_j(x)\big|_{\underline{x}_c}, j = 0, 1, ..., l$, where p_j are the l multivariate polynomials up to degree d that are augmented to the weight calculation matrix, in this case $l = 3$. The matrix equation (2) can be written more concisely as

$$\begin{bmatrix} A & P_d \\ P_d^T & 0 \end{bmatrix} \begin{bmatrix} \underline{w} \\ \underline{\beta}_d \end{bmatrix} = \begin{bmatrix} \mathcal{L}\underline{\phi} \\ \mathcal{L}\underline{p}_d \end{bmatrix} \tag{3}$$

Here $\mathcal{L}\underline{p}_d$ is a vector of size $l \times 1$ with entries that contain the operator $\mathcal{L}$ applied to the polynomials up to degree d, evaluated at the stencil center $\underline{x}_c$. P_d is a matrix of size $n \times l$ with entries that contain the polynomials up to degree d, evaluated at each node in the stencil $\{x_i\}_{i=1}^n$. After solving for the weights, the $\{\beta_i\}_{i=1}^l$ are discarded (see Section 5.2.3.3 in [9]). With PHS RBFs, the left-hand-side matrix will be positive definite in the constrained subspace, as long as sufficiently high-degree polynomials are added, as shown in [4].

The user can specify the order of convergence d of the PHS+Poly derivative approximation to the exact operator by choosing to include polynomial terms up through degree d in the weight calculation matrix, independent of the degree m of the PHS (see [5, 6]).

3 Calculating Differentiation Weights on $\mathbb{S}^2$

3.1 Difficulty with RBF-FD PHS+Poly on $\mathbb{S}^2$

Combining PHS RBFs with polynomials is straightforward in planar geometries. A naive implementation of the RBF-FD PHS+Poly algorithm in 3D Cartesian coordinates on nodes which lie on a sphere manifests immediate issues: 1) there is no guarantee that the action of the operator will lie on the sphere, 2) the polynomials are not linearly independent (e.g. $x^2 + y^2 + z^2 = 1$), leading to a singular matrix in (2).

3.2 Other RBF-FD PHS+Poly methods on $\mathbb{S}^2$

Some methods have been proposed to implement RBF-FD PHS+Poly-like methods on $\mathbb{S}^2$. For example, a natural analogy to polynomials in Cartesian space is the spherical harmonics on the surface of the sphere. Replacing the polynomials up through degree d with spherical harmonics up through degree d in the weight calculation matrix (2) was shown in [21] to have similar effects on the order of accuracy as regular RBF-FD PHS+Poly in $\mathbb{R}^3$. Spherical harmonics are simply restrictions of polynomials in $\mathbb{R}^3$ to $\mathbb{S}^2$ and the spherical harmonics form an orthonormal basis for functions on $\mathbb{S}^2$. Thus, it is unsurprising that spherical harmonics can play the same role on $\mathbb{S}^2$ as polynomials do in $\mathbb{R}^3$ for RBF-FD.

Another candidate for implementing an RBF-FD PHS+Poly type method on $\mathbb{S}^2$, presented in [22], is RBF-Least Orthogonal Interpolation (RBF-LOI), in which a different orthogonal polynomial basis is computed for each stencil and this polynomials basis is used analogously to the typical monomials in the RBF-FD PHS+Poly weight-calculation matrix. The method to be represented in the next section requires less than four times as many nodes per stencil for a particular choice of d (e.g., in [22] $n = 169$ are used for 5th degree polynomial reproduction, where we use only $n = 37$) and requires fewer computations per stencil for the same-sized stencil. The following section describes it in detail.

3.3 Novel method: Using Householder reflections on $\mathbb{S}^2$

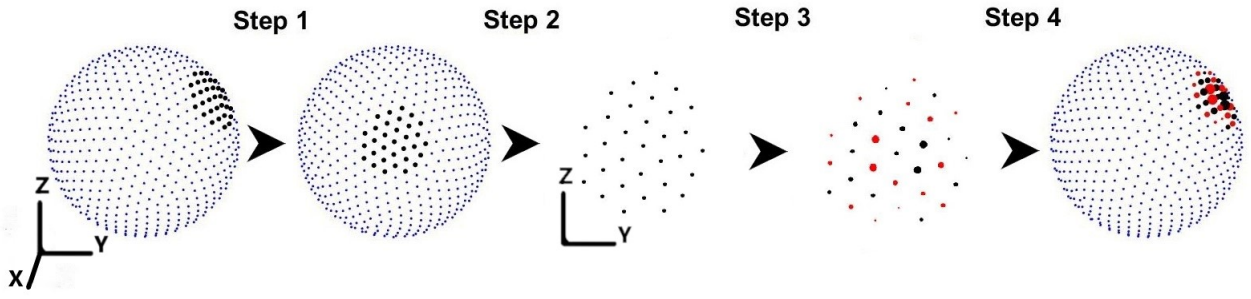

Figure 1: A visualization of the novel method for calculating RBF-FD weights on $\mathbb{S}^2$

Given a node set $\{\underline{x}_k\}_{k=1}^N$ on $\mathbb{S}^2$, we wish to approximate $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, and $\frac{\partial}{\partial z}$ restricted to $\mathbb{S}^2$ with RBF-FD PHS+Poly. The proposed method can be summarized in four steps (visualized

in Figure 1), with the code given in Appendix A:

1. (Step 1) With a kd-tree search, as MATLAB's knnsearch, find the indices of the n nearest nodes, $\{\underline{x}_i\}_{i=1}^n$, to the point $\underline{x}_c$ on the sphere. This defines the k^{th} stencil as shown in Step 1 in Figure 1. This is then performed for all nodes N on the sphere, resulting in the input matrix IDX of size $N \times n$ in the code given in Appendix A.

2. (Step 2) Perform a Householder reflection on each point in the original stencil. This Householder reflection is almost a rotation, but not quite; however, it is unitary. It is defined by a 3x3 matrix Q, such that (x_c, y_c, z_c) becomes the point $(1, 0, 0)$. Q is defined by line 18 in the code in Appendix A. The use of Q to transform all nodes in the original stencil so that they are now located around the x-axis but still on the sphere is done by line 19 in the code in Appendix A.

3. (Step 3) Ignoring the new x-coordinates (they will be very close to one), consider this as a stencil purely in the y,z-plane. This can be done since it is tangent at the origin of the y,z plane and we are only considering first derivatives which involve no curvature. Next, the column vectors of the differentiation weights for the k^{th} stencil, w_y^k and w_z^k, are calculated by 2-D RBF-FD PHS+Poly as is done in (2). This is performed on line 20 in the code given in Appendix A, where the subroutine for calculating RBF-FD PHS+Poly weights is given in Appendix B of the paper [5].

4. (Step 4) However, this has only given us the differentiation weights for $\frac{\partial}{\partial y}$ and $\frac{\partial}{\partial z}$ at $(1, 0, 0)$. What is needed is w_x^k, w_y^k, w_z^k at to the original sphere position for the stencil. It turns out that we get these 3 column vectors by taking the two vectors w_y^k, w_z^k we computed in the previous step and multiply this $n \times 2$ matrix from the right with a 2×3 matrix extracted out of the 3×3 Q matrix. The result is an $n \times 3$ matrix of differentiation weights, w_x^k, w_y^k, w_z^k, to be used at the original stencil location centered at node location (x_c, y_c, z_c). This is done on line 22 in the code in Appendix A.

Notice the above steps are nested in a 'for' loop and repeated for all k nodes on the sphere to create the entire differentiation matrix (DM) that is stored in sparse format at the end of the code in Appendix A.

4 Transforming to spherical coordinates

It is important to note that the algorithm presented above calculates differentiation weights in Cartesian coordinates on the sphere; however, it straightforward to use the method for problems posed in latitude(ϕ)-longitude(λ) spherical coordinates, as will be done in the first test case. An application of the chain rule reveals that

$$\frac{\partial}{\partial \phi} = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y} \tag{4}$$

$$\frac{\partial}{\partial \lambda} = -\frac{zx}{x^2+y^2}\frac{\partial}{\partial x} - \frac{zy}{x^2+y^2}\frac{\partial}{\partial y} + (x^2+y^2)\frac{\partial}{\partial z} \tag{5}$$

In a 3D domain discretized as nested spherical shells, we can approximate $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, and $\frac{\partial}{\partial z}$ at $\underline{x}_c$ by the relations

$$\frac{\partial}{\partial x} = \left(\frac{\partial}{\partial x_s}\right) + \frac{x}{r}\frac{\partial}{\partial r}, \qquad \frac{\partial}{\partial y} = \left(\frac{\partial}{\partial y_s}\right) + \frac{y}{r}\frac{\partial}{\partial r}, \qquad \frac{\partial}{\partial z} = \left(\frac{\partial}{\partial z_s}\right) + \frac{z}{r}\frac{\partial}{\partial r}. \tag{6}$$

where the subscript s denotes restriction to the sphere using the algorithm in Section 3.3. This is the methodology used in the second transport test case. Because it is assured that the derivatives approximated using RBF-FD PHS+Poly have no radial components, it is simple to combine them with any radial discretization scheme. An illustration of the two separate stencils is given in Figure 2, where in the radial direction centered finite differences is used.

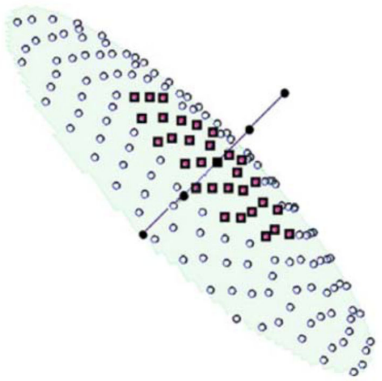

Figure 2: An illustration of the nodes used in the discretization of $\partial/\partial x$, $\partial/\partial y$, $\partial/\partial z$ designed to be accurate at the central black square.

5 Numerical Test Cases

In this section, we investigate the performance of the method by implementing it on two standard test cases from the literature: solid body rotation on $\mathbb{S}^2$ from [28] and tracer transport by Hadley-like meridional circulation in an idealized atmosphere from [15]. In each test case, the field should return to the initial condition after a given time period. Thus, the exact solution at the final time is known, making it straightforward to compute errors.

In both of the test cases, the PDE is given by

$$\frac{\partial q(\underline{x},t)}{\partial t} = -\mathbf{v}(\underline{x},t)\cdot\nabla q(\underline{x},t). \tag{7}$$

$$q(\underline{x},t)_{t=0} = q_0(\underline{x}). \tag{8}$$

The PDE represents the advection of a field $q(\underline{x},t)$ with initial condition $q_0(\underline{x})$ at time $t=0$ by a given velocity field $\mathbf{v}(\underline{x},t)$. In the first test case, $\mathbf{v}$ is constrained to the sphere; in the second case, to the 3D spherical shell defined by the space between Earth's surface, $r=z=0$ and the height $r=z=12000m$. The main focus of the investigation is on the accuracy and convergence of the methods, measured in the relative l_1, l_2, and l_∞ norms. The standard definitions of these norms can be found in [15].

5.1 Node Sets

The node sets used are the minimal energy (ME) and maximal determinant (MD); see the website [29], as well as icosahedral node sets. Results using ME, MD and icosahedral node sets were found to be similar. The derivation and properties of ME, MD, and icosahedral node sets are described in [13, 24, 30]. A nice property of these node sets is that the typical node spacing, h (defined as the mesh norm, see [14, 29]) decays approximately uniformly with the inverse square of the total number of nodes, $h \sim \frac{1}{\sqrt{N}}$. To produce the 3D node sets used in the second test case, we use copies of radial translates of node sets on $\mathbb{S}^2$ such that the nodes have a nested shell-like structure, with N_r nodes per shell and N_s shells.

5.2 Hyperviscosity

It is fairly typical, when approximating convective terms, to obtain differentiation matrices that contain spurious eigenvalues with small positive real parts. This can cause instabilities when time-stepping, especially when an explicit time discretization is used, as is the case in this study. The current remedy for this issue is to add a small artificial diffusion (hyperviscosity) term to the convective operator which effectively shifts spurious eigenvalues of the original differentiation matrix into the negative half eigenplane, so that all eigenvalues fit within the time stability region. This hyperviscosity operator takes the form of Δ^k, where Δ represents the typical spatial Laplacian. This transforms the purely convective equation (7) to

$$\frac{\partial q}{\partial t} = \mathbf{v} \cdot \nabla q + \gamma_N \Delta^k q. \tag{9}$$

Strategies for choosing the parameters k and γ_N in the hyperviscosity formulation are discussed in [7, 11, 21]. In this work, we follow the parameter choice procedure suggested in Appendix A of [7] in order to push eigenvalues into the region of stability for 4th order explicit Runge-Kutta time-stepping. We choose Gaussian type hyperviscosity with $k = 4$; for the MATLAB code see [9], Section 5.3.2.2. Note that neither test case requires hyperviscosity on the time scale called for by the test case. However, to show the robustness of the method, when the cosine bell initial condition in the first test case is rotated for more than tens of rotations, hyperviscosity is required to ensure stability.

5.3 Test Case 1: Solid body rotation on $\mathbb{S}^2$

We consider the standard test case 1 from Williamson, et al. [28]. The test case involves revolving a cos bell on $\mathbb{S}^2$ according to a prescribed velocity field. The zonal and meridional components of the steady velocity field for this test case in spherical coordinates ($-\pi \leq \lambda \leq \pi, -\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$) are respectively given by

$$u(\lambda, \theta) = \sin(\theta)\cos(\lambda)\sin(\alpha) - \cos(\theta)\cos(\alpha), \quad v(\lambda, \theta) = \cos(\lambda)\sin(\alpha) \tag{10}$$

This velocity field (not time-dependent) results in solid body rotation at an angle of α with respect to the polar axis. We choose $\alpha = \pi/2$, corresponding to advection directly over poles. The choice of α is irrelevant to the results since any node can act as the pole. We test two

Method	N	l_1	l_2	l_∞
Finite Volume [27]	9600	$3.5E{-}2$	$2.0E{-}2$	$1.4E{-}2$
RBF-FD PHS+Poly	9604	$2.6E{-}2$	$9.6E{-}3$	$1.0E{-}2$
Spectral Elem. [25]	24576	$5.4E{-}3$	$2.6E{-}3$	$2.3\ E{-}3$
SL RBF-PU (n=84) [23]	23104	Unreported	$4.2E{-}3$	$5.1E{-}3$
RBF-FD PHS+Poly	25600	$1.0E{-}2$	$3.5E{-}3$	$4.6E{-}3$
Discont. Galerkin [18]	38400	$9.2E{-}3$	$3.8E{-}3$	$4.0E{-}3$
RBF-FD PHS+Poly	31250	$5.9E{-}3$	$2.7E{-}3$	$3.6E{-}3$

Table 1: A comparison in the l_1, l_2, and l_∞ norms of various methods in the literature with the present method for stencil size $n = 55$, $\phi(r) = r^3$ and, $d = 5$ (polynomials up to 5th degree in 2D) for one revolution of the cos bell test case with various degrees of freedom, N.

initial conditions: q_1, a compactly supported cosine bell with a jump in the second derivative, and q_2, an infinitely-smooth Gaussian bell to test high-order convergence.

The cosine bell initial condition, $q_1 \in \mathcal{C}^1(\mathbb{S}^2)$, is centered at $(x = 1, y = 0, z = 0)$ and given as [28]:

$$q_1(\underline{x} = x, y, z) = \begin{cases} \frac{1}{2}\left(1 + \cos(\frac{\pi r}{R_b}\right) & r < R_b, \\ 0 & r \geq R_b, \end{cases} \tag{11}$$

where, $r = \arccos(x)$, and the support is set as $R_b = 1/3$. The Gaussian bell initial condition, $q_2 \in \mathcal{C}^\infty(\mathbb{S}^2)$, is also centered at the same place and given by

$$q_2(\underline{x}) = e^{-6r^2} \tag{12}$$

The test calls for one rotation of the bell around the sphere, corresponding to $T = 2\pi$.

5.3.1 Numerical results

We use the RBF-FD PHS+Poly method described in Section 3.3 to create the spatial DMs and 4th order Runge-Kutta in time. For the cosine bell test case, we use 3rd degree PHS, $\phi(r) = r^3$, with added polynomials up to 5th degree. Table 1 compares the accuracy in the relative l_1, l_2, and l_∞ to other methods used in the literature, such as spectral element, discontinuous Galerkin, finite volume, and RBF-PU (Partition of Unity) methods. With a similar number of degrees of freedom, the RBF-FD PHS+Poly method is shown to be very competitive and in most cases gives slightly lower errors.

Due to the jump in the second derivative of the compactly supported cosine bell, convergence with respect to total number of nodes is approximately second order, as can be seen in the left panel of Figure 3. However, using a smooth Gaussian bell initial condition q_2, the order of convergence of the method is limited only by the choice of d and n in the model, and achieves approximately 6th order convergence with $d = 6$, as can be seen in the right panel of Figure 3. Any order spatial convergence can be achieved by simply increasing the parameters d and n in the model.

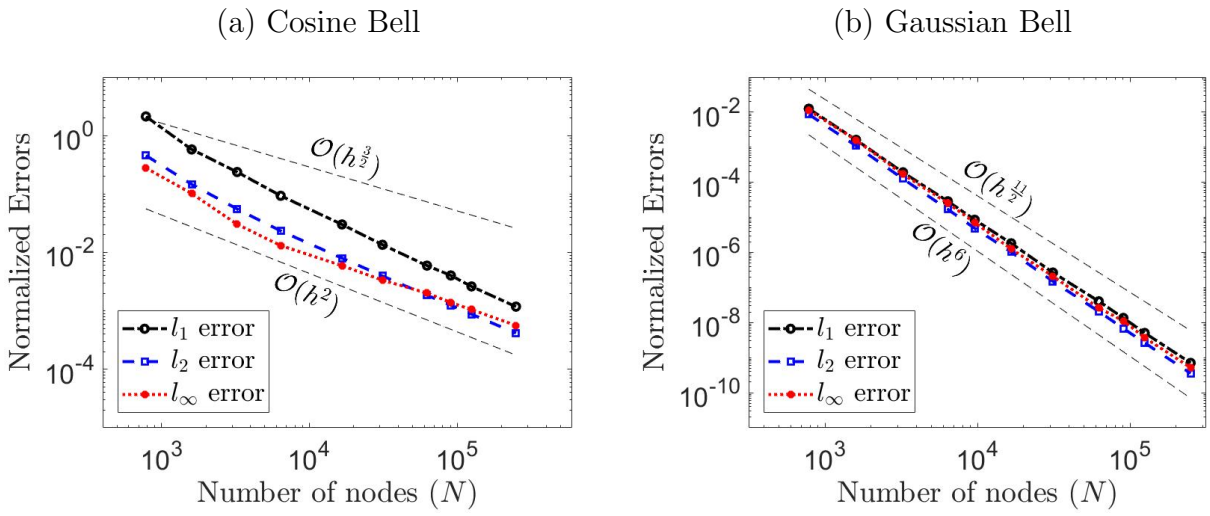

Figure 3: Using $d = 5$ for the cosine bell test case and $d = 6$ for the Gaussian bell, $n = 37$ nodes per stencil, and ME node sets, (a) a log-log plot of convergence of error for the cosine bell initial condition (q_0) with dashed lines representing 3/2 and 2nd -order convergence and (b) a log-log plot of convergence of error for the Gaussian bell initial condition (q_1) with dashed lines representing 11/2 and 6th -order convergence. Note that the reference slopes are given as powers of h, since these powers directly reflect the methods order of accuracy; h is here related to N as $h = O(1/\sqrt{N})$.

In Figure 4, the l_2 errors after one rotation for different choices of n, d, and N are plotted for solid body rotation with a cos bell and Gaussian initial condition. The dashed lines in these plots display the stencil size n for a given d at which the matrix system in (3) will become singular, (e.g. there are 10 polynomials up to degree $d = 3$, which require a minimum of 11 distinct node points). Due to the discontinuity in the second derivative of the cosine bell, the best error achieved at $N = 25600$ regardless of the degree of the polynomial used, is between $O(10^{-3})$ and $O(10^{-2})$. This is in contrast to the C^∞ Gaussian bell, where the error is completely controlled by d , regardless of what stencil size is used.

An artificial diffusion term of the form described in Section 5.2 was added to the the PDE to ensure stability of the method when time-stepping for longer than one revolution. We tuned the parameter γ_N with $N = 1600$ such that the eigenvalues of the right hand side matrix in Equation (9) had no positive real parts. This yielded a hyperviscosity parameter of $\gamma_{1600} \approx 10^{-10}$. Then, we used the scaling law $\gamma_N = (\frac{1600}{N})^k \gamma_{1600}$ to find values for γ_N at higher resolutions [7]. Figure 5 displays the effect of the hyperviscosity operator on the eigenvalues of the differential operator with $N = 1600$ and $N = 6400$ nodes. The figure shows that spurious eigenvalues in the spatial differentiation operator have been pushed into the stability region of the 4th-order Runge-Kutta time-stepper, with little disturbance of the physically relevant eigenvalues.

The hyperviscosity operator ensures that the PDE can be time-stepped for much longer time periods without becoming unstable. In order to illustrate this advantage, we let the cosine bell rotate 1000 times around the sphere in the case of $N = 25600$ MD nodes. The exact solution and the error after $T = 1, 10, 100$, and 1000 revolutions are plotted in Figure 6.

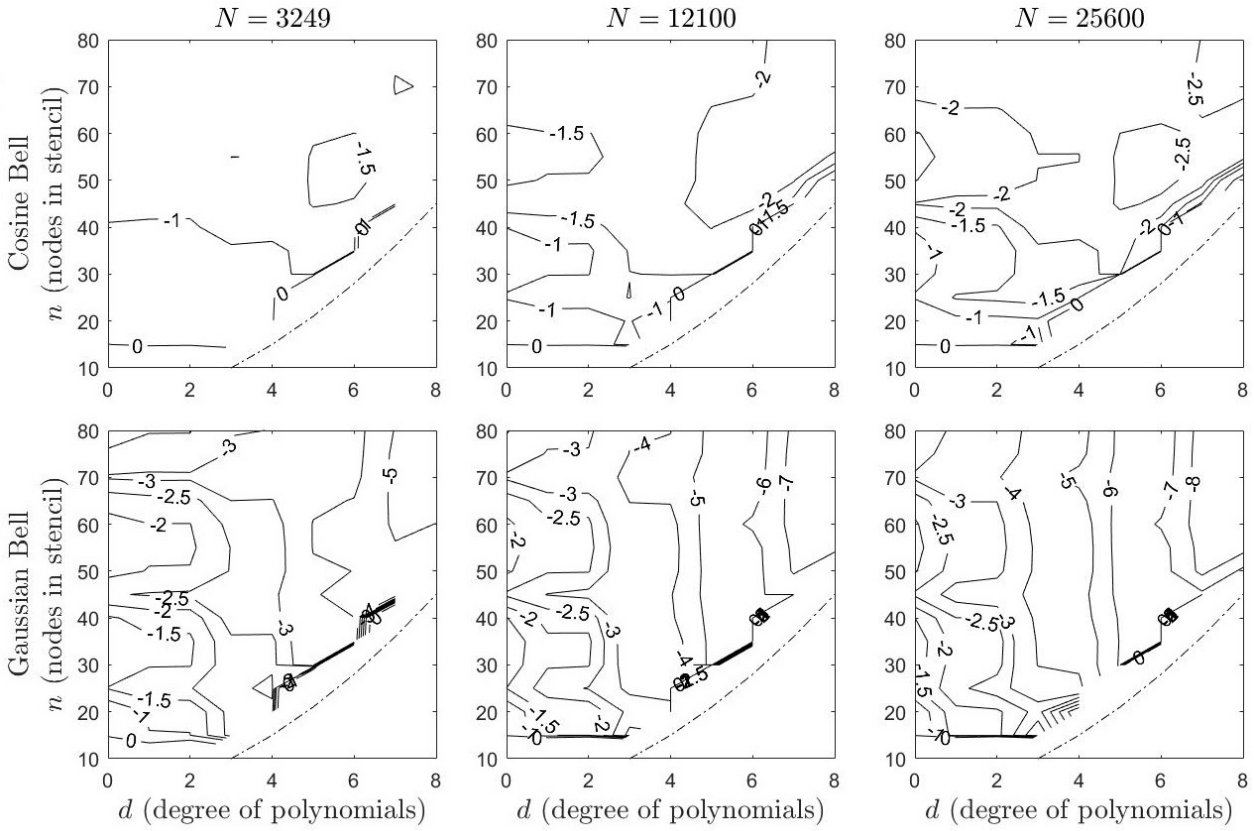

Figure 4: Contour plots of Log10 of the l_2 error after one rotation without hyperviscosity as a function of d and n for different N; top row is the solid body rotation test case with a cosine bell; bottom row is the same test case with an infinitely smooth Gaussian bell initial condition. The curved dotted line in the bottom right of each plot represents the point at which weight calculation matrices become ill-posed due to the inclusion of too many polynomial terms for stencil size n, $n \leq \binom{d+2}{d}$ in 2D. Contour lines represent interpolations, since both n and d can only take on integer values.

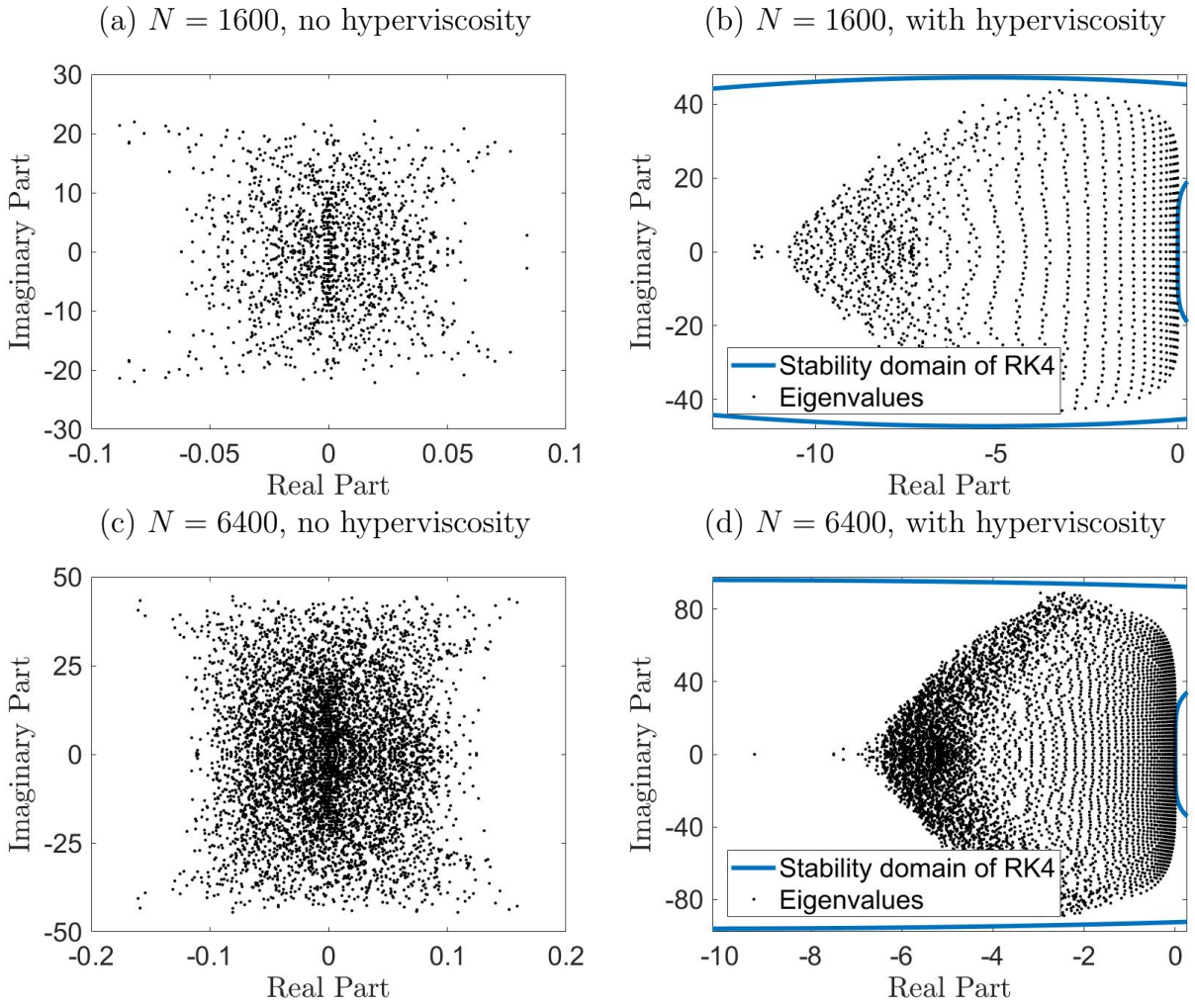

Figure 5: The effect of hyperviscosity on the eigenvalues of the spatial differentiation matrix with $N = 1600$ and $N = 6400$ nodes.

Note, the decrease in the height of the bell after 1000 revolutions is only about 4% percent. Instead, the overall error is mainly concentrated around the base of the cosine bell, where the second-order discontinuity in the initial condition causes dispersive errors. For $N \gg 5000$, time-stepping for 1000 rotations required slightly different tuning of the parameter γ_N to ensure stability; γ_N is approximately halved if $N \geq O(10^5)$. The results above compare very favorably with earlier local RBF calculations in the literature. In [23], the bell was convected 10 rotations using both a local and partition of unity RBF-based semi-Lagrangian method, with $n = 49$ and $N = 40,962$. In [11] (where the concept of hyperviscosity for RBF-FD was introduced), an $n = 74$ calculation with $N = 25,600$ nodes was run for 1, 10 and $1,000$ rotations. In all cases cited above, the dominant error at the final time occurred around the base of the bell, as with in this case, and was the same order of magnitude.

Computational time as a function of accuracy (by varying N with ME node sets) for the solid body rotation test case was tested on a dual core, 2.7 Ghz laptop with $d = 5$, $n = 37$. Figure 7 shows that computational time versus accuracy are approximately linear for both

Table 2: Constants used in the Hadley-like meridional circulation test case

Constant	Value	Description
τ	86400s	Period of motion (here 1 day)
K	5	Number of overturning cells
u_0	40ms^{-1}	Reference zonal velocity
w_0	0.15ms^{-1}	Reference vertical velocity
z_1	2000m	Lower boundary of tracer layer
z_2	5000m	Upper boundary of tracer layer
a	$6.37122 \times 10^6\text{m}$	Radius of the Earth
g	9.80616ms^{-2}	Gravity
c_p	1004.5J kg^{-1} K^{-1}	Specific heat capacity of dry air
R_d	287.0J kg^{-1} K^{-1}	Gas constant for dry air
κ	$R_d/c_p = 2/7$	Ratio of R_d to c_p
z_{top}	12000m	Height position of the model top
T_0	300K	Isothermal atmospheric temperature

pre-processing of differentiation matrices and for time-stepping one rotation, as predicted based on the expected computational complexity of $\mathcal{O}(N)$.

5.4 Test Case 2: 3D Hadley-like meridional circulation

The Hadley-like meridional circulation cell test case is defined in [15] and was used as one of the Dynamical Core Model Intercomparison Project's (DCMIP) tracer test cases in 2012 [12]. A dynamical core governs the evolution of the dynamical processes in weather forecasting and climate modeling. Many transport algorithms in dynamical cores are horizontally-vertically split due to the difference in spatial scales. The initial tracer field is time-stepped for one day according to the transport equation (7) with a prescribed nonlinear time-dependent velocity field that deforms it in the Earth's atmosphere .

The zonal, meridional, and vertical velocity field for this test is specified as

$$u(\lambda, \phi, z, t) = u_0 \cos(\phi) \tag{13}$$

$$v(\lambda, \phi, z, t) = -\frac{a w_0 \pi \rho_0}{K z_{\text{top}} \rho} \cos(\phi) \sin(K\phi) \times \cos\left(\frac{\pi z}{z_{\text{top}}}\right) \cos\left(\frac{\pi t}{\tau}\right) \tag{14}$$

$$w(\lambda, \phi, z, t) = \frac{w_0 \rho_0}{K \rho} \left[-2 \sin(K\phi) \sin(\phi) + K \cos(\phi) \cos(K\phi)\right] \sin\left(\frac{\pi z}{z_{\text{top}}}\right) \cos\left(\frac{\pi t}{\tau}\right) \tag{15}$$

where the above constants with explanations are listed in Table 2.

The initial tracer field is defined as

$$q_0(\lambda, \phi, z) = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{2\pi(z-z_0)}{z_2-z_1}\right)\right] \text{ if } z_1 < z < z_2, \\ 0 \qquad\qquad\qquad\qquad \text{otherwise,} \end{cases} \tag{16}$$

where $z_0 = \frac{1}{2}(z_1 + z_2)$. Since this initial condition contains a discontinuity in the second-derivative, the maximum order of convergence should theoretically be 2nd order. However, we do get orders of convergence greater up to 2.8, depending on the error norm used.

5.4.1 Numerical results

In this test case, we consider evenly-spaced nested spheres of nodes. For tangential derivatives, we use r^3 PHS with $d = 5$ and $n = 55$. In the radial direction, we use centered five-point finite differences on equi-spaced nodes to calculate radial differentiation weights for interior nodes. Near the boundary, we use one-sided five-point finite differences as calculated in [8]. Fourth order explicit Runge-Kutta is used for time-stepping. Although hyperviscosity might allow for longer time integration without running into instabilities, we found it to be unnecessary for the one day time integration called for by this test case. The tracer field after $0h$, $12h$, and $24h$, as well as the error after $24h$ is shown in Figure 8. The error appears mostly to be due to dispersive errors caused by the discontinuity in the second derivative of the initial condition in the radial direction.

Tables 3, 4, and 5 show results in the l_1, l_2, and l_∞ norms, respectively, in comparison with various methods in the atmospheric literature, each of which is described briefly in the next section. Time steps of $\delta t = 1800s$, $\delta t = 720s$, $\delta t = 480s$ were used respectively for each resolution of $2^\circ \times 2^\circ$ (220km) with 30 (400m) nested shells, $1^\circ \times 1^\circ$ (120km) with 60 nested shells (200m), and $1/2^\circ \times 1/2^\circ$ (60km) with 120 (100m) nested shells. For the lowest resolution, we used a $12,100$ node MD node set. For the higher resolutions, we used icosahedral node sets of size $40,962$ nodes and $163,842$ nodes. The RBF-FD PHS+Poly method error is better by approximately an order of magnitude in all resolutions as well as the rate of convergence, regardless of the norm used. The improvement in error can be partially attributed to the RBF-FD method does not require a deformation of the underlying domain, such as the cubed sphere or latitude-longitude grid, as is done in the other methods.

5.4.2 Brief description of comparison methods

CAM is NCAR's Community Atmosphere Model, described in [19]. In CAM-FV, the finite-volume version of CAM, the horizontal tracer transport component uses a flux-form semi-Lagrangian Finite Volume method on a lat-long grid. In CAM-SE, a high-order continuous Galerkin spectral element method on a cubed-sphere mesh is used for discretizatio in the horizontal direction, which is described by Taylor et al. [25]. MCore is a dynamical core, described in [26], that uses high-order upwind finite-volume methods on a cubed-sphere grid (also used in CAM-SE). FV3 is the dynamical core to be used in NOAA's next generation Environmental Modelling System infrastructure. It uses finite volumes on a cubed sphere with a Lagrangian, terrain-following vertical coordinate, described in [16, 17]

Table 3: Comparison of advection schemes in the relative l_1 norm for the DCMIP test case

Resolution	CAM-FV	CAM-SE	MCore	FV3	RBF-FD PHS+Poly
$2^\circ \times 2^\circ$ 30 shells	$1.8E{-}1$	$1.3E{-}1$	$1.4E{-}1$	$8.4E{-}2$	$2.9E{-}2$
$1^\circ \times 1^\circ$ 60 shells	$4.1E{-}2$	$2.9E{-}2$	$2.9E{-}2$	$2.2E{-}2$	$3.2E{-}3$
$1/2^\circ \times 1/2^\circ$ 120 shells	$1.2E{-}2$	$1.0E{-}2$	$6.3E{-}3$	$1.5E{-}2$	$6.1E{-}4$
Convergence	1.93	1.86	2.22	1.33	2.80

Table 4: Comparison of advection schemes in the relative l_2 norm for the DCMIP test case

Resolution	CAM-FV	CAM-SE	MCore	FV3	RBF-FD PHS+Poly
$2^\circ \times 2^\circ$ 30 shells	$2.0E{-}1$	$1.4E{-}1$	$1.7E{-}1$	$9.7E{-}2$	$1.8E{-}2$
$1^\circ \times 1^\circ$ 60 shells	$5.4E{-}2$	$3.2E{-}2$	$4.6E{-}2$	$2.1E{-}2$	$3.1E{-}3$
$1/2^\circ \times 1/2^\circ$ 120 shells	$1.6E{-}2$	$1.2E{-}2$	$1.1E{-}2$	$1.5E{-}2$	$7.4E{-}4$
Convergence	1.84	1.79	1.94	1.46	2.37

Table 5: Comparison of advection schemes in the relative l_∞ norm for the DCMIP test case

Resolution	CAM-FV	CAM-SE	MCore	FV3	RBF-FD PHS+Poly
$2^\circ \times 2^\circ$ 30 shells	$4.7E{-}1$	$3.8E{-}1$	$4.2E{-}1$	$2.3E{-}1$	$8.9E{-}2$
$1^\circ \times 1^\circ$ 60 shells	$1.6E{-}1$	$1.0E{-}1$	$1.6E{-}1$	$5.4E{-}2$	$1.1E{-}2$
$1/2^\circ \times 1/2^\circ$ 120 shells	$4.7E{-}2$	$4.6E{-}2$	$4.3E{-}2$	$2.9E{-}2$	$8.1E{-}3$
Convergence	1.66	1.52	1.64	1.61	2.35

5.5 Summary

We present a new numerical algorithm based on Householder reflections for calculating differentiation weights to first derivatives in Cartesian coordinates, $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, and $\frac{\partial}{\partial z}$, on the sphere using RBF-FD polyharmonic splines supplemented with 2D polynomials. The code, attached to this paper, is concise - only 17 lines. The beauty of the method is that it only requires defining N 3×3 Householder matrices, N being the total number of points on the sphere. Each Householder matrix transforms a point on the sphere to one of the coordinate axis, here chosen to be (1,0,0); however (0,1,0) or (0,0,1) would have worked just as well. The same matrix is then used to transform the entire stencil associated with that point. The method was tested on two standard advection test cases, one on a sphere and the second within a 3D spherical shell. It was shown to be competitive, and in most cases outperformed a variety of methods currently used, including spectral elements, finite elements, finite volume, and other RBF methods in terms of accuracy and convergence.

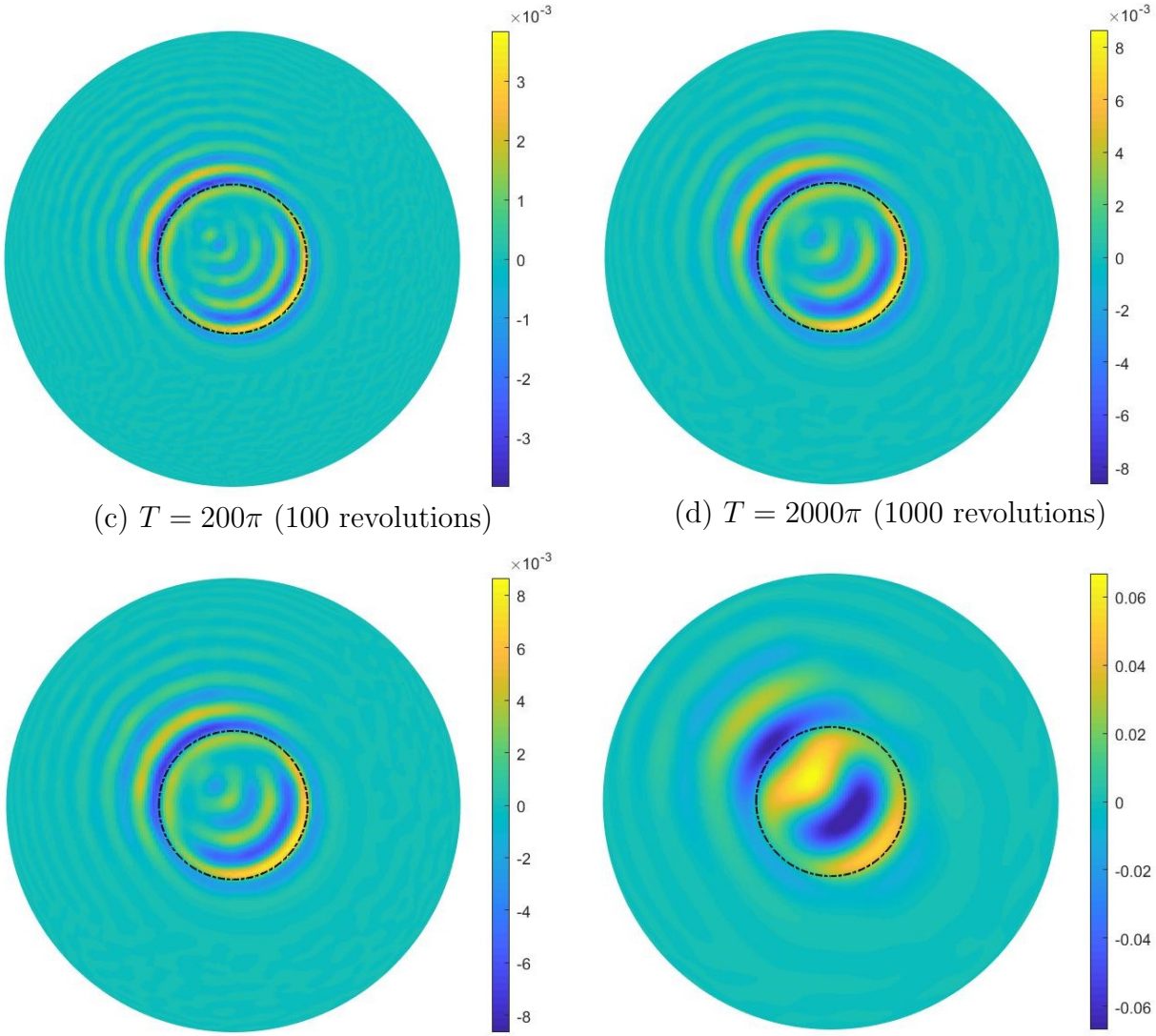

Figure 6: Error in the solid body rotation test case with $n = 37$, $d = 5$, and an MD node set of size $N = 25600$. The error appears to increase approximately logarithmically with the number of revolutions even for long periods of time, implying that the hyperviscosity effectively ensures stability. Strategies for tuning the hyperviscosity parameter can be found in [7, 11, 21]

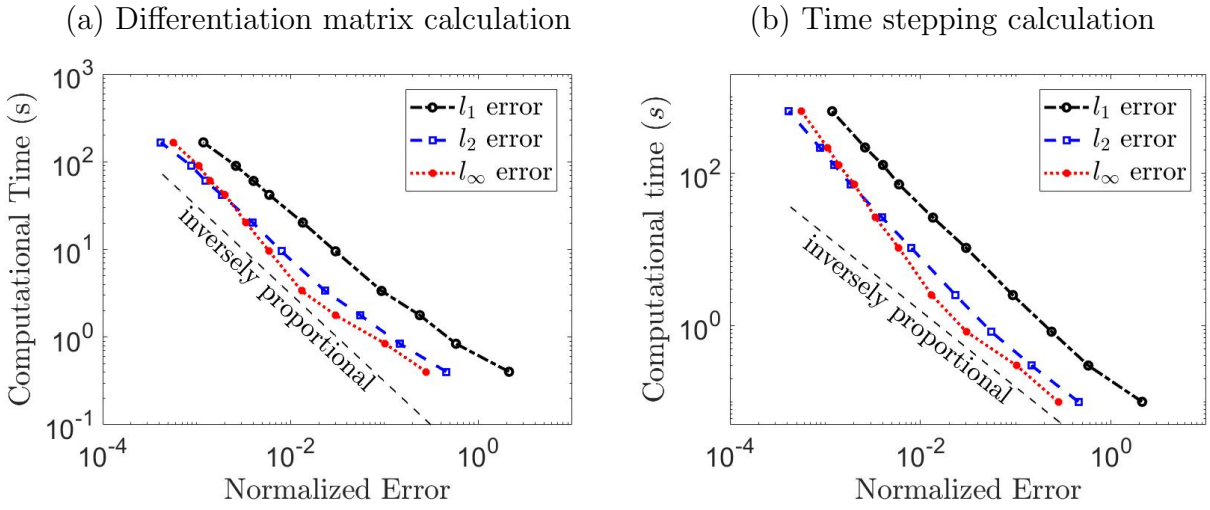

Figure 7: For the solid body rotation test case, with varying size ME node sets, $d = 5$ and $n = 37$(a) a log-log plot of computation time to calculate differentiation matrices as a function of accuracy for the solid body rotation test case. (b) a loglog plot of computation time to evolve the initial condition one rotation as a function of accuracy for the solid body rotation test case.

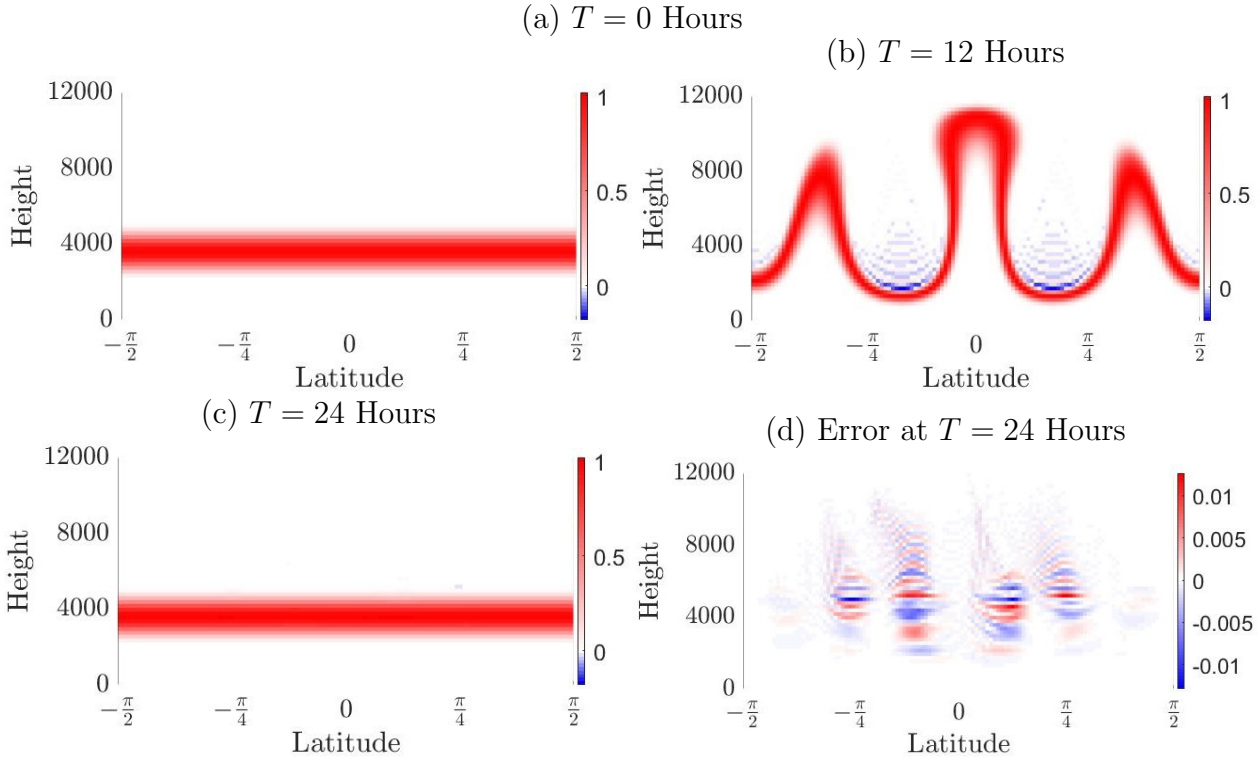

Figure 8: A representative cross section of the tracer field at $\lambda = \pi$ for the Hadley-like meridional circulation test case with $n = 55$ and $d = 5$, 60 shells in the vertical and a horizontal resolution of $1^\circ \times 1^\circ$. The tracer field is the same at all values of λ. Plots (a)-(c) are the tracer field itself, while plot (d) is the error in the tracer field after 24 hours.

Appendix A:

```
1  function [Dx,Dy,Dz] = D_xyz_PHS(xyz,IDX,m,d)
2
3  % Calculates the RBF-FD DMs associated to d/dx, d/dy, d/dz
4  % Input parameters
5  %      xyz   All nodes on the sphere surface, array size (N,3)
6  %      IDX   Array (N,n) of n neighbors for each of N nodes on
       sphere
7  %        m   Power used in the radial function, r^m
8  %        d   Degree of polynomial terms included
9  % Output parameters
10 % Dx,Dy,Dz   Sparse NxN matrices with RBF-FD weights
11
12 [N,n] = size(IDX);              % Find problem size
13 Wx = zeros(N,n); Wy = zeros(N,n); Wz = zeros(N,n);
14 for k = 1:N                     % Loop over N stencils
15     X = xyz(IDX(k,:),:);       % Find nodes in kth stencil
16     Xt = -X(1,:)'; y1 = 1; if Xt(1) < 0; y1 = -1; end;
17     Xt(1) = y1+Xt(1); Wh = Xt/norm(Xt);
18     Q = eye(3)-2*(Wh*Wh');     % Define Householder matrix Q
19     X = X*Q';
20     w = RBF_FD_PHS_pol_weights_sph (X(:,2),X(:,3),m,d);
21                       % Get weights in plane using 2-D algorithm
22     w = w*Q(2:3,:); % Transform weights to orginal stencil
23     Wx(k,:) = w(:,1)'; Wy(k,:) = w(:,2)'; Wz(k,:) = w(:,3)';
24 end
25 it = (1:N)'; it = it(:,ones(1,n));
26 Dx = sparse(it(:),IDX(:),Wx(:),N,N);
27 Dy = sparse(it(:),IDX(:),Wy(:),N,N);
28 Dz = sparse(it(:),IDX(:),Wz(:),N,N);
29         % Re-arrange weights into DMs for d/dx, d/dy, d/dz
```

This Matlab code performs the algorithm described in Section 3.3. Lines 16-19 calculate and apply the Householder reflection to the stencil. Line 20 references a subroutine which can be found in Appendix B of [5] for calculating PHS+Polys RBF-FD weights in a plane. Line 22 transforms the weights back to the original stencil position.

References

[1] V. Bayona, N. Flyer, and B. Fornberg. On the role of polynomials in RBF-FD approximations: III. Behavior near domain boundaries. *Journal of Computational Physics*, 380:378–399, 2019.

[2] V. Bayona, N. Flyer, B. Fornberg, and G.A. Barnett. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs. *Journal of Computational Physics*, 332:257–273, 2017.

[3] V. Bayona, N. Flyer, G.M. Lucas, and A.J.G. Baumgaertner. A 3-D RBF-FD solver for modeling the atmospheric global electric circuit with topography (GEC-RBFFD v1. 0). *Geoscientific Model Development*, 8(10):3007, 2015.

[4] G.E. Fasshauer. *Meshfree approximation methods with MATLAB*, volume 6. World Scientific, 2007.

[5] N. Flyer, G.A. Barnett, and L.J. Wicker. Enhancing finite differences with radial basis functions: Experiments on the Navier–Stokes equations. *Journal of Computational Physics*, 316:39–62, 2016.

[6] N Flyer, B. Fornberg, V. Bayona, and G.A. Barnett. On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy. *Journal of Computational Physics*, 321:21–38, 2016.

[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. *Journal of Computational Physics*, 231(11):4078–4095, 2012.

[8] B. Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, 51(184):699–706, 1988.

[9] B. Fornberg and N. Flyer. *A primer on radial basis functions with applications to the geosciences*. SIAM Press, 2015.

[10] B. Fornberg and N. Flyer. Solving PDEs with radial basis functions. *Acta Numerica*, 24:215–258, 2015.

[11] B. Fornberg and E. Lehto. Stabilization of RBF-generated finite difference methods for convective PDEs. *Journal of Computational Physics*, 230(6):2270–2285, 2011.

[12] D.M. Hall, P.A. Ullrich, K.A. Reed, C. Jablonowski, R.D. Nair, and H.M. Tufo. Dynamical core model intercomparison project (DCMIP) tracer transport test results for CAM-SE. *Quarterly Journal of the Royal Meteorological Society*, 142(697):1672–1684, 2016.

[13] D.P. Hardin, T. Michaels, and E.B. Saff. A comparison of popular point configurations on S^2. *Dolomites Research Notes on Approximation*, 9(1), 2016.

[14] K. Jetter, J. Stöckler, and J.D. Ward. Error estimates for scattered data interpolation on spheres. *Math. Comput.*, 68:733–747, 1999.

[15] J. Kent, P.A. Ullrich, and C. Jablonowski. Dynamical core model intercomparison project: Tracer transport test cases. *Quarterly Journal of the Royal Meteorological Society*, 140(681):1279–1293, 2014.

[16] S. Lin. A vertically Lagrangian finite-volume dynamical core for global models. *Monthly Weather Review*, 132(10):2293–2307, 2004.

[17] S. Lin and R.B. Rood. An explicit flux-form semi-Lagrangian shallow-water model on the sphere. *Quarterly Journal of the Royal Meteorological Society*, 123(544):2477–2498, 1997.

[18] R.D. Nair, S.J. Thomas, and R.D. Loft. A discontinuous Galerkin transport scheme on the cubed sphere. *Monthly Weather Review*, 133(4):814–828, 2005.

[19] R.B. Neale, C. Chen, A. Gettelman, P.H. Lauritzen, S. Park, D.L. Williamson, A.J. Conley, R. Garcia, D. Kinnison, J. Lamarque, et al. Description of the NCAR community atmosphere model (CAM 5.0). *NCAR Tech. Note NCAR/TN-486+ STR*, 1(1):1–12, 2010.

[20] J.A. Reeger and B. Fornberg. Numerical quadrature over the surface of a sphere. *Studies in Applied Mathematics*, 137(2):174–188, 2016.

[21] V. Shankar and A.L. Fogelson. Hyperviscosity-based stabilization for radial basis function-finite difference (RBF-FD) discretizations of advection-diffusion equations. *Journal of Computational Physics*, 372(1):616–639, 2018.

[22] V. Shankar, A. Narayan, and R.M. Kirby. RBF-LOI: Augmenting radial basis functions (RBFs) with least orthogonal interpolation (LOI) for solving PDEs on surfaces. *Journal of Computational Physics*, 373:722–735, 2018.

[23] V. Shankar and G.B. Wright. Mesh-free semi-Lagrangian methods for transport on a sphere using radial basis functions. *Journal of Computational Physics*, 366:170–190, 2018.

[24] I.H. Sloan and R.S. Womersley. Extremal systems of points and numerical integration on the sphere. *Advances in Computational Mathematics*, 21(1-2):107–125, 2004.

[25] M. Taylor, J. Tribbia, and M. Iskandarani. The spectral element method for the shallow water equations on the sphere. *Journal of Computational Physics*, 130(1):92–108, 1997.

[26] P.A. Ullrich and C. Jablonowski. MCore: A non-hydrostatic atmospheric dynamical core utilizing high-order finite-volume methods. *Journal of Computational Physics*, 231(15):5078–5108, 2012.

[27] P.A. Ullrich, C. Jablonowski, and B. Van Leer. High-order finite-volume methods for the shallow-water equations on the sphere. *Journal of Computational Physics*, 229(17):6104–6134, 2010.

[28] D.L. Williamson, J.B. Drake, J.J. Hack, R. Jakob, and P.N. Swarztrauber. A standard test set for numerical approximations to the shallow water equations in spherical geometry. *Journal of Computational Physics*, 102(1):211–224, 1992.

[29] R.S. Womersley and I. H. Sloan. Interpolation and cubature on the sphere. `https://web.maths.unsw.edu.au/~rsw/Sphere/`.

[30] R.S. Womersley and I.H. Sloan. How good can polynomial interpolation on the sphere be? *Advances in Computational Mathematics*, 14(3):195–226, 2001.