SIAM J. SCI. COMPUT.
Vol. 42, No. 4, pp. A2371–A2401

A ROBUST HYPERVISCOSITY FORMULATION FOR STABLE RBF-FD DISCRETIZATIONS OF ADVECTION-DIFFUSION-REACTION EQUATIONS ON MANIFOLDS*

VARUN SHANKAR†, GRADY B. WRIGHT‡, AND AKIL NARAYAN§

Abstract. We present a new hyperviscosity formulation for stabilizing radial basis function-finite difference (RBF-FD) discretizations of advection-diffusion-reaction equations on manifolds $\mathbb{M} \subset \mathbb{R}^3$ of codimension 1. Our technique involves automatic addition of artificial hyperviscosity to damp out spurious modes in the differentiation matrices corresponding to surface gradients, in the process overcoming a technical limitation of a recently developed Euclidean formulation. Like the Euclidean formulation, the manifold formulation relies on von Neumann stability analysis performed on auxiliary differential operators that mimic the spurious solution growth induced by RBF-FD differentiation matrices. We demonstrate high-order convergence rates on problems involving surface advection and surface advection-diffusion. Finally, we demonstrate the applicability of our formulation to advection-diffusion-reaction equations on manifolds described purely as point clouds. Our surface discretizations use the recently developed RBF-least orthogonal interpolation method and, with the addition of hyperviscosity, are now empirically high-order accurate, stable, and free of stagnation errors.

Key words. radial basis functions, high-order methods, manifolds, transport, advection-diffusion

AMS subject classifications. 65M06, 65M70, 65M12, 65L20

DOI. 10.1137/19M1288747

1. Introduction.

Radial basis functions (RBFs) are a powerful and flexible tool for generating numerical methods for the solution of partial differential equations (PDEs). RBF collocation methods are easily applied to solving PDEs on irregular domains using scattered node layouts [5, 7, 44]. RBF-based methods also generalize naturally to the solution of PDEs on manifolds $\mathbb{M} \subset \mathbb{R}^3$ using only the Euclidean distance measure in the embedding space and Cartesian coordinates; see, for example, [1, 12, 13, 14, 16, 20, 22, 30, 31, 38, 39, 40]. We focus exclusively on polynomially augmented RBF-finite difference (RBF-FD) methods for their ability to harness the best features of both polynomial and RBF approximation.

It is natural to consider two types of (linear) stability in the context of RBF-FD methods: local stability and global stability. Local stability refers to stability in the procedure of computing RBF-FD weights on each stencil. Global stability, on the other hand, refers to eigenvalue characterizations of differentiation matrices in RBF-FD discretizations of time-dependent PDEs: A globally stable discretization

*Submitted to the journal's Methods and Algorithms for Scientific Computing section September 23, 2019; accepted for publication (in revised form) April 21, 2020; published electronically August 17, 2020.

https://doi.org/10.1137/19M1288747

Funding: The work of the first author was supported by the NSF through grants DMS-1521748 and CISE 1714844. The work of the second author was supported by the NSF through grant CCF 1717556. The work of the third author was partially supported by the NSF through grant DMS-1720416 and by AFOSR through grant FA9550-15-1-0467.

†Corresponding author. School of Computing, University of Utah, Salt Lake City, UT 84112 (shankar@cs.utah.edu).

‡Department of Mathematics, Boise State University, Boise, ID 83725 (gradywright@boisestate.edu).

§Department of Mathematics, and Scientific Computing and Imaging (SCI) Institute, University of Utah, Salt Lake City, UT 84112 (akil@sci.utah.edu).

for hyperbolic or elliptic operators would imply the absence of eigenvalues with positive real parts. Historically, RBF-FD methods have been locally unstable due to ill-conditioning in the RBF interpolation matrix [8, 42], manifesting as *stagnation* in errors as the number of nodes is increased. Fortunately, work in the past two decades has helped overcome this on Euclidean domains, either by changing the RBF basis [9, 15, 17, 18], by using rational approximations in the complex plane [19, 45], or by augmenting RBFs with high-degree polynomials [4, 10, 11]. On manifolds the local stability problem has been addressed by using tangent plane projections on meshes to convert the problem to a Euclidean one [32, 33, 34]; least squares [28]; the closest-point method, which extends the problem to ambient space [29]; or a careful restriction of multivariate polynomials to the manifold [38].

In the context of global stability, the augmenting of polyharmonic spline (PHS) RBFs with polynomials also appears to enhance global stability for elliptic operators on Euclidean domains [3], though this appears to depend on the node sets used (see [35, 36] for evidence to the contrary of [3]). However, the global stability problems in discretizations of hyperbolic operators are a topic of ongoing research. Unfortunately, while the RBF-least orthogonal interpolation (LOI) method presented in [38] appears to be both locally and globally stable for elliptic operators on manifolds, it (like all RBF-FD methods) is not globally stable for hyperbolic operators. On the sphere (and in Euclidean domains), the current state of the art is to add a stabilizing artificial hyperviscosity term of the form $\gamma_1 \Delta^{\gamma_2}$, $\gamma_1 \in \mathbb{R}$, $\gamma_2 \in \mathbb{N}$, an approach that was first used in the spectral methods literature under the title "spectral (super) viscosity" [23, 24, 41]. This was introduced to the RBF-FD context in [16], with empirical formulas for γ_1 and γ_2 provided in [12]. However, until recently, selecting γ_1 varied across applications [10].

In recent work [36], the first author derived the first closed-form quasi-analytic formula for γ_1 and adapted a fully analytic formula from [23, 24] for γ_2 on Euclidean domains (applied to linear PDEs). These formulas generalize the spectral super-viscosity formulas [23, 24, 41] to scattered nodes and RBF-FD discretizations and eliminate the need for parameter hand-tuning. The key contribution of that work was to use a 1D von Neumann analysis on a *mathematical model* of the spurious growth modes of RBF-FD operators via so-called *auxiliary derivatives* and *auxiliary PDEs*. This approach allowed the author to derive quasi-analytic hyperviscosity formulas for stabilizing gradients and Laplacians on Euclidean domains of arbitrary dimension.

The goal of this article is to extend the Euclidean analysis of [36] to manifolds $\mathbb{M} \subset \mathbb{R}^3$, thereby developing a robust quasi-analytic hyperviscosity formulation for RBF-FD discretizations on manifolds. When combined with the recently developed RBF-LOI method, the resulting numerical method is free of stagnation errors and is empirically both locally and globally stable for the examples we have investigated. To the best of our knowledge, this is the first globally stable polynomially augmented RBF-FD method of projection type for hyperbolic problems on manifolds other than the sphere. In addition, the use of the overlapped RBF-FD formulation [21, 35, 36, 38] makes the assembly of differentiation matrices efficient, especially for higher-order methods. Our hyperviscosity formulation is independent of RBF-LOI and can be used straightforwardly with other RBF-FD methods based on the projection approach [38, 40], the closest-point approach [28, 29], or the orthogonal gradients method [30, 31].

The remainder of this paper is organized as follows. In the next section, we review the RBF-LOI method on manifolds. In section 3, we review the Euclidean hyperviscosity formulation from [36] and use it to motivate our new quasi-analytic hyperviscosity formulation for manifolds; we verify that our formulation is valid for

important classes of multistage and multistep time integration schemes. We validate our methods on advection and advection-diffusion problems in section 4 by measuring errors and convergence rates. In section 5, we present applications of our method to solving nonlinear advection-diffusion-reaction equations on manifolds other than the sphere and torus. We conclude with a summary of our results and a discussion of future work in section 6.

2. RBF-LOI on surfaces. This section explains our spatial discretization methodology, which is based upon three main ideas:

- Section 2.1: Polynomially augmented RBF-FD is used as the fundamental approximation [4, 36, 44].
- Section 2.2: The choice of basis for the polynomials is made through the LOI procedure to ensure polynomial unisolvency for nodes on manifolds [38].
- Section 2.3: Overlapping is used to reduce the number of local stencil approximations that must be computed [21, 35, 36, 38].

We refer to our algorithm, which is the combination of these ingredients, as RBF-LOI on surfaces.

2.1. RBF-FD on surfaces. Let $X = \{\boldsymbol{x}_k\}_{k=1}^N$ be a set of nodes on $\mathbb{M} \subset \mathbb{R}^3$. Given a fixed $k \in \{1, 2, \ldots, N\}$, we will describe RBF approximations to surface differential operators over a *local stencil* of $n << N$ nodes for each $\boldsymbol{x}_k$. To denote the stencil nodes we use index sets as follows: For each $\boldsymbol{x}_k$, let P_k consist of $\boldsymbol{x}_k$ and its $n-1$ nearest neighbors (measured using the standard Euclidean distance), and let the set $\{\mathcal{I}_1^k, \ldots, \mathcal{I}_n^k\}$ denote the indices into the global node set X of the nodes in P_k, with $\mathcal{I}_1^k = k$. We refer to P_k as the local stencil for $\boldsymbol{x}_k$. The standard RBF-FD approach constructs such a stencil for each $k = 1, \ldots, N$, while the overlapped RBF-FD approach can use the same stencil for more than one node in X, thus reducing the total number of stencils and substantially reducing the computational cost of constructing the RBF-FD differentiation matrices (see section 2.3). For simplicity in the remainder of this discussion, we work with stencil $k = 1$; hence, all our quantities will feature sub/superscripts "1" that can be replaced by "k" to apply to a general stencil.

Suppose we wish to approximate the surface gradient $\nabla_{\mathbb{M}}$, defined in Cartesian coordinates, as

$$\nabla_{\mathbb{M}} = (I - \boldsymbol{n}\boldsymbol{n}^T)\nabla =: [\mathcal{G}^x, \mathcal{G}^y, \mathcal{G}^z]^T, \tag{1}$$

where $\boldsymbol{n}$ is the unit outward normal to the surface and ∇ is the $\mathbb{R}^3$ gradient. We will focus our discussion on the surface gradient; the surface Laplacian is then computed from the surface gradient in an iterated fashion as in [38, 40].

Focusing on the $\mathcal{G}^x$ component of $\nabla_{\mathbb{M}}$, the overlapped RBF-FD weights for every node in the stencil P_1 are computed using a *family* of polynomially augmented local RBF interpolants on P_1 parametrized by variables $\boldsymbol{x}$ and $\boldsymbol{y}$,

$$s_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{n} (w^x)_j(\boldsymbol{y}) \|\boldsymbol{x} - \boldsymbol{x}_{\mathcal{I}_j^1}\|^m + \sum_{i=1}^{M} \lambda_i(\boldsymbol{y}) h_i^1(\boldsymbol{x}), \tag{2}$$

where all superscripts "1" refer to the stencil index, $\|\cdot\|$ denotes the two norm, m is odd and corresponds to the order of the PHS RBF, and $\left\{h_i^1(\cdot)\right\}_{i=1}^M$ is a basis for trivariate polynomials of degree ℓ. In 3D Euclidean domains, a standard choice for the polynomial basis would be M monomials corresponding to the total-degree ℓ of

the trivariate polynomial subspace. On a (locally) algebraic manifold, this choice typically results in a numerical lack of polynomial unisolvency on the local stencil. The RBF-LOI augmentation described in section 2.2 mitigates this issue. Regardless, the polynomial coefficients $\lambda_i(\boldsymbol{y})$ serve as Lagrange multipliers that enforce polynomial reproduction on the RBF-FD weights; this is discussed further below.

Here $\boldsymbol{x}$ refers to the nodes used to compute the weights, while $\boldsymbol{y}$ refers to the location at which the weights $(w^x)_j(\boldsymbol{y})$ are computed. In other words, each interpolant in the family is given by varying $\boldsymbol{y}$, with the n RBF-FD weights for that interpolant given by $(w^x)_j(\boldsymbol{y})$.[1] In the standard RBF-FD method, $\boldsymbol{y} \equiv \boldsymbol{x}_1$ (the stencil center), which allows us to omit the $\boldsymbol{y}$ parameter altogether; in the overlapped RBF-FD method, $\boldsymbol{y}$ allows for computation of weights in some radius around the stencil center (see section 2.3).

The overlapped RBF-FD weights for the operators $\mathcal{G}^x$,$\mathcal{G}^y$, and $\mathcal{G}^z$ at all stencil points $\boldsymbol{x}_{\mathcal{I}_j^1}$ (every point $\boldsymbol{y} \in P_1$) are computed by imposing certain conditions on an appropriate version of (2); e.g., for $\mathcal{G}^y$ we replace (w^x) in (2) with (w^y). Considering the operator $\mathcal{G}^x$ without loss of generality, we impose the following two (sets of) conditions on the interpolants (2):

$$s_1(\boldsymbol{x}_{\mathcal{I}_j^1}, \boldsymbol{x}_{\mathcal{I}_i^1}) = \left(\mathcal{G}^x \|\boldsymbol{x} - \boldsymbol{x}_{\mathcal{I}_j^1}\|^m\right)\Big|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}_i^1}}, \qquad i = 1,\dots,n, j = 1,\dots,n, \tag{3a}$$

(3b)
$$\sum_{j=1}^{n} (w^x)_j^1(\boldsymbol{x}_{\mathcal{I}_k^1}) h_i^1(\boldsymbol{x}_{\mathcal{I}_j^1}) = \left(\mathcal{G}^x h_i^1(\boldsymbol{x})\right)\big|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}_k^1}}, \qquad k = 1,\dots,n, i = 1,\dots,M.$$

The first set of conditions enforces that $s_1(\boldsymbol{x}, \boldsymbol{y})$ interpolate the derivatives of the PHS RBF at all the points in P_1. The second set of conditions enforces polynomial reproduction/exactness on *all* the overlapped RBF-FD weights using the polynomial coefficients $\lambda_i(\boldsymbol{y})$. If a degree-$\ell$ polynomial space is employed for $h^1(\boldsymbol{x})$, then $M = \binom{\ell+d}{d}$; we discuss choosing ℓ further below. The (family of) interpolants (2) and the two conditions (3a)–(3b) can be collected into the following block linear system:

$$\begin{bmatrix} A_1 & H_1 \\ H_1^T & O \end{bmatrix} \begin{bmatrix} G_1^x \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} B_{A_1} \\ B_{H_1} \end{bmatrix}, \tag{4}$$

where

$$(A_1)_{ij} = \|\boldsymbol{x}_{\mathcal{I}_i^1} - \boldsymbol{x}_{\mathcal{I}_j^1}\|^m, \qquad i,j = 1,\dots,n, \tag{5}$$

$$(H_1)_{ij} = h_j^1(\boldsymbol{x}_{\mathcal{I}_i^1}), \qquad i = 1,\dots,n,\ j = 1,\dots,M, \tag{6}$$

$$(B_{A_1})_{ij} = \mathcal{G}^x \|\boldsymbol{x} - \boldsymbol{x}_{\mathcal{I}_j^1}\|^m\Big|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}_i^1}}, \qquad i,j = 1,\dots,n, \tag{7}$$

$$(B_{H_1})_{ij} = \mathcal{G}^x h_j^1(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}_j^1}}, \qquad i = 1,\dots,M,\ j = 1,\dots,n, \tag{8}$$

$$O_{ij} = 0, \qquad i,j = 1,\dots,M. \tag{9}$$

G_1^x is the $n \times n$ local matrix of overlapped RBF-FD weights for the operator $\mathcal{G}^x$, with each column containing the RBF-FD weights for a point $\boldsymbol{y} \in P_1$. Assuming H_1 has full column rank, the linear system (4) has a unique solution. The above procedure can be

[1] One could also (equivalently) describe the procedure for computing $(w^x)_j(\boldsymbol{y})$ as interpolation of some function at the nodes $\boldsymbol{x}$, followed by evaluation at the nodes $\boldsymbol{y}$ [40].

repeated with the operators $\mathcal{G}^y$ and $\mathcal{G}^z$ to obtain the local differentiation matrices G_1^y and G_1^z. By construction, the *columns* of G_1^x and its counterparts populate the rows of the global differentiation matrices G^x, G^y, and G^z; for an assembly algorithm, see Algorithm 1 in [36]. As described above, the RBF-FD surface gradient weights are computed for every point in the stencil. This allows one to compute RBF-FD weights for the local surface Laplacian as

$$L_1 = \left(G_1^x \tilde{G}_1^x\right)^T + \left(G_1^y \tilde{G}_1^y\right)^T + \left(G_1^z \tilde{G}_1^z\right)^T, \tag{10}$$

where $\tilde{G}_1^x$, $\tilde{G}_1^y$, and $\tilde{G}_1^z$ contain only the columns of G_1^x, G_1^y, and G_1^z corresponding to the nodes at which weights are desired (some small neighborhood around the stencil center) [38].

The local stencil size n, the polyharmonic spline order m, the polynomial degree ℓ, and the corresponding polynomial space dimension M are not specified above. Our selection for these parameters is done mostly as in [36, 38]. We set $m = 2\ell + 1$ and set ℓ to $\ell = \xi + \theta - 1$, where θ is the order of the differential operator and ξ is the desired order of approximation. Once ℓ is computed, we use $M = \binom{\ell+d}{d}$, where d is the dimension of the ambient space. Next, the stencil size is chosen to be $n = 2M + 1$ for advection problems [38] and $n = 2M + \lfloor \log(2M) \rfloor$ for mixed PDEs [36], with d the dimension of the ambient space (in this paper, $d = 3$).

The entire procedure above must be performed for each stencil; two practical issues that arise in such a procedure are that (a) the columns of H_1 may not be linearly independent and result in lack of uniqueness for the system (4) and that (b) computing the weights by looping over each stencil can be quite expensive even if only performed once (despite its formal $O(N)$ computational complexity). The next two sections make modifications to the above procedure that mitigate these two issues.

2.2. The LOI-generated polynomial basis. Standard RBF-FD discretizations on manifolds can result in matrices H_1 in (4) that are rank-deficient, making (4) a noninvertible system. The cause of this issue is that if X lies on a manifold, then local stencils often lie on an approximate algebraic variety, which destroys polynomial unisolvency on that stencil.

The RBF-LOI procedure in [38] circumvents this issue by numerically constructing a tailored polynomial space on each stencil in a way that ensures unisolvency. The main idea is to start with the total polynomial space of degree ℓ described in the previous section and then perform adaptation on this space by identifying (numerical) rank deficiencies. These numerical rank deficiencies are eliminated to within a tunable tolerance τ by removing or adding certain basis elements. We give precise choices for τ in section 5 detailing our numerical results and in Algorithm 1.

We refer the reader to [26, 38] for a more complete algorithm description. In brief, the algorithm takes as input a local stencil (e.g., P_1) and an a priori determined orthonormal polynomial basis and outputs a polynomial space (along with new basis functions) ensuring unisolvency. The main workhorse is the LOI algorithm [26], which involves only standard numerical linear algebraic factorization operations and detects rank deficiency via the parameter τ. The input orthonormal basis is chosen as the tensor-product Chebyshev polynomials on the Euclidean bounding box of stencil P_1. The LOI algorithm outputs the basis functions $\{h_j^1(\cdot)\}_{j=1}^M$, which are subsequently used in (2) and (4). We have observed that use of this procedure eliminates solvability issues for (4).

2.3. Overlapped RBF-FD. Computing standard RBF-FD weights for each stencil k, $k = 1, \ldots, N$, can be very computationally expensive. An overlap strategy pioneered in [35] substantially reduces this heavy computational cost. We briefly review this strategy here. As with section 2.1, we specialize our discussion and our notation to stencil $k = 1$ and note that generalizations can be performed by replacing various sub/superscripts "1" with "k".

Let $\delta \in (0, 1]$ denote the *overlap parameter*, and define the *stencil retention distance* as

$$\rho_1 = (1 - \delta) \max_{j=1,\ldots,n} \|\boldsymbol{x}_{\mathcal{I}_1^1} - \boldsymbol{x}_{\mathcal{I}_j^1}\|, \tag{11}$$

where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^3$. The parameter ρ_1 defines the radius of the ball $\mathbb{B}_1$ centered at the node $\boldsymbol{x}_{\mathcal{I}_1^1}$. Let r_1 denote the number of nodes in P_1 that lie in $\mathbb{B}_1$ and R_1 contain the set of global indices of these r_1 nodes from P_1. Then the overlapped RBF-FD method involves computing RBF-FD weights for *all* the nodes whose indices are in R_1. Note that we use R_1 to determine the columns of G_x^1, G_y^1, and G_z^1 to use to form $\tilde{G}_1^x$, $\tilde{G}_1^y$, and $\tilde{G}_1^z$ in (10).

The overlapped RBF-FD method makes one important modification to the standard RBF-FD method: To avoid computing multiple sets of RBF-FD weights for a node $\boldsymbol{x}_k$, we also require that weights computed for some node $\boldsymbol{x}_k$ never be recomputed by some other stencil $P_i, i \neq k$. Let N_δ denote the total number of stencils. For a quasi-uniform node set X, $N_\delta = \frac{N}{p}$, where $p = \max\left((1 - \delta)^d n, 1\right)$ and d is the dimension (in the above discussion, $d = 3$). If $\delta = 1$, then this gives us $N_\delta = N$, recovering the standard RBF-FD method. However, if $\delta < 1$, then $N_\delta \ll N$, giving a significant speedup over the standard RBF-FD method. For a more detailed complexity analysis, see [35]. Given the polynomial degree ℓ described in section 2.1, we choose the overlap parameter δ as follows:

$$\delta = \begin{cases} 0.7 & \text{if } \ell \leq 4, \\ 0.5 & \text{if } 4 < \ell \leq 6, \\ 0.4 & \text{if } \ell > 6. \end{cases} \tag{12}$$

We find that these values of δ result in stable differentiation matrices while also facilitating the rapid assembly of these matrices.

3. Hyperviscosity formulations on manifolds. This section describes our new hyperviscosity formulation for manifolds. First, we review our Euclidean formulation in section 3.1. Next, we present our new manifold hyperviscosity formulation for the surface advection equation (section 3.2) and the surface advection-diffusion equation (section 3.3). We then discuss computing growth exponents for our model of spurious growth in section 3.5. Finally, we conclude by discussing both the selection of γ_2 in the $\gamma_1 \Delta_{\mathbb{M}}^{\gamma_2}$ hyperviscosity term (section 3.6) and the numerical approximation of the operator itself (section 3.7). Our methodology is summarized in Algorithm 1, which uses the surface advection-diffusion equation as an example but also specifies choices for pure surface advection. In Algorithm 1, an underlined quantity (such as $\underline{C}$ or $\underline{U^x}$) refers to an evaluation of the corresponding scalar field on a node set X, resulting in a vector (an array) of length N. In addition, Algorithm 1 uses the $\circ$ notation to denote the elementwise (or Hadamard) product between underlined quantities.

3.1. Review of Euclidean hyperviscosity formulations. We now review the hyperviscosity formulation for Euclidean domains $\Omega \in \mathbb{R}^d$ developed in [36]. Let

Algorithm 1 Hyperviscosity-stabilized RBF-FD discretizations on manifolds

Given: $X = \{\boldsymbol{x}_j\}_{j=1}^N$, a set of nodes on the manifold.
Given: h, the average separation distance between nodes.
Given: ξ, the desired order of approximation of the numerical method.
Given: ν, the surface diffusion coefficient.
Given: $\boldsymbol{u}$, a velocity field tangent to the surface at every point.
Given: $c_0(\boldsymbol{x}) = c(\boldsymbol{x}, 0)$, an initial condition.
Generate: $\underline{C} \approx c(\boldsymbol{x}, t)|_X$, the numerical solution to $\frac{\partial c}{\partial t} + \nabla_{\mathbb{M}} \cdot (c\boldsymbol{u}) = \nu \Delta_{\mathbb{M}} c$.

Set the polynomial degree to $\ell = \xi + 1$ if $\nu \neq 0$, otherwise use $\ell = \xi$.
Set the PHS RBF exponent to $m = 2\ell + 1$.
Let $M = \binom{\ell+d}{d}$, where d is the spatial dimension.
Set the stencil size to $n = 2M + \lfloor \ln 2M \rfloor$ if $\nu \neq 0$, otherwise set $n = 2M + 1$.
Set the hyperviscosity exponent to $\gamma_2 = \lfloor \ln n \rfloor$ if solution is smooth, otherwise set $\gamma_2 = 2$.
if $\nu \neq 0$ **then**
    Set the LOI tolerance to $\tau = 10^{-3}$ if $\ell \leq 4$, else set $\tau = 10^{-4}$.
else
    Set the LOI tolerance to $\tau = 0.05$ if $\ell < 4$, $\tau = 10^{-3}$ if $\ell \in [4, 5]$, or $\tau = 10^{-4}$ if $\ell > 5$.
end if
Set the overlap parameter δ according to (12).
Compute local differentiation matrices G_j^x, G_j^y, and G_j^z using (4) and its counterparts.
Compute local differentiation matrices L_j using (10).
Assemble local differentiation matrices into (sparse) global differentiation matrices G^x, G^y, G^z, and L using Algorithm 1 from [36].
Compute (sparse) global differentiation matrix for hyperviscosity operator as $H = L^{\gamma_2}$.
Estimate τ_x, τ_y, and τ_z, the real parts of eigenvalues with largest real parts for G^x, G^y, and G^z, respectively, as described in section 3.5.
Estimate growth exponents q_x, q_y, and q_z using (52) and its counterparts.
Estimate $\eta(\boldsymbol{x})$ using (34), then estimate $\bar{\eta}$ using (36).
Evaluate components of $\boldsymbol{u}$ (u^x, u^y, and u^z) on X to obtain $\underline{U}^x$, $\underline{U}^y$, and $\underline{U}^z$.
Compute hyperviscosity coefficient γ_1 using (35) if $\nabla_{\mathbb{M}} \cdot \boldsymbol{u} = 0$, else use (48).
Solve

$$\frac{\partial \underline{C}}{\partial t} + G^x \left(\underline{U}^x \circ \underline{C}\right) + G^y \left(\underline{U}^y \circ \underline{C}\right) + G^z \left(\underline{U}^z \circ \underline{C}\right) = \nu L \underline{C} + \gamma_1 H \underline{C} \tag{13}$$

using a suitable time integration scheme (typically an IMEX method [2]).

$c(x, t)$ be a scalar field satisfying the linear advection equation

$$\frac{\partial c}{\partial t} + u \frac{\partial c}{\partial x} = 0. \tag{14}$$

Let $c(x, t) = \hat{c}(t) e^{i\hat{k}x}$, where $\hat{k}$ is the wavenumber. In addition, let $c^{n+1} = c(t_{n+1}, x)$ and $c^n = c(t_n, x)$. Using the traditional von Neumann analysis framework, we write $c^{n+1} = \varrho c^n$, where ϱ is an amplification/growth factor and $|\varrho| \leq 1$ is necessary for time

stability. Consider a forward Euler[2] discretization of (14), with the above relations substituted in. This yields

$$\frac{\varrho - 1}{\triangle t} + ui\hat{k} = 0. \tag{15}$$

This equation of course shows that $|\varrho| > 1$ for forward Euler, indicating the need for stabilization or alternative time discretizations. However, the problem can actually be more severe after spatial discretization. In the case of RBF-FD discretizations, the *numerical* gradient typically introduces a spurious growth mode into the solution. Consider the *auxiliary derivative* (an analytical analogue to the numerical gradient) by its action on plane waves as

$$\frac{\tilde{\partial} c}{\partial x} = (i\hat{k} - \tau_x \hat{k}^{q_x}) e^{i\hat{k}x},$$

which assumes that $u > 0$. Here, $\tau_x \in \mathbb{R}$ can be thought of as the real part of the eigenvalue with the largest real part in the spectrum of the RBF-FD differentiation matrix. We focus on the case where $\tau_x > 0$ since $\tau_x \leq 0$ does not correspond to spurious growth. The quantity q_x is a growth exponent that can be estimated analytically when τ_x is known. Consequently, in the fully discretized setting, (15) is transformed into

$$\frac{\varrho - 1}{\triangle t} + ui\hat{k} = u\tau_x \hat{k}^{q_x}. \tag{16}$$

Our goal now is to eliminate the right-hand side of (16) to nullify the spurious growth mode. This can be done by adding a constant times the power of the Laplacian to the right-hand side of (14), i.e., $\gamma_1 \Delta^{\gamma_2}$. Substituting plane waves into the modified equation and dividing out the exponential gives the following equation for the amplification factor

$$\frac{\varrho - 1}{\triangle t} + ui\hat{k} = u\tau_x \hat{k}^{q_x} + \gamma_1 (-1)^{\gamma_2} \hat{k}^{2\gamma_2}. \tag{17}$$

From this, one obtains a formula for γ_1:

$$\gamma_1 = (-1)^{1-\gamma_2} u\tau_x \hat{k}^{q_x - 2\gamma_2}.$$

On a given node set of spacing h, the largest wavenumber that can be represented is $\hat{k} = 2h^{-1}$. This substitution yields a simple formula for γ_1:

$$\gamma_1 = (-1)^{1-\gamma_2} u\tau_x 2^{q_x - 2\gamma_2} h^{2\gamma_2 - q_x}.$$

For Euclidean domains $\Omega \subset \mathbb{R}^3$, the advection equation is given by

$$\frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \nabla c = 0, \tag{18}$$

assuming that $\nabla \cdot \boldsymbol{u} = 0$. We now have three differentiation matrices for the gradient (G^x, G^y, and G^z respectively). Each of these has its own spurious growth factor τ_x, τ_y,

[2]The reader may find it odd that the discussion uses forward Euler, which is unstable on (14) even in the absence of spurious growth modes. However, this was chosen purely for simplicity of analysis and to demonstrate how one would cancel the spurious growth modes. In section 3.2, we extend the analysis to more commonly used time integrators.

and τ_z respectively, and the corresponding growth exponents q_x, q_y, and q_z. Taking into account these generalizations, we obtain a new formula for γ_1:

(19)
$$\gamma_1 = (-1)^{1-\gamma_2} 3^{-\gamma_2} \left(u_x \tau_x 2^{q_x - 2\gamma_2} h^{2\gamma_2 - q_x} + u_y \tau_y 2^{q_y - 2\gamma_2} h^{2\gamma_2 - q_y} + u_z \tau_z 2^{q_z - 2\gamma_2} h^{2\gamma_2 - q_z} \right).$$

If one desires a precomputed/fixed formula for γ_1, it may be convenient to approximate the above formula by

(20)
$$\gamma_1 \approx (-1)^{1-\gamma_2} 3^{-\gamma_2} \|\boldsymbol{u}\| \left(\tau_x 2^{q_x - 2\gamma_2} h^{2\gamma_2 - q_x} + \tau_y 2^{q_y - 2\gamma_2} h^{2\gamma_2 - q_y} + \tau_z 2^{q_z - 2\gamma_2} h^{2\gamma_2 - q_z} \right).$$

The above formula assumes that the node spacing h is uniform along all the spatial directions, but this restriction can easily be removed. For quasi-uniform nodes on some domain $\Omega \subset \mathbb{R}^d$, we set $h = N^{-1/d}$, where N is the number of nodes used for discretization. On the other hand, if the nodes are instead scattered, we set h to be the smallest distance between all pairs of nodes. This formula is slightly more general than the formula given in [36], but as outlined in that work, the exponents q_x, q_y, and q_z can be estimated analytically on a given node once the spurious eigenvalues have been estimated. This analysis carries over to a wide class of explicit, implicit, and implicit-explicit time integrators. Unfortunately, it does not necessarily carry over to manifolds.

3.2. Hyperviscosity for the surface advection equation. We turn our attention to the *surface* advection equation,

$$\frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \nabla_{\mathbb{M}} c = 0, \tag{21}$$

where we assume for simplicity of analysis that $\nabla_{\mathbb{M}} \cdot \boldsymbol{u} = 0$. We discuss the case of $\nabla_{\mathbb{M}} \cdot \boldsymbol{u} \neq 0$ in section 3.4.2. Once again, the RBF-FD discretization of the surface gradient operator can be modeled by the following *auxiliary PDE*:

$$\frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \tilde{\nabla}_{\mathbb{M}} c = 0, \tag{22}$$

where the *auxiliary surface gradient* $\tilde{\nabla}_{\mathbb{M}}$ is defined by its action on $c(\boldsymbol{x}, t)$ as

$$\tilde{\nabla}_{\mathbb{M}} = [\tilde{\mathcal{G}}^x, \tilde{\mathcal{G}}^y, \tilde{\mathcal{G}}^z]^T,$$

which can be written out componentwise as

$$\tilde{\mathcal{G}}^x c = (\mathcal{G}^x - \tau_x \hat{k}_x^{q_x}) c, \qquad \tilde{\mathcal{G}}^y c = (\mathcal{G}^y - \tau_y \hat{k}_y^{q_y}) c, \qquad \tilde{\mathcal{G}}^z c = (\mathcal{G}^z - \tau_z \hat{k}_z^{q_z}) c. \tag{23}$$

Here, $\hat{k}_x$, $\hat{k}_y$, and $\hat{k}_z$ are wavenumbers. Substituting this definition of the auxiliary surface gradient into (22) gives

$$\frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \nabla_{\mathbb{M}} c = (\boldsymbol{u} \cdot \boldsymbol{\tau}) c, \qquad \boldsymbol{\tau} \coloneqq [\tau_x \hat{k}^{q_x}, \tau_y \hat{k}^{q_y}, \tau_z \hat{k}^{q_z}]^T. \tag{24}$$

Our approach to cancel out the spurious term on the right is to use artificial hyperviscosity based on the surface Laplacian $\Delta_{\mathbb{M}}$ since the RBF-FD discretization of this operator appears to be globally stable on quasi-uniform node sets (empirically speaking) [22, 38, 40]. This gives the following PDE:

$$\frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \nabla_{\mathbb{M}} c = (\boldsymbol{u} \cdot \boldsymbol{\tau}) c + \gamma_1(\boldsymbol{x}) \Delta_{\mathbb{M}}^{\gamma_2} c, \tag{25}$$

where $\gamma_1(\boldsymbol{x})$ is some undetermined function which will eventually be reduced to a single constant γ_1.

3.2.1. Explicit Runge–Kutta (RK) methods. We will now present a hyperviscosity formulation applicable to all explicit s-stage RK methods. The growth factor ϱ for an explicit s-stage RK method takes the form of the degree-s Taylor polynomial

$$\varrho(z) = \sum_{j=0}^{s} \frac{z^j}{j!}.$$

The value of z depends on the equation being solved. To see this, we will first examine the forward Euler method applied to (25). First, we observe that the growth factor $\varrho(z)$ for forward Euler (RK1) is given by

$$\varrho(z) = 1 + z. \tag{26}$$

We now substitute a plane wave of the form $e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}$ into (25), where $\hat{\mathbf{k}} = [\hat{k}_x, \hat{k}_y, \hat{k}_z]^T$ is the vector of wavenumbers, and use the growth factor for forward Euler. This gives us an equation of the form

$$\frac{\varrho - 1}{\triangle t}\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} + \boldsymbol{u}^n \cdot \nabla_{\mathbb{M}}\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} = (\boldsymbol{u}^n \cdot \boldsymbol{\tau})\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} + \gamma_1(\boldsymbol{x})\Delta_{\mathbb{M}}^{\gamma_2}\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}. \tag{27}$$

To solve the above equation for ϱ, we first need to simplify the $\boldsymbol{u}^n \cdot \nabla_{\mathbb{M}}\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}$ term. This can be written as

$$\boldsymbol{u}^n \cdot \nabla_{\mathbb{M}}\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} = \boldsymbol{u}^n \cdot i \begin{bmatrix} (1-n_x^2)k_x - n_x n_y k_y - n_x n_z k_z \\ -n_y n_x k_x + (1-n_y^2)k_y - n_y n_z k_z \\ -n_z n_x k_x - n_z n_y k_y + (1-n_z^2)k_z \end{bmatrix} \hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} = (\boldsymbol{u} \cdot Pi\mathbf{k})\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}. \tag{28}$$

This simplification gives us

$$\frac{\varrho - 1}{\triangle t}\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} + (\boldsymbol{u}^n \cdot Pi\mathbf{k})\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} = (\boldsymbol{u}^n \cdot \boldsymbol{\tau})\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} + \gamma_1(\boldsymbol{x})\Delta_{\mathbb{M}}^{\gamma_2}\hat{c}e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}. \tag{29}$$

In the Euclidean case $\mathbb{M} \equiv \mathbb{R}^d$, we could now divide out the plane wave $e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}$ to obtain an expression for γ_1 since the plane wave is an eigenfunction of the $\mathbb{R}^d$ Laplacian. Unfortunately, for submanifolds $\mathbb{M} \subset \mathbb{R}^d$, plane waves are *not* eigenfunctions of the surface Laplacian $\Delta_{\mathbb{M}}$. If we were to instead use the $\mathbb{R}^3$ Laplacian Δ in place of $\Delta_{\mathbb{M}}$, we could simply use (20). However, it may in general be difficult to compute a reasonable discretization of Δ when the only points available to us are on the manifold $\mathbb{M}$. While this has been attempted for nonpolynomially augmented RBF-FD [12, 16], our attempt at doing so using the RBF-LOI procedure in section 2 resulted in an approximation to Δ on the manifold that had positive real eigenvalues.

Our proposed approach to solving for $\gamma_1(\boldsymbol{x})$ is to approximate the action of $\Delta_{\mathbb{M}}^{\gamma_2}$ on plane waves as a function of the action of Δ^{γ_2}. More specifically, we write

$$\Delta_{\mathbb{M}}^{\gamma_2} e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} \approx \eta(\boldsymbol{x})\Delta^{\gamma_2} e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} = \eta(\boldsymbol{x})(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2} e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}, \tag{30}$$

where $\eta(\boldsymbol{x})$ is some unknown function. This approach allows us to treat plane waves as eigenfunctions of the surface Laplacian, overcoming the primary hurdle in computing $\gamma_1(\boldsymbol{x})$. We can now use (30) to solve for ϱ in (29), giving

$$\varrho \approx 1 - \triangle t(\boldsymbol{u}^n \cdot Pi\hat{\mathbf{k}}) + \triangle t(\boldsymbol{u}^n \cdot \boldsymbol{\tau}) + \triangle t\gamma_1\eta(\boldsymbol{x})(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}.$$

Comparing the above to (26) gives us an expression for z:

$$z = -\triangle t(\boldsymbol{u}^n \cdot Pi\hat{\mathbf{k}}) + \triangle t(\boldsymbol{u}^n \cdot \boldsymbol{\tau}) + \triangle t\gamma_1\eta(\boldsymbol{x})(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}.$$

For an s-stage explicit RK method, we therefore have

(31)
$$\varrho(z) = \sum_{j=0}^{s} \frac{1}{j!}\left(-\triangle t(\boldsymbol{u}^n \cdot Pi\hat{\mathbf{k}}) + \triangle t(\boldsymbol{u}^n \cdot \boldsymbol{\tau}) + \triangle t\gamma_1\eta(\boldsymbol{x})(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}\right)^j.$$

We now split z into two components, $z = z_1 + z_2$, each associated with different terms in the PDE:

$$z_1 \coloneqq -\triangle t(\boldsymbol{u}^n \cdot Pi\hat{\mathbf{k}}), \quad z_2 \coloneqq \triangle t(\boldsymbol{u}^n \cdot \boldsymbol{\tau}) + \triangle t\gamma_1\eta(\boldsymbol{x})(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}.$$

The term z_1 is that obtained for the semidiscrete surface advection equation without growth or hyperviscosity, while z_2 captures the growth and hyperviscosity terms. In the absence of spurious growth and hyperviscosity, we have the following identity:

$$\varrho(z) = \varrho(z_1) = \sum_{j=0}^{s} \frac{z_1^j}{j!} = \frac{e^{z_1}\Gamma(s+1, z_1)}{s!}, \tag{32}$$

where $\Gamma(a, b)$ is the upper incomplete gamma function. However, if we allow growth and hyperviscosity, we have

$$\varrho(z) = \varrho(z_1, z_2) = \sum_{j=0}^{s} \frac{(z_1 + z_2)^j}{j!} = \frac{e^{z_1+z_2}\Gamma(s+1, z_1+z_2)}{s!}. \tag{33}$$

Subtracting (32) from (33) defines a term $\tilde{\varrho}$:

$$\tilde{\varrho} \coloneqq \varrho(z_1, z_2) - \varrho(z_1) = \frac{e^{z_1}\left(e^{z_2}\Gamma(s+1, z_1+z_2) - \Gamma(s+1, z_1)\right)}{s!}.$$

Clearly $\tilde{\varrho}$ is the contribution of growth and hyperviscosity to the growth factor $\varrho(z)$. If $\tilde{\varrho} \equiv 0$, we would recover the growth factor for the explicit s-stage RK method on the pure semidiscrete surface advection equation. $\tilde{\varrho}$ can be zeroed out by setting $z_2 = 0$, which in turn implies

$$\begin{aligned} &\triangle t(\boldsymbol{u}^n \cdot \boldsymbol{\tau}) + \triangle t\gamma_1\eta(\boldsymbol{x})(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2} = 0, \\ &\implies \gamma_1(\boldsymbol{x}) = \frac{(-1)^{1-\gamma_2}\boldsymbol{u} \cdot \boldsymbol{\tau}}{\eta(\boldsymbol{x})\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}}. \end{aligned}$$

Using the definition of $\boldsymbol{\tau}$ in (24) and making the simplifying assumption $\hat{k} = \hat{k}_x = \hat{k}_y = \hat{k}_z = 2h^{-1}$ yields

$$\gamma_1(\boldsymbol{x}) = \frac{(-1)^{1-\gamma_2}3^{-\gamma_2}}{\eta(\boldsymbol{x})}\left(u_x\tau_x 2^{q_x-2\gamma_2}h^{2\gamma_2-q_x} + u_y\tau_y 2^{q_y-2\gamma_2}h^{2\gamma_2-q_y} + u_z\tau_z 2^{q_z-2\gamma_2}h^{2\gamma_2-q_z}\right).$$

To allow for precomputation, we once again use the 2-norm instead:

$$\gamma_1(\boldsymbol{x}) \approx \frac{(-1)^{1-\gamma_2} 3^{-\gamma_2} \|\boldsymbol{u}\|}{\eta(\boldsymbol{x})} \left(\tau_x 2^{q_x-2\gamma_2} h^{2\gamma_2-q_x} + \tau_y 2^{q_y-2\gamma_2} h^{2\gamma_2-q_y} + \tau_z 2^{q_z-2\gamma_2} h^{2\gamma_2-q_z}\right).$$

The function $\eta(\boldsymbol{x})$ is in general not known exactly, but it can be approximated on a given node set X on the manifold $\mathbb{M}$ as follows: Let L be the approximation to the surface Laplacian computed as in section 2. Then we can write

$$\eta(\boldsymbol{x}_j) = \frac{\left(L^{\gamma_2}\, e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}\Big|_X\right)_j}{\left((-1)^{\gamma_2}\left(\hat{k}_x^2+\hat{k}_y^2+\hat{k}_z^2\right)^{\gamma_2} e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}\Big|_X\right)_j}, \qquad j = 1,\dots,N, \tag{34}$$

where N is the total number of nodes. Here, $\hat{k}_x$, $\hat{k}_y$, and $\hat{k}_z$ each are set to $2h^{-1}$. If the nodes are quasi-uniform, we set $h = N^{-1/2}$ for manifolds $\mathbb{M} \subset \mathbb{R}^d$ of codimension 1. Finally, we define a single constant γ_1 as

$$\gamma_1 = \mathbb{E}_X\left(\gamma_1(\boldsymbol{x})\right),$$

which is to be read as the "real-valued expectation/mean of $\gamma_1(\boldsymbol{x})$ over the node set X." This can in turn be written as

$$\gamma_1 = \frac{(-1)^{1-\gamma_2} 3^{-\gamma_2} \|\boldsymbol{u}\|}{\bar{\eta}} \left(\tau_x 2^{q_x-2\gamma_2} h^{2\gamma_2-q_x} + \tau_y 2^{q_y-2\gamma_2} h^{2\gamma_2-q_y} + \tau_z 2^{q_z-2\gamma_2} h^{2\gamma_2-q_z}\right), \tag{35}$$

where $\bar{\eta}$ is designed to be real-valued and positive, given explicitly by

$$\bar{\eta} = \frac{1}{N}\sum_{j=1}^{N} \left(|\operatorname{Re}(\eta(\boldsymbol{x}_j))|\right). \tag{36}$$

Though we have made a series of simplifying assumptions, we now have a simple and computable formula for γ_1 using (35) (and (36)). As we have shown, this formula is applicable to all s-stage–explicit RK methods and any linear methods whose growth factor is expressible as the series (3.2.1).

3.2.2. Explicit linear multistep methods. Explicit linear multistep methods (LMMs) are a popular class of methods for solving time-dependent problems. While their stability regions are not as large as the corresponding explicit RK methods, explicit LMMs are still competitive from the computational cost standpoint due to fewer function evaluations. We will now verify the hyperviscosity formulation in the context of these methods.

Before proceeding, we remark that (35) (derived for RK methods) could be obtained if we had directly set the approximation for $\Delta_{\mathbb{M}}^{\gamma_2}$ to be

$$\Delta_{\mathbb{M}}^{\gamma_2} e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} \approx \bar{\eta}(-1)^{\gamma_2}\left(\hat{k}_x^2+\hat{k}_y^2+\hat{k}_z^2\right)^{\gamma_2} e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}. \tag{37}$$

It is also useful to define an analogous map for the surface Laplacian itself,

$$\Delta_{\mathbb{M}} e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} \approx -\bar{\omega}\left(\hat{k}_x^2+\hat{k}_y^2+\hat{k}_z^2\right) e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}, \tag{38}$$

where $\bar{\omega}$ is computed analogously to $\bar{\eta}$. We use both these new maps going forward. Unlike the explicit RK methods, deriving the growth factor for explicit LMMs is a slightly more elaborate procedure. the generic ODE

$$\frac{\partial c}{\partial t} = \mu c.$$

The s-step–explicit LMM (the Adams–Bashforth method of order s) applied to this ODE gives

$$c^{n+1} = c^n + \triangle t \sum_{j=0}^{s-1} \alpha_j \mu c^{n-j},$$

where α_j are some real-valued coefficients. Now, let $c^{n-s+1} = \hat{c} e^{i\hat{k}\cdot\boldsymbol{x}}$ and $c^{n-s+2} = \varrho c^{n-s+1}$, where ϱ is the growth factor. Substituting into the s-step Adams–Bashforth method, we obtain a polynomial equation for the growth factor G:

$$\varrho^s - \varrho^{s-1}\left(1 + \alpha_0 \triangle t\mu\right) - \varrho^{s-2}\alpha_1 \triangle t\mu - \varrho^{s-3}\alpha_2 \triangle t\mu - \cdots - \alpha_{s-1}\triangle t\mu = 0. \tag{39}$$

In the context of the surface advection equation with spurious growth and hyperviscosity, we have

$$\begin{aligned} \triangle t\mu &= z_1 + z_2, \\ z_1 &= -\triangle t(\boldsymbol{u}^n \cdot P\mathbf{k}), \\ z_2 &= \triangle t(\boldsymbol{u}^n \cdot \boldsymbol{\tau}) + \triangle t\gamma_1\bar{\eta}(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}. \end{aligned}$$

Substituting the above equations into (39) gives

$$\begin{aligned} &\varrho^s - \varrho^{s-1}\left(1 + \alpha_0 z_1 + \alpha_0 z_2\right) \\ &\quad - \varrho^{s-2}\alpha_1(z_1 + z_2) - \varrho^{s-3}\alpha_2(z_1 + z_2) - \cdots - \alpha_{s-1}(z_1 + z_2) = 0. \end{aligned}$$

We can collect all the z_1 and z_2 terms separately to yield

$$\begin{aligned} &\left[\varrho^s - \varrho^{s-1}\left(1 + \alpha_0 z_1\right) - \varrho^{s-2}\alpha_1 z_1 - \cdots - \alpha_{s-1} z_1\right] \\ &\quad - z_2\left[\varrho^{s-1}\alpha_0 + \varrho^{s-2}\alpha_1 + \cdots + \alpha_{s-1}\right] = 0. \end{aligned}$$

The first bracketed term above is the equation for the growth factor ϱ on the pure semidiscrete surface advection equation, while the second bracketed term accounts for growth and hyperviscosity. To cancel out the latter, we either require the growth factor ϱ to satisfy the polynomial $\varrho^{s-1}\alpha_0 + \varrho^{s-2}\alpha_1 + \cdots + \alpha_{s-1} = 0$ or can simply set $z_2 = 0$. Doing the latter yields

$$\triangle t(\boldsymbol{u}^n \cdot \boldsymbol{\tau}) + \triangle t\gamma_1\bar{\eta}(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2} = 0,$$

which results in the same expression for γ_1 as in the case of explicit RK methods, once again matching (35).

3.3. Hyperviscosity for the surface advection-diffusion equation. We now consider the surface advection-diffusion equation defined as

$$\frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \nabla_{\mathbb{M}} c = \nu \Delta_{\mathbb{M}} c.$$

This is a PDE of mixed character, with the ratio $\frac{\|\boldsymbol{u}\|}{\nu}$ determining whether the transport of c is dominated by advection or diffusion (over some characteristic length scale). In regimes where both advection and diffusion play an important role in the transport, numerical solution of the advection-diffusion equation is often done via an implicit-explicit (IMEX) method [2]. These methods allow us to advance the stiff diffusion term implicitly in time while advancing the advection term explicitly. When considering the *stabilized* auxiliary surface advection-diffusion equation,

$$\frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \tilde{\nabla}_{\mathbb{M}} c = \nu \Delta_{\mathbb{M}} c + \gamma_1 \Delta_{\mathbb{M}}^{\gamma_2} c,$$

we must now decide whether to treat the hyperviscosity term explicitly or implicitly in time. If the problem contains enough diffusion to impact transport but insufficient diffusion to stabilize the auxiliary surface gradient, it may be reasonable to add a hyperviscosity term and treat it implicitly along with the stiff diffusion term (for efficiency and stability). While our formula for γ_1 is appropriate for the explicit treatment of the hyperviscosity term, we have yet to show whether this formula is sufficient for stability if the hyperviscosity term is treated implicitly. We will focus on a popular IMEX time integrator: the semi-implicit backward difference formula of order 2 (SBDF2) [2]. First, consider a generic ODE of the form

$$\frac{dc}{dt} = \mu_1 c + \mu_2 c.$$

If we decide to treat the μ_1 term explicitly and the μ_2 term implicitly, the SBDF2 discretization of this ODE is

$$\frac{3c^{n+1} - 4c^n + c^{n-1}}{2\triangle t} = 2\mu_1 c^n - \mu_1 c^{n-1} + \mu_2 c^{n+1}.$$

Substituting in a plane wave expression and using von Neumann analysis, we get a quadratic equation for the growth factor ϱ:

$$\left(1 - \frac{2}{3}\mu_2 \triangle t\right) \varrho^2 - \frac{4}{3}\varrho\,(1 + \triangle t \mu_1) + \frac{1}{3}\,(1 + 2\triangle t \mu_1) = 0. \tag{40}$$

In the context of the surface advection-diffusion equation, μ_1 represents the action of the surface advection and spurious growth operators on plane waves and can be written using (28) as

$$\mu_1 = -\boldsymbol{u} \cdot Pi\hat{\mathbf{k}} + \boldsymbol{u} \cdot \boldsymbol{\tau}, \tag{41}$$

where we now assume for simplicity that the velocity $\boldsymbol{u}$ is constant in time. μ_2 represents the action of the surface Laplacian and surface hyperviscosity operators on plane waves. To derive an expression for μ_2, we now also need the approximation (38), which allows us to write μ_2 as

$$\mu_2 = -\nu\bar{\omega}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right) + \gamma_1 \bar{\eta}(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}. \tag{42}$$

Substituting (41) and (42) into (40), collecting the surface advection and diffusion terms into a term t_1, and collecting the spurious growth and hyperviscosity terms

into the term t_2, we have

$$t_1 = \left(1 + \frac{2}{3}\nu\triangle t\bar{\omega}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)\right)\varrho^2 - \frac{4}{3}\varrho\left(1 - \boldsymbol{u}\cdot Pi\hat{\mathbf{k}}\right) + \frac{1}{3}\left(1 - 2\boldsymbol{u}\cdot Pi\hat{\mathbf{k}}\right),$$
$$t_2 = -\frac{2}{3}\triangle t\varrho^2\gamma_1\bar{\eta}(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2} + \frac{2}{3}\triangle t\boldsymbol{u}\cdot\boldsymbol{\tau}(1 - 2\varrho),$$
$$t_1 + t_2 = 0.$$

Here, t_1 is obtained when discretizing the surface advection-diffusion equation, while t_2 contains all additional terms. We thus require that $t_2 = 0$, i.e.,

$$-\varrho^2\gamma_1\bar{\eta}(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2} + \boldsymbol{u}\cdot\boldsymbol{\tau}(1 - 2\varrho) = 0.$$

The roots of this quadratic equation are given by

$$\varrho = -\frac{\boldsymbol{u}\cdot\boldsymbol{\tau}}{\gamma_1\bar{\eta}(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}} \pm \frac{\sqrt{(\boldsymbol{u}\cdot\boldsymbol{\tau})\left(\boldsymbol{u}\cdot\boldsymbol{\tau} + \gamma_1\bar{\eta}(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}\right)}}{\gamma_1\bar{\eta}(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}}.$$

For time stability, we require that $|\varrho| \leq 1$. In addition to the CFL condition, this can be achieved if

$$\boldsymbol{u}\cdot\boldsymbol{\tau} + \gamma_1\bar{\eta}(-1)^{\gamma_2}\left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2} = 0.$$

This results in the following expression for γ_1 after substituting in the definition of $\boldsymbol{\tau}$, using the wavenumber-h relationship, and pulling out the magnitude of the velocity:

$$\gamma_1 = \frac{(-1)^{1-\gamma_2}3^{-\gamma_2}\|\boldsymbol{u}\|}{\bar{\eta}}\left(\tau_x 2^{q_x-2\gamma_2}h^{2\gamma_2-q_x} + \tau_y 2^{q_y-2\gamma_2}h^{2\gamma_2-q_y} + \tau_z 2^{q_z-2\gamma_2}h^{2\gamma_2-q_z}\right).$$

Once again, this is identical to (35). We have thus demonstrated that our formula is applicable for both explicit and implicit discretizations of the hyperviscosity operator. The above analysis carries over to higher-order SBDF methods, though it is more tedious and requires computing the roots of cubic and higher-degree polynomial equations in ϱ.

3.4. Additional considerations.

3.4.1. On stabilization from model diffusion. In the setting of the surface advection-diffusion equation, it is reasonable to wonder how much model diffusion is required to cancel out the spurious growth modes in the discretized advection operator. Fortunately, our framework for computing γ_1 also allows us to estimate a lower bound on the required amount of stabilizing model diffusion. Consider the auxiliary surface advection-diffusion equation given by

$$\frac{\partial c}{\partial t} + \boldsymbol{u}\cdot\tilde{\nabla}_{\mathbb{M}}c = \nu\Delta_{\mathbb{M}}c.$$

The goal is to estimate a lower bound on ν that allows us to cancel out spurious growth modes. Proceeding with the von Neumann analysis for SBDF2 as before, now using the map (38), and requiring that $|G| \leq 1$, we obtain

$$\nu = \frac{-\|\boldsymbol{u}\|}{3\bar{\omega}}\left(\tau_x 2^{q_x-2}h^{2-q_x} + \tau_y 2^{q_y-2}h^{2-q_y} + \tau_z 2^{q_z-2}h^{2-q_z}\right). \tag{43}$$

We illustrate the utility of this formula with an example. Assume that all quantities except ν and $\bar{\omega}$ are $O(1)$. Then $\nu \approx \frac{h}{6\bar{\omega}}$. In this setting, for a coarse node set on the sphere ($N = 2562$ icosahedral nodes), experiments show that $\bar{\omega} \approx 0.03$, which implies that $\nu \gtrsim 5N^{-1/2}$ is the least amount of model diffusion required to cancel spurious growth modes. In other words, if $\nu \gtrsim 5h$ and $\|\boldsymbol{u}\| = O(1)$, no hyperviscosity is required for stabilization. A careful analysis of the model diffusion using our technique may obviate the need for numerical hyperviscosity. However, in this work, we primarily focus on parameter regimes requiring hyperviscosity.

3.4.2. On divergent velocity fields. Our analysis thus far assumed that $\nabla_{\mathbb{M}} \cdot \boldsymbol{u} = 0$. We now deal with the more general case of a divergent velocity field. Consider now the auxiliary divergent surface advection equation (with added hyperviscosity)

$$\frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \tilde{\nabla}_{\mathbb{M}} c = -c\tilde{\nabla}_{\mathbb{M}} \cdot \boldsymbol{u} + \gamma_1 \Delta_{\mathbb{M}}^{\gamma_2} c. \tag{44}$$

Once again using the forward Euler discretization for simplicity, standard von Neumann analysis for the variable c (together with (37)) gives us

$$\frac{\varrho - 1}{\triangle t} + \boldsymbol{u}^n \cdot Pi\mathbf{k} = -\tilde{\nabla}_{\mathbb{M}} \cdot \boldsymbol{u}^n + \boldsymbol{u}^n \cdot \boldsymbol{\tau} + \gamma_1 \bar{\eta}(-1)^{\gamma_2} \left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}. \tag{45}$$

The first term in the right-hand side can be expanded using the definition of the growth model, giving us

$$\frac{\varrho - 1}{\triangle t} + \boldsymbol{u}^n \cdot Pi\mathbf{k} = -\nabla_{\mathbb{M}} \cdot \boldsymbol{u}^n + 2\boldsymbol{u}^n \cdot \boldsymbol{\tau} + \gamma_1 \bar{\eta}(-1)^{\gamma_2} \left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2}. \tag{46}$$

Note the factor of 2 in front of the $\boldsymbol{u}^n$ term. We now want γ_1 so that

$$2\boldsymbol{u}^n \cdot \boldsymbol{\tau} + \gamma_1 \bar{\eta}(-1)^{\gamma_2} \left(\hat{k}_x^2 + \hat{k}_y^2 + \hat{k}_z^2\right)^{\gamma_2} = 0. \tag{47}$$

This in turn implies that γ_1 is given as

$$\gamma_1 = \frac{2(-1)^{1-\gamma_2} 3^{-\gamma_2}}{\bar{\eta}} \left(u_x \tau_x 2^{q_x - 2\gamma_2} h^{2\gamma_2 - q_x} + u_y \tau_y 2^{q_y - 2\gamma_2} h^{2\gamma_2 - q_y} + u_z \tau_z 2^{q_z - 2\gamma_2} h^{2\gamma_2 - q_z}\right),$$

which can be made amenable to precomputation by setting

$$\gamma_1 = \frac{2(-1)^{1-\gamma_2} 3^{-\gamma_2} \|\boldsymbol{u}\|}{\bar{\eta}} \left(\tau_x 2^{q_x - 2\gamma_2} h^{2\gamma_2 - q_x} + \tau_y 2^{q_y - 2\gamma_2} h^{2\gamma_2 - q_y} + \tau_z 2^{q_z - 2\gamma_2} h^{2\gamma_2 - q_z}\right). \tag{48}$$

It is straightforward to show that this analysis applies to all explicit RK and multistep methods and the SBDF2 method for the surface advection-diffusion equation. Though we focus on divergence-free fields in this article, our experiments have shown that the above formula works well to stabilize the surface advection equation when divergent velocity fields are used.

3.5. Computing growth exponents. We now discuss how to compute the growth exponents q_x, q_y, and q_z. While this is similar to the Euclidean case from [36], the presence of a manifold adds certain complications. Let $X = \{x_j\}_{j=1}^N$ be a set of nodes on the manifold where we wish to compute the discrete differentiation matrices

G^x, G^y, and G^z. Consider the function $f(\boldsymbol{x}) = e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}$. Its analytical surface gradient is given by

$$\mathbf{g}(\boldsymbol{x}) = \nabla_{\mathbb{M}} f(\boldsymbol{x}) = Pi\hat{\mathbf{k}} f(\boldsymbol{x}) = [g^x(\boldsymbol{x}), g^y(\boldsymbol{x}), g^z(\boldsymbol{x})]^T. \tag{49}$$

Let $\underline{f}$, $\underline{g}^x$, $\underline{g}^y$, and $\underline{g}^z$ be the evaluations of $f(\boldsymbol{x})$, $g^x(\boldsymbol{x})$, $g^y(\boldsymbol{x})$, and $g^z(\boldsymbol{x})$ on the node set X. The *approximate* surface gradient of $f(\boldsymbol{x})$ on the node set X is given componentwise by matrix-multiplication with the differentiation matrices:

$$\tilde{\underline{g}}^x = G^x\underline{f},\ \tilde{\underline{g}}^y = G^y\underline{f},\ \tilde{\underline{g}}^z = G^z\underline{f}. \tag{50}$$

From our growth model, we know that G^x, G^y, and G^z are represented by the auxiliary differential operators $\tilde{\mathcal{G}}^x$, $\tilde{\mathcal{G}}^y$, and $\tilde{\mathcal{G}}^z$ respectively. Consequently, ignoring truncation errors, the growth model gives us

$$\tilde{\underline{g}}^x = \underline{g}^x - \tau_x\hat{k}_x^{q_x}\underline{f},\ \tilde{\underline{g}}^y = \underline{g}^y - \tau_y\hat{k}_y^{q_y}\underline{f},\ \tilde{\underline{g}}^z = \underline{g}^z - \tau_z\hat{k}_z^{q_z}\underline{f}. \tag{51}$$

Focusing momentarily on the x-component without loss of generality, this allows us to write

$$\|\underline{g}^x - \tilde{\underline{g}}^x\| = \tau_x\hat{k}^{q_x}\|\underline{f}\|,$$

with similar expressions for the y and z components. Since $\tau_x\|\underline{f}\| \neq 0$, we divide through by this quantity, take natural logarithms, and rearrange to obtain

$$q_x = \frac{\ln\left(\|\underline{g}^x - \tilde{\underline{g}}^x\|\right) - \ln\left(\tau_x\|\underline{f}\|\right)}{\ln\hat{k}}, \tag{52}$$

with similar expressions for q_y and q_z. The usual substitution of $\hat{k} = 2h^{-1}$ gives us specific values of q_x, q_y, and q_z for a given node set once the τ_x, τ_y, and τ_z have been computed. As mentioned previously, these represent the real part of the eigenvalue with the largest real part in the spectrum of G^x, G^y, and G^z, respectively. Notice that if we over- or underestimate these quantities, the formula (52) and its counterparts automatically under- or overestimate q_x, q_y, and q_z. This makes the formula for γ_1 robust to loose tolerances used in estimating τ_x, τ_y, and τ_z. In our experiments, we use the following iterative procedure to estimate these spurious eigenvalues:

1. attempt to estimate τ_x, τ_y, and τ_z using an implicitly restart Arnoldi method with a loose tolerance of 10^{-3}, looking for the eigenvalue with the largest real part;
2. if no eigenvalues are found to this tolerance, double the tolerance until one can be found.

3.6. Selecting γ_2. In the spectral methods literature, Ma recommends that γ_2 in $\gamma_1\Delta^{\gamma_2}$ be chosen according to the relation $\gamma_2 \leq O(\ln N)$, where N is the total number of points used in the discretization [23, 24]. In [36], our Euclidean formulation used a version of this scaling law so that $\gamma_2 = \lfloor 1.5\ln n\rfloor$, where n is the RBF-FD stencil size.

On manifolds, we set $\gamma_2 = \lfloor \ln n\rfloor$ if the solution is smooth. This allows $\gamma_1\Delta^{\gamma_2}$ to approach the spectral case as $n \to N$ and filters higher frequencies as the stencil size is increased. If the solution is not smooth, we set $\gamma_2 = 2$ and do not scale with increasing n. In our experiments, we discovered undesirable oscillations when using $\gamma_2 > 3$ for test cases with nonsmooth solutions. The rationale is simple: If the solution c is not sufficiently smooth, $\gamma_1\Delta^{\gamma_2}c$ may not even be continuous if γ_2 is large. In general, if the smoothness of the solution is not known, it may be possible to use the native space norm as a smoothness indicator [6] and set γ_2 accordingly. We leave this approach for future work.

3.7. RBF approximation to $\gamma_1\Delta_{\mathbb{M}}^{\gamma_2}$. Following the procedure outlined in section 2, we compute $\Delta_{\mathbb{M}}^{\gamma_2}$ by using RBF-LOI to obtain the local surface Laplacian L_j on each stencil. We then assemble the local L_j matrices into a global sparse matrix L for the surface Laplacian and compute the differentiation matrix for the hyperviscosity operator as $H = L^{\gamma_2}$. This approach proved successful since the spectrum of L was always well behaved in our experiments (no eigenvalues with positive real parts).[3] This is in contrast to our Euclidean formulation, where we approximated Δ^{γ_2} directly on each stencil [36]. The advantage of the new approach is that it obviates the need to separately approximate the H matrix. At first glance, the observant reader may notice that this approximation to H is potentially a very low order one (possibly only first-order accurate if γ_2 is large enough). However, since $\gamma_1 = O(h^{2\gamma_2})$, the scaling with γ_1 makes $\gamma_1 H$ a reasonable approximation to $\gamma_1\Delta_{\mathbb{M}}^{\gamma_2}$ in the sense that $\gamma_1 H$ possess similar spectral properties. We leave the tackling of potential spurious eigenvalues in L (and therefore H) to future work but note that we did not encounter this problem for any of the test cases in this article.

4. Role of node sets and parameters. Though our hyperviscosity formulation is automatic, the actual magnitude of the parameter γ_1 on a given node set is dictated by the behavior of the function $\eta(\boldsymbol{x})$ and the largest wavenumber $\hat{k} \approx 2h^{-1}$. In the case of $\eta(\boldsymbol{x})$, we advocated computing a real-valued average $\bar{\eta}$; in the case of $\hat{k}$, we approximated as $2h^{-1}$, where h was chosen as $\frac{1}{\sqrt{N}}$, an *average* measure of node spacing. We now explore the impact of these choices on stability and accuracy.

4.1. Stability on nonuniform node sets. We now explore the effect of these choices on the spectrum of the *combined* discrete differentiation matrix $G_H = -G^x - G^y - G^z + \gamma_1 H$, assuming that the velocity is constant spatially. Ideally, the spectrum of this combined differential operator should have no eigenvalues with positive real parts. While the node sets used in section 5 are all quasi-uniform, it is instructive to see the influence of nonuniformity on the value of γ_1 and hence the spectrum of G_H. First, we define three different quantities:

1. $\bar{h} = \frac{1}{\sqrt{N}}$, which we call the *average node spacing*;
2. h_q, the *separation radius*, defined as the minimal distance between any pair of points in the node set X;[4]
3. h_ρ, the *mesh norm* or the *fill distance*, defined as the radius of the largest ball that can be fit between any pair of nodes.

In our formulas for γ_1, all three of these quantities are candidates for the variable h. For $N = 2400$ (quasi-uniform) staggered nodes on a torus with inner radius $1/3$ and outer radius 1, for instance, we have $\bar{h} = 0.0204$, $h_q = 0.0629$, and $h_\rho = 0.0871$. On the other hand, when we select a random subset of size $N = 1800$ from these staggered nodes, the nodes are nonuniform, and $h_\rho = 0.1043$. Essentially, we expect h_q and h_ρ to be more different on nonuniform nodes. We set $\xi = 4$ and use Algorithm 1 to compute the different differentiation matrices. Finally, we compute the matrix G_H and show its eigenvalues when each of the measures of node spacing are used above. The results are shown in Figure 1. These results show that while all three measures of h seem reasonable, the $h = \bar{h}$ (Figure 1 (left)) is the best choice if the goal is to

[3] We also explored approximating the hyperviscosity operator locally on each stencil and then assembling the local operators, but this approach failed for larger k due to spurious positive real eigenvalues in the spectrum of the resulting differentiation matrix.

[4] h_q is referred to by the symbol q in the RBF literature and is often defined as half the value used in this work.

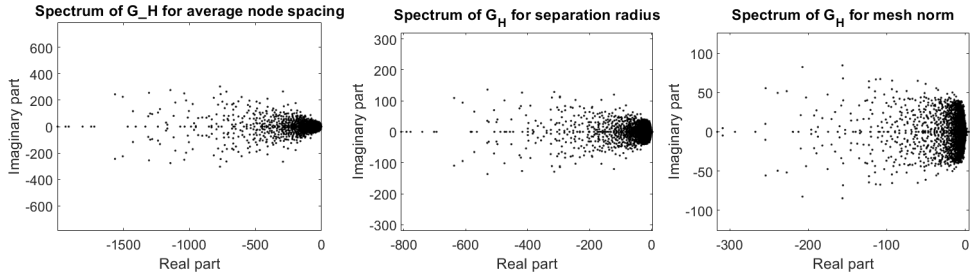


FIG. 1. *Spectrum of G_H for $N = 1800$ nonuniform nodes on the torus when using $\bar{h}$ (left), h_q (middle), and h_ρ (right) in the formula for γ_1.*

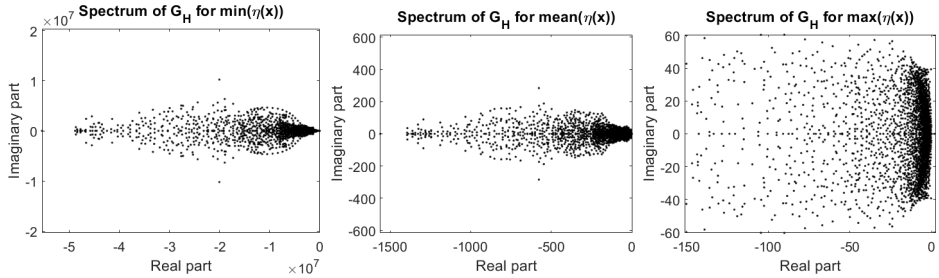


FIG. 2. *Spectrum of G_H for $N = 2400$ quasi-uniform nodes on the torus when using* $\min(\eta(\boldsymbol{x}))$ *(left), $\bar{\eta}$ (middle), and* $\max(\eta(\boldsymbol{x}))$ *(right) in the formula for γ_1.*

guarantee the absence of an eigenvalue with any positive real part in the spectrum of G_H. Though it is hard to see, Figure 1 (middle) shows that using $h = h_q$ actually results in an eigenvalue with a very small real part, and Figure 1 (right) shows that using $h = h_\rho$ results in an even larger real part. On a quasi-uniform node set, we have found that any of these measures of node spacing are appropriate since they are all closer to each other. When solving PDEs, nonuniform node sets may arise in the context of h-refinement. Such nonuniform node sets are still more uniform than the node sets used in this experiment, so our experiment represents an extreme case. For the remainder of this article, we restrict our attention to quasi-uniform node sets and use $h = \bar{h}$.

4.2. Influence of $\boldsymbol{\eta(x)}$ on stability. The formula for γ_1 contains the term $\bar{\eta}$, which is the real-valued expectation of the function $\eta(\boldsymbol{x})$. However, this choice was made simply to reduce $\eta(\boldsymbol{x})$ to a single positive number for computational convenience. It is possible to use other quantities, such as the maximum or minimum values. We now explore the influence of this choice on the spectrum of G_H. Once again, we set $\xi = 4$ and use $N = 2400$ quasi-uniform nodes on the torus. The results of the experiment are shown in Figure 2. The results are quite interesting. Using $\min(\eta(\boldsymbol{x}))$ as in Figure 2 (left) scatters the eigenvalues of G_H very far into the left half of the complex plane, both along the real and the imaginary axes (note the scale). While this differential operator is globally stable in the eigenvalue sense, our experiments indicate it is unusable for practical simulations, as it significantly reduces the accuracy of our method. Figure 2 (middle) shows that using $\bar{\eta}$ results in a much smaller scatter and has successfully shifted over eigenvalues with positive real parts. Finally, Figure 2 (right) shows that using $\max(\eta(\boldsymbol{x}))$ in fact produces the least scatter (again, note the scale) but produces an eigenvalue with a small positive real part (in this case,

≈ 1.5). While $\bar{\eta}$ is clearly the safe choice, this indicates that $\max(\eta(\boldsymbol{x}))$ can also be used in simulations if the time step Δt is such that any small positive eigenvalue falls within the stability region of the time integrator.

4.3. Effect of the analytic approximation to the eigenvalues of $\boldsymbol{\Delta_{\mathbb{M}}^{\gamma_2}}$. Our technique for deriving an analytic expression for γ_1 relied on analytically approximating the action of $\Delta_{\mathbb{M}}^{\gamma_2}$ on plane waves as a function of the action of Δ^{γ_2}. This was expressed through (30), which we repeat here for convenience:

$$\Delta_{\mathbb{M}}^{\gamma_2} e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} \approx \eta(\boldsymbol{x})\Delta^{\gamma_2} e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}} = \eta(\boldsymbol{x})(-1)^{\gamma_2}\left(\hat{k}_x^2+\hat{k}_y^2+\hat{k}_z^2\right)^{\gamma_2} e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}.$$

Here, $\eta(\boldsymbol{x})$ is computed numerically on a given node set X by (34). Our goal is to study the effect of this approximation when the eigenvalues of $\Delta_{\mathbb{M}}^{\gamma_2}$ are known. Let $f(\boldsymbol{x}) = e^{i\hat{\mathbf{k}}\cdot\boldsymbol{x}}$. To test this, we set $\gamma_2 = 1$ and compare $\bar{\Delta}_{\mathbb{M}} f = \bar{\eta}(\hat{k}_x^2+\hat{k}_y^2+\hat{k}_z^2)f$ with Lf, where L is the differentiation matrix that corresponds to $\Delta_{\mathbb{M}}$. Once again, we used staggered nodes on the torus for this experiment, with node sets of size $N = 2400, 5400, 9600, 21600$, and 38400, setting $\xi = 2, 4, 6$. We then measured the quantity $\|\left(\bar{\Delta}_{\mathbb{M}} f\right)_X - Lf\|/\|Lf\|$. We found that this was approximately constant (≈ 0.44) under both spatial refinement and order refinement with a magnitude that increased as γ_2 was increased. In fact, the quantity $\bar{\Delta}_{\mathbb{M}}^{\gamma_2} f$ appears to consistently overestimate $L^{\gamma_2} f = Hf$, indicating that our formulation for γ_1 is likely overly strict, especially as γ_2 is increased. While this is reassuring for our time stability estimates for RK and IMEX methods, we plan in future work to explore spatially variable hyperviscosity formulations and analytic derivations for γ_2 to further automate these parameter choices. In addition, since the true surface Laplacian of f can be written as $\Delta_{\mathbb{M}} f = \Delta f - 2H\frac{\partial f}{\partial \boldsymbol{n}} - \frac{\partial^2 f}{\partial \boldsymbol{n}^2}$, where H is the mean curvature, we also plan to explore spatially variable hyperviscosity formulations that take into account the curvature H.

5. Results.

5.1. Surface advection. We now investigate the convergence properties of our stabilized RBF-FD methods on the surface advection equation. To the best of our knowledge, this is the first instance of a high-order RBF-FD method for transport on surfaces other than the sphere. On the sphere, we use two test cases: solid-body rotation of a cosine bell [43] and deformational flow of two Gaussians [25]. On the torus, we measure the transport of both cosine bells and Gaussians in a time-independent, spatially varying flow field. In all cases, the velocity fields return the initial conditions to their original spatial locations on the sphere and torus, allowing us to measure convergence rates. Our main focus is on convergence rates measured in the relative ℓ_2-norm, measured by testing our methods on quasi-uniform node sets of increasing sizes. We use icosahedral nodes on the sphere and staggered nodes on the torus, both as used in [22, 38, 40]. In all cases, we set ξ, the desired order of accuracy for the spatial derivatives, equal to ℓ (since a first-order differential operator is being approximated) and use $\xi = 2, 4, 6$. All subsequent parameters (including the overlap parameter δ) were set using Algorithm 1. For both tests, we use the classical third-order explicit Runge–Kutta method (RK3). We chose RK3 instead of the more popular explicit RK4 to demonstrate that our hyperviscosity formulation does not rely on the larger stability region of explicit RK4.

5.1.1. Advection on the sphere. We focus on two test cases for advection on the sphere. In both test cases, for a node set with N nodes, we set the time step to

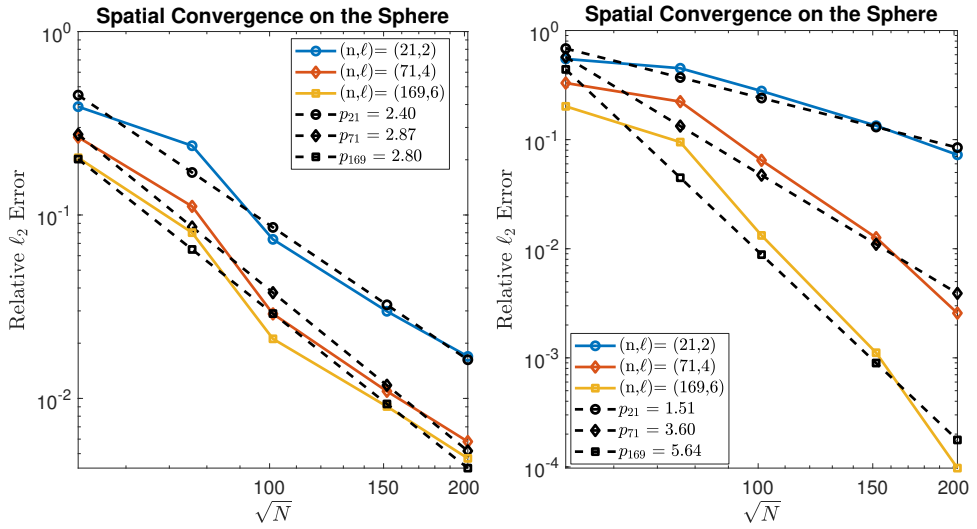


FIG. 3. *Convergence on the sphere for the surface advection equation of a cosine bell in a steady flow (left) and Gaussian in a deformational flow (right). The figure shows relative ℓ_2-error as a function of $\sqrt{N}$ for different values of stencil n and polynomial degree ℓ. The dashed lines indicate lines of best fit, and their slopes are indicated in the legend as a measure of convergence rates.*

$\triangle t = \frac{0.3}{\sqrt{N}}$, which roughly corresponds to a Courant number of 0.3. This time step was chosen to approximately allow spatial errors to dominate over temporal errors.

Solid body rotation of a cosine bell. Our first test on the sphere is the solid body rotation test case from [43], which involves advection of a cosine bell in a steady velocity field. The components of the steady velocity field for this test case in spherical coordinates ($-\pi \leq \lambda \leq \pi$, $-\pi/2 \leq \theta \leq \pi/2$) are given by

$$u(\lambda,\theta) = \sin(\theta)\sin(\lambda)\sin(\alpha) - \cos(\theta)\cos(\alpha), \quad v(\lambda,\theta) = \cos(\lambda)\sin(\alpha), \tag{53}$$

where α is the angle of rotation with respect to the equator. We set $\alpha = \pi/2$ to advect the initial condition over the poles and use a change of basis to obtain the velocity field in these coordinates. The initial condition is a compactly supported cosine bell centered at $(1,0,0)$,

$$c(\boldsymbol{x},0) = \begin{cases} \frac{1}{2}\left(1+\cos\left(\frac{\pi r}{R_\mathrm{b}}\right)\right) & \text{if } r < R_\mathrm{b}, \\ 0 & \text{if } r \geq R_\mathrm{b}, \end{cases}$$

where $\boldsymbol{x} = (x,y,z)$, $r = \arccos(x)$, and $R_\mathrm{b} = 1/3$. This initial condition is $C^1(\mathbb{S}^2)$, and one full revolution of the initial condition over the sphere requires simulation up to time $T = 2\pi$. The results of this simulation are shown in Figure 3 (left), which shows that increasing the polynomial degree ℓ does not increase convergence rates when advecting the cosine bell, simply due to the limited smoothness of the cosine bell. However, as has been observed in the literature [12, 39], increasing ℓ does improve the accuracy.

Deformational flow of Gaussian bells. The second test case involves advecting two Gaussian bells defined by

$$c(\boldsymbol{x},0) = 0.95\left(e^{-5\|\boldsymbol{x}-\mathbf{p}_1\|_2^2} + e^{-5\|\boldsymbol{x}-\mathbf{p}_2\|_2^2}\right)$$

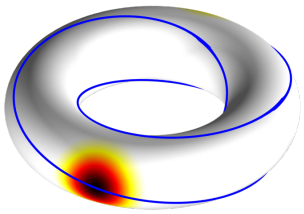


FIG. 4. *Cosine bell initial condition for the torus test together with the path the solution advects over (according to* (54)*) superimposed on top in blue.*

in a time-dependent deformational velocity field [25] whose components in spherical components are

$$u(\lambda,\theta,t) = \frac{10}{T}\cos\left(\frac{\pi t}{T}\right)\sin^2\left(\lambda - \frac{2\pi t}{T}\right)\sin(2\theta) + \frac{2\pi}{T}\cos(\theta),$$
$$v(\lambda,\theta,t) = \frac{10}{T}\cos\left(\frac{\pi t}{T}\right)\sin\left(2\lambda - \frac{2\pi t}{T}\right)\cos(\theta).$$

Here, $\mathbf{p}_1 = \left(\sqrt{3}/2, 1/2, 0\right)$ and $\mathbf{p}_2 = \left(\sqrt{3}/2, -1/2, 0\right)$. The flow field returns the solution to its initial position at final time $T = 5$. This solution is $C^\infty(\mathbb{S}^2)$ and is ideal for measuring convergence rates. The convergence results in the ℓ_2-norm are shown in Figure 3, which shows that increasing ℓ gives close to predicted convergence rates when advecting the Gaussian bells despite the complexity of the deformational flow test case.

5.1.2. Advection on the torus. For advection on the torus, we use the same velocity field for both tests. For a given set of points on the torus with inner radius $\frac{1}{3}$ and outer radius 1 in Cartesian coordinates (x, y, z), we define the following parametric coordinates:

$$\rho = \sqrt{x^2+y^2},\ \phi = \frac{1}{3}\tan^{-1}\left(\frac{y}{x}\right),\ \theta = \phi - \frac{1}{2}\tan^{-1}\left(\frac{z}{1-r}\right),$$
$$\rho_1 = 1 + \frac{1}{3}\cos(2(\phi-\theta)),\ \rho_2 = -2r_1\sin(2(\phi-\theta)).$$

Using these value, we then define the following time-independent tangential velocity field $\boldsymbol{u} = (u, v, w)^T$ on the torus with respect to the Cartesian basis:

$$\begin{aligned} u(\phi,\theta) &= \rho_1\cos(3\phi) - \rho_2(3\sin(3\phi)), \\ v(\phi,\theta) &= \rho_1\sin(3\phi) + \rho_2(3\cos(3\phi)), \\ w(\phi,\theta) &= -\frac{2}{3}\cos(2(\phi-\theta)). \end{aligned} \tag{54}$$

For a particle placed at the initial position (x_0, y_0, z_0) on the torus, this velocity field will advect the particle around a $(3, 2)$ torus knot, returning the particle back to its initial position at $T = 2\pi$ time units. See Figure 4 for an illustration of the solution path.

We advect two different bell-type initial conditions, similar to the sphere, in this velocity field. The centers of these initial conditions are at the locations $\boldsymbol{p}_1 = (1 + 1/3, 0, 0)$ and $\boldsymbol{p}_2 = -\boldsymbol{p}_1$. In both test cases, for a node set with N nodes, we set the time step to $\triangle t = \frac{0.3}{u_{max}\sqrt{N}}$, which roughly corresponds to a Courant number

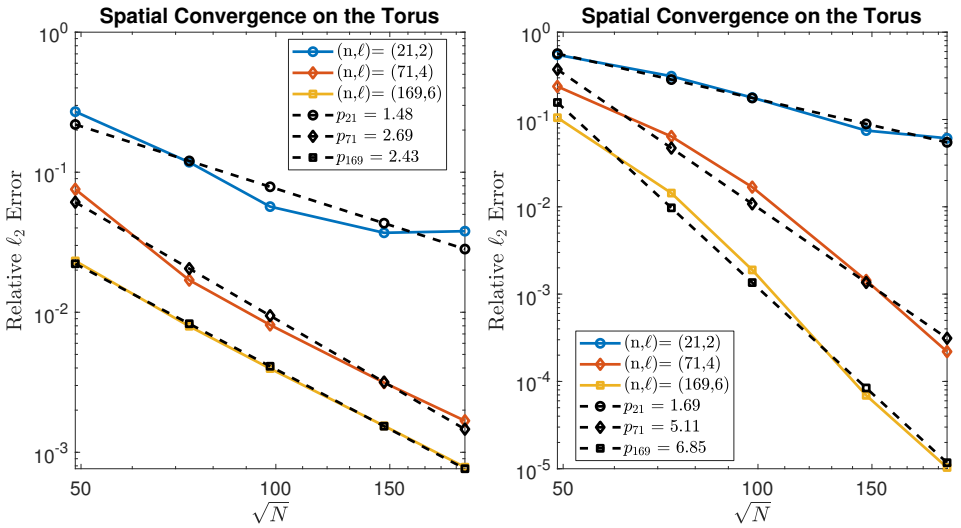


FIG. 5. *Convergence on the torus for the surface advection equation of two cosine bells in a steady flow (left) and two Gaussian bells in the same flow (right). The figure shows relative ℓ_2-error as a function of $\sqrt{N}$ for different values of stencil n and polynomial degree ℓ. The dashed lines indicate lines of best fit, and their slopes are indicated in the legend as a measure of convergence rates.*

of 0.3; here, $u_{max} = 4.1$ is the maximum pointwise magnitude of the velocity field over space. Again, this time step was chosen to approximately allow spatial errors to dominate over temporal errors introduced by RK3 integration.

Cosine bells. The first test case is the advection of a pair of compactly supported cosine bells given by the following initial condition:

$$c(\boldsymbol{x}, 0) = 0.1 + 0.9(q_1 + q_2),$$

where q_1 and q_2 are compactly supported functions given by

$$q_{1,2} = \begin{cases} \frac{1}{2}\left(1 + \cos\left(2\pi r_{1,2}(\boldsymbol{x})\right)\right) & \text{if } r_{1,2}(\boldsymbol{x}) < 0.5, \\ 0 & \text{if } r_{1,2}(\boldsymbol{x}) \geq 0.5, \end{cases}$$

and $r_{1,2}(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{p}_{1,2}\|$. As on the sphere, the solution has only one continuous derivative. The convergence results for advecting the cosine bells are shown in Figure 5 (left), which clearly shows that increasing the polynomial degree ℓ does not increase convergence rates when advecting the cosine bell but again does improve the accuracy (as on the sphere).

Gaussian bells. The second test case involves advecting a pair of Gaussian bells given by the following initial condition:

$$c(x, y, z, 0) = e^{-a(x+(1+1/3))^2+y^2-1.5az^2} + e^{-a(x-(1+1/3))^2+y^2-1.5az^2},$$

where $a = 20$. This centers two Gaussians at the same locations as the cosine bells from the first test case, with the rapid falloff ensuring locality without destroying smoothness. The convergence results in the ℓ_2-norm are shown in Figure 5 (right), which again shows that increasing ℓ gives close to predicted convergence rates when the solution is smooth.

5.2. Surface advection-diffusion. Next, we investigate the convergence properties of our stabilized RBF-FD method on the surface advection-diffusion equation. Once again, we focus on the sphere and the torus, using the same node sets as in the pure advection case. In this set of tests, we simply check convergence against a manufactured solution on both surfaces. However, since we are now approximating both first- and second-order differential operators, we set $\ell = \xi+1$, use $\xi = 2, 3, 4$, and set all other parameters according to Algorithm 1. All time-stepping was done using the semi-implicit backward differentiation formula of order 4 (SBDF4) [2], which is an IMEX method. While this is nowhere near optimal for high Peclet numbers, we use it to illustrate that our hyperviscosity formulation can be stepped implicitly for efficiency. In this set of tests, we use Peclet numbers of 1 and 100 to study the stability and convergence of our method in different parameter regimes, mimicking similar tests for Euclidean domains from [36]. For these tests, we select the smaller time step between one that corresponds to a Courant number of 0.3 and one that attempts to keep the temporal error below the spatial error. In other words, the time step $\triangle t$ is given by

$$\triangle t = \min\left(0.3N^{-\frac{1}{2}}, N^{-\frac{\xi}{8}}\right),$$

where N is the number of nodes, ξ is the desired spatial order of convergence, and the factor of $\xi/8$ comes from the fourth-order convergence of SBDF4 scheme. Here, we assume the nodes are quasi-uniform with a spacing of approximately $N^{-\frac{1}{2}}$.

5.2.1. Advection-diffusion on the sphere. For this test, we prescribe a solution and then manufacture a forcing term that makes the solution hold true. Our prescribed/manufactured solution is generated using the Y_4^3 real-valued spherical harmonic; more specifically, we shift it and multiply it by a time-dependent term. This is given by

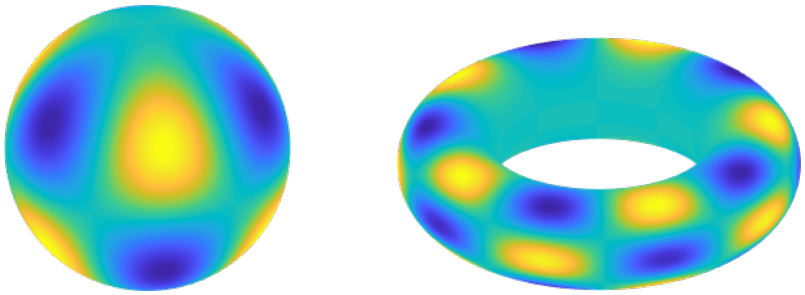
$$c(x, y, z, t) = 1 + \frac{3}{4}\sqrt{\frac{35}{2\pi}}\left(x^2 - 3y^2\right) xz \sin(t).$$

The forcing term is computed analytically as

$$f = \frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \nabla_{\mathbb{S}^2} c - \nu \Delta_{\mathbb{S}^2} c,$$

where $\Delta_{\mathbb{S}^2} c = -20(c-1)$; the initial condition is shown at time $t = 10^{-3}$ in Figure 6 (left). It is important to note that the forcing term does *not* contain the hyperviscosity operator. This allows us to correctly verify the impact of the hyperviscosity formulation on convergence to the manufactured solution. The velocity field $\boldsymbol{u}$ is the same as the steady velocity field from (53), with $\alpha = \pi/2$. We ran the simulation out to final time $T = 2\pi$ (one full period). The results are shown in Figure 7. Going from left to right across Figure 7, we see that convergence rates increase as the Peclet number is increased. This occurs due to our choice of $\ell = \xi + 1$. As we increase the Peclet number, the influence of the advection operator on transport increases relative to that of the diffusion operator. Since the advection operator is a first-order differential operator, this leads to an extra order of accuracy for the mixed character PDE since approximating the advection operator only requires a choice of $\ell = \xi$ (as seen in the previous subsection).

5.2.2. Advection-diffusion on the torus. We again prescribe a solution and then manufacture a forcing term that makes the solution hold true. Our prescribed/



(a) (b)

FIG. 6. *Manufactured solutions at time $t = 10^{-3}$ for surface advection-diffusion tests:* (a) *sphere and* (b) *torus.*

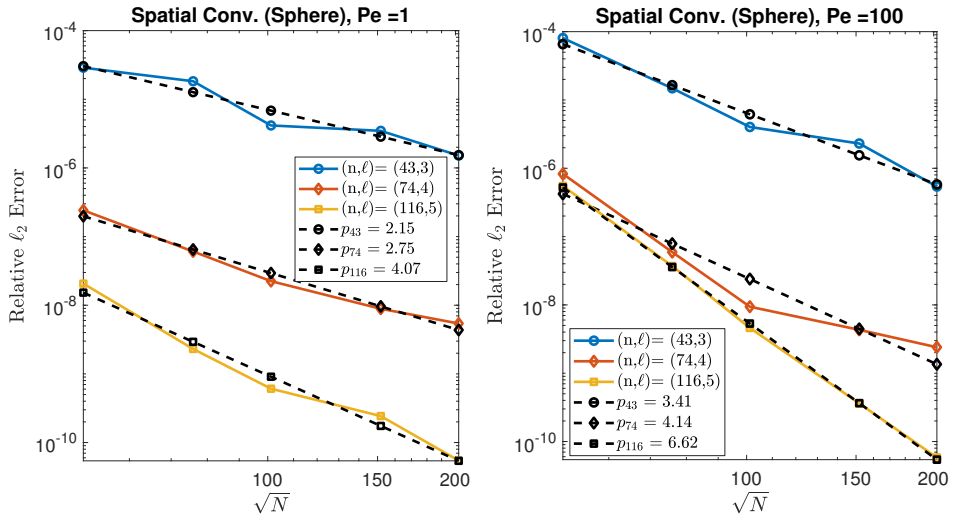


FIG. 7. *Convergence on the sphere for the surface advection-diffusion equation at Peclet numbers $Pe = 1$ (left) and $Pe = 100$ (right). The figure shows relative ℓ_2-error as a function of $\sqrt{N}$ for different values of stencil n and polynomial degree ℓ. The dashed lines indicate lines of best fit, and their slopes are indicated in the legend as a measure of convergence rates.*

manufactured solution is given by the following smooth function:

$$c(x, y, z, t) = 1 + \frac{1}{8} x \left(x^4 - 10x^2y^2 + 5y^4\right) \left(x^2 + y^2 - 60z^2\right) \sin(t).$$

This initial condition is shown in Figure 6 (right). The forcing term is computed analytically as

$$f = \frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \nabla_{\mathbb{T}} c - \nu \Delta_{\mathbb{T}} c,$$

where $\mathbb{T}$ represents the torus. Setting $\rho = \sqrt{x^2 + y^2}$, the surface Laplacian of the manufactured solution is

(55)

$$\Delta_{\mathbb{T}} c = \frac{-3}{8\rho^2} x \left(x^4 - 10x^2y^2 + 5y^4\right) \left(10248\rho^4 - 34335\rho^3 + 41359\rho^2 - 21320\rho + 4000\right) \sin(t).$$

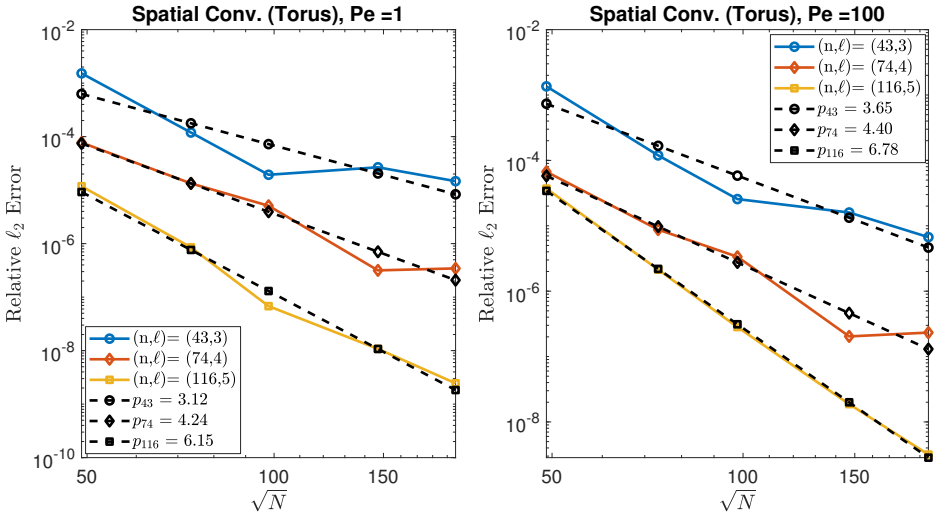


FIG. 8. *Convergence on the torus for the surface advection-diffusion equation at Peclet numbers* $Pe = 1$ *(left) and* $Pe = 100$ *(right). The figure shows relative* ℓ_2*-error as a function of* $\sqrt{N}$ *for different values of stencil* n *and polynomial degree* ℓ*. The dashed lines indicate lines of best fit, and their slopes are indicated in the legend as a measure of convergence rates.*

Again, the forcing term does not account for the hyperviscosity operator. The velocity field $\boldsymbol{u}$ is the same as the steady velocity field given by (54). The simulation was run out to final time $T = \pi$. The results are shown in Figure 8. Going from left to right across Figure 8, we once again see that convergence rates increase as the Peclet number is increased, though they appear to be higher than predicted even at low Peclet number. This illustrates that our method is stable under spatial and order refinement and that the hyperviscosity term vanishes appropriately, thereby allowing us to recover the manufactured solution. Note that the implicit time-stepping of the hyperviscosity operator was done purely for illustrative purposes at the higher Peclet numbers. For sufficiently high Peclet numbers, it would likely suffice to use fully explicit time-stepping of all terms in the PDE with the RK4 method.

6. Applications. We now present applications of our hyperviscosity model to advection-diffusion-reaction systems on two surfaces: the red blood cell surface [20, 38, 40] and Dupin's cyclide [20, 40]. In all cases, we advance our simulations in time using the SBDF2 method [2] with a time step of $\triangle t = 10^{-3}$. For the spatial discretization, we use $\xi = 4$ and once again set all other parameters according to Algorithm 1. Node sets on the cyclide were generated using the software Distmesh [27], and the algorithms from [37] were used to generate node sets on the red blood cell. In the latter case, these consist of generalized spiral nodes on the sphere mapped to the red blood cell, with supersampling and decimation ensuring quasi uniformity on the red blood cell.

6.1. Advective Cahn–Hilliard on Dupin's cyclide. We simulate an advective version of the Cahn–Hilliard equation on Dupin's cyclide. This is a nonlinear PDE governing phase separation with applications to both engineering and biology.

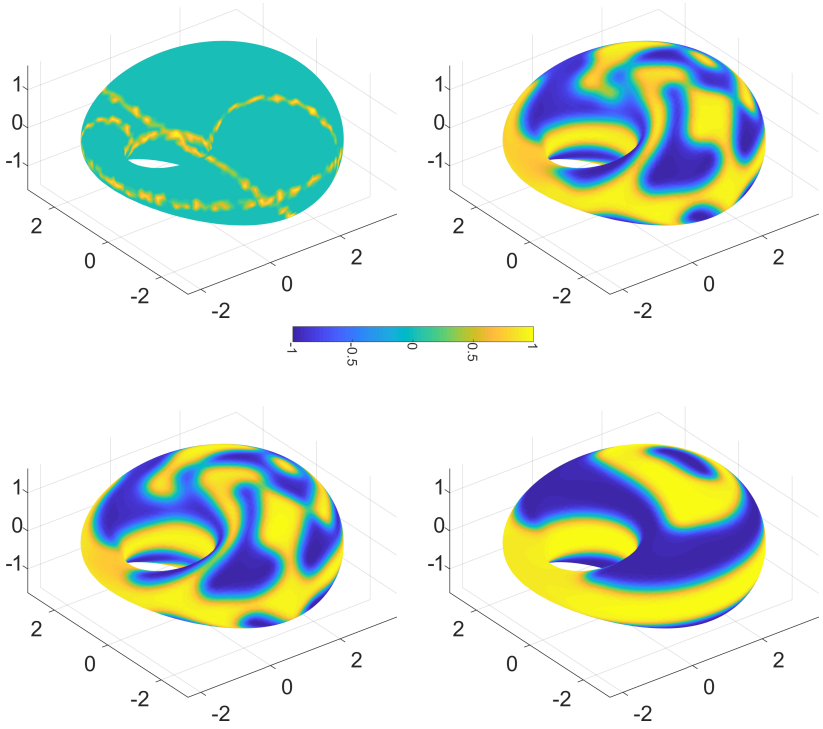


FIG. 9. *Solution of the advective Cahn–Hilliard equation* (56) *on Dupin's cyclide. The top left shows the initial condition, the bottom right shows the final solution at time* $t = 10$*, and the other figures are intermediate snapshots.*

The PDE (with artificial hyperviscosity) is given by

$$\frac{\partial c}{\partial t} + \nabla_{\mathbb{M}} \cdot (c\boldsymbol{u}) = \nu \Delta_{\mathbb{M}} c^3 - \nu \Delta_{\mathbb{M}} c - \nu \beta \Delta_{\mathbb{M}}^2 c + \gamma_1 \Delta_{\mathbb{M}}^{\gamma_2} c, \tag{56}$$

where $\Delta_{\mathbb{M}}^2$ is the surface bi-Laplacian and the $\gamma_1 \Delta^{\gamma_2}$ term is added solely to cancel the spurious growth in $\nabla_{\mathbb{M}} \cdot (c\boldsymbol{u})$. The velocity field $\boldsymbol{u}$ is computed as the surface curl of a streamfunction ψ whose gradient is given by

$$\nabla \psi = -10 \cos\left(\frac{\pi}{2} t\right) [1, 0, 0]^T, \tag{57}$$

$$\boldsymbol{u} = \boldsymbol{n} \times \nabla \psi. \tag{58}$$

Since the velocity field $\boldsymbol{u}$ is surface-incompressible ($\nabla_{\mathbb{M}} \cdot \boldsymbol{u} = 0$) by construction, the solutions $c = 1$ and $c = -1$ are critical points of the advection-diffusion-reaction system, causing the initial condition to separate over time into these two phases. We simulate (56) on Dupin's cyclide to time $t = 10$ using a random initial condition (Figure 9(a)), with $\nu = 0.5$ and $\beta = 0.02$. As in [38], we approximate the surface bi-Laplacian as $B = L^2$, where L is the differentiation matrix for the surface Laplacian. To balance stability and efficiency, we step the nonlinear terms $\nabla_{\mathbb{M}} \cdot (c\boldsymbol{u})$ and $\Delta_{\mathbb{M}} c^3$ explicitly in time and step all other terms implicitly. The results for $N = 8266$ nodes are shown in Figure 9. While the advection term causes the phases to occasionally remix, they separate unless forced to remix by the advection term. However, we

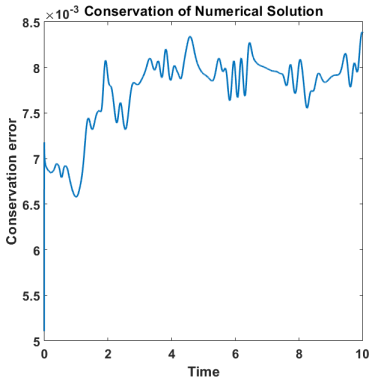


FIG. 10. *Conservation error in the numerical solution to the advective Cahn–Hilliard equation* (56) *on Dupin's cyclide as a function of time.*

observed that the phases remained separated without remixing for a sufficiently large final time. We also compute the conservation errors in the numerical solution as a function of time. If c is the numerical solution and c_0 is its initial value at time $t = 0$, the conservation error is given as $|\int_{\mathbb{M}} (c - c_0)|$. The integral over the surface $\mathbb{M}$ is computed using an eighth-order–accurate RBF-based quadrature technique from [34]. The conservation error is shown as a function of time in Figure 10. The figure shows that the conservation error grows in time. However, we have observed that increasing the order ξ or the number of nodes N significantly improves these errors, as seen in [39]. If necessary, the time-stepping can be modified to ensure conservation [28], but such an experiment is beyond the scope of this work.

6.2. Turing spots with advection on the red blood cell. Finally, we solve a (reaction-)coupled advection-diffusion-reaction system on an idealized red blood cell. Specifically, we simulate the pattern-generating Turing system given by

$$\frac{\partial c_1}{\partial t} + \nabla_{\mathbb{M}} \cdot (c_1 \mathbf{w}) = \delta_1 \Delta_{\mathbb{M}} c_1 + \alpha c_1 \left(1 - \tau_1 c_2^2\right) + c_2 \left(1 - \tau_2 c_1\right) + \gamma_1 \Delta_{\mathbb{M}}^{\gamma_2} c_1, \tag{59}$$

$$\frac{\partial c_2}{\partial t} + \nabla_{\mathbb{M}} \cdot (c_2 \mathbf{w}) = \delta_2 \Delta_{\mathbb{M}} c_2 + \beta c_2 \left(1 + \frac{\alpha \tau_1}{\beta} c_1 c_2\right) + c_1 \left(\eta_1 + \tau_2 c_2\right) + \gamma_1 \Delta_{\mathbb{M}}^{\gamma_2} c_2, \tag{60}$$

where the hyperviscosity terms are only added to cancel spurious growth modes in the surface gradient terms. We use the parameters $\delta_1 = 0.0011$, $\delta_2 = 0.0021$, $\tau_1 = 0.02$, $\tau_2 = 0.2$, $\alpha = 0.899$, $\beta = -0.91$, and $\eta_1 = -\alpha$, with the initial condition as shown in Figure 11(a). This set of parameter choices promotes spot formation in the absence of advection [38, 40]. The velocity field $\mathbf{w}$ is simply a scaled version of (57)–(58); i.e., we set $\mathbf{w} = 0.1\boldsymbol{u}$. We chose this scaling because the Turing patterns (specifically, spots) are sensitive to the choice of the velocity field, with a very fast velocity field inhibiting spot formation. As before, the advection and reaction terms were stepped explicitly in time, while the hyperviscosity and diffusion terms are stepped implicitly. The simulations were run out to a final time of $t = 800$, and the results $N = 2553$ nodes are shown in Figure 11. We see both the formation of spots from the initial condition in Figure 11(b) and their motion due to the velocity field (going from Figure 11(b) to Figure 11(c)).

7. Summary and future work. We have presented a novel, automatic procedure for stabilizing discretizations of linear advection and advection-diffusion equa-

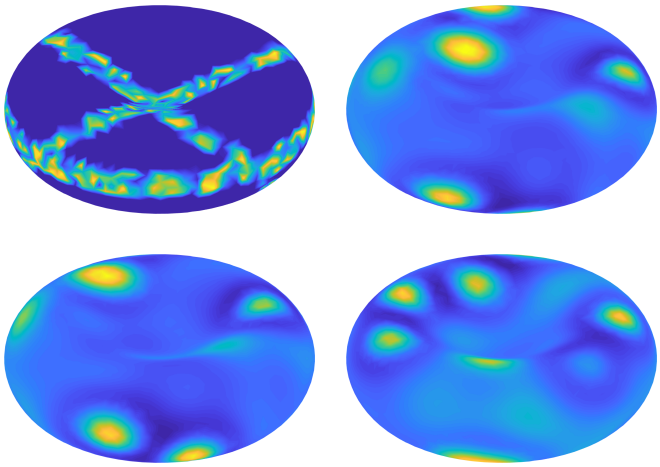


FIG. 11. *Solution of the advective Turing equations* (59)–(60) *on an idealized red blood cell. The top left shows the initial condition, the bottom right the final solution at time* $t = 800$*, and the other figures are intermediate snapshots. A brighter color indicates a higher concentration of* c_1*, and a darker color indicates a lower concentration.*

tions on manifolds. In particular, we have investigated the procedure in the context of overlapping RBF-LOI discretizations. The geometric, high-order flexibility of RBF-LOI discretizations when paired with automatically parameterized hyperviscosity yields a numerical algorithm that is efficient, accurate, and locally and globally stable for all examples we have investigated. We have tested our algorithm on surface advection and surface advection-diffusion problems on manifolds of codimension 1 embedded in $\mathbb{R}^3$. The automated choice of hyperviscosity essentially eliminates tedious and expensive hand-tuning of stabilization parameters when applying our RBF-LOI algorithm across variegated PDEs and geometries.

The mathematical strategy we have developed for automated tuning of the hyperviscosity term provides general guidance for stabilizing both linear and linearized PDEs on manifolds, can be easily applied to non-RBF–type discretizations, and is easily generalized to manifolds embedded in arbitrary Euclidean dimensions. Our analysis also provides diagnostic guidance for determining when discretized advection-diffusion systems possess sufficient diffusion to obviate the need for stabilizing hyperviscosity terms.

REFERENCES

[1] K. A. AITON, *A Radial Basis Function Partition of Unity Method for Transport on the Sphere*, master's thesis, Boise State University, Boise, ID, 2014.

[2] U. M. ASCHER, S. J. RUUTH, AND B. T. R. WETTON, *Implicit-explicit methods for time-dependent PDEs*, SIAM J. Numer. Anal., 32 (1997), pp. 797–823.

[3] V. BAYONA, N. FLYER, AND B. FORNBERG, *On the role of polynomials in RBF-FD approximations:* III. *Behavior near domain boundaries*, J. Comput. Phys., 380 (2019), pp. 378–399, https://doi.org/10.1016/j.jcp.2018.12.013.

[4] 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*, J. Comput. Phys., 332 (2017), pp. 257–273.

[5] V. BAYONA, M. MOSCOSO, M. CARRETERO, AND M. KINDELAN, *RBF-FD formulas and convergence properties*, J. Comput. Phys., 229 (2010), pp. 8281–8295.

[6] J. Behrens and A. Iske, *Grid-free adaptive semi-Lagrangian advection using radial basis functions*, Comput. Math. Appl., 43 (2002), pp. 319–327.
[7] O. Davydov and D. T. Oanh, *Adaptive meshless centres and RBF stencils for Poisson equation*, J. Comput. Phys., 230 (2011), pp. 287–304.
[8] G. E. Fasshauer, *Meshfree Approximation Methods with MATLAB*, Interdisciplinary Mathematical Sciences 6, World Scientific, Singapore, 2007.
[9] G. E. Fasshauer and M. J. McCourt, *Stable evaluation of Gaussian radial basis function interpolants*, SIAM J. Sci. Comput., 34 (2012), pp. A737–A762.
[10] N. Flyer, G. A. Barnett, and L. J. Wicker, *Enhancing finite differences with radial basis functions: Experiments on the Navier-Stokes equations*, J. Comput. Phys., 316 (2016), pp. 39–62.
[11] N. Flyer, B. Fornberg, V. Bayona, and G. A. Barnett, *On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy*, J. Comput. Phys., 321 (2016), pp. 21–38.
[12] 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.
[13] N. Flyer and G. B. Wright, *Transport schemes on a sphere using radial basis functions*, J. Comput. Phys., 226 (2007), pp. 1059–1084.
[14] N. Flyer and G. B. Wright, *A radial basis function method for the shallow water equations on a sphere*, Proc. Roy. Soc. A, 465 (2009), pp. 1949–1976.
[15] B. Fornberg, E. Larsson, and N. Flyer, *Stable computations with Gaussian radial basis functions*, SIAM J. Sci. Comput., 33(2) (2011), pp. 869–892.
[16] B. Fornberg and E. Lehto, *Stabilization of RBF-generated finite difference methods for convective PDEs*, J. Comput. Phys., 230 (2011), pp. 2270–2285.
[17] B. Fornberg, E. Lehto, and C. Powell, *Stable calculation of Gaussian-based RBF-FD stencils*, Comput. Math. Appl., 65 (2013), pp. 627–637.
[18] 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.
[19] 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.
[20] E. J. Fuselier and G. B. Wright, *A high-order kernel method for diffusion and reaction-diffusion equations on surfaces*, J. Sci. Comput., 56 (2013), pp. 535–565.
[21] S. D. Lawley and V. Shankar, *Asymptotic and numerical analysis of a stochastic PDE model of volume transmission*. submitted, 2018.
[22] E. Lehto, V. Shankar, and G. B. Wright, *A radial basis function (RBF) compact finite difference (FD) scheme for reaction-diffusion equations on surfaces*, SIAM J. Sci. Comput., 39 (2017), pp. A2129–A2151.
[23] H. Ma, *Chebyshev–Legendre spectral viscosity method for nonlinear conservation laws*, SIAM J. Numer. Anal., 35 (1998), pp. 869–892.
[24] H. Ma, *Chebyshev–Legendre super spectral viscosity method for nonlinear conservation laws*, SIAM J. Numer. Anal., 35 (1998), pp. 893–908.
[25] R. D. Nair and P. H. Lauritzen, *A class of deformational flow test cases for linear transport problems on the sphere*, J. Comput. Phys., 229 (2010), pp. 8868–8887.
[26] A. Narayan and D. Xiu, *Stochastic collocation methods on unstructured grids in high dimensions via interpolation*, SIAM J. Sci. Comput., 34 (2012), pp. A1729–A1752.
[27] P.-O. Persson and G. Strang, *A simple mesh generator in Matlab*, SIAM Rev., 46 (2004), pp. 329–345, https://doi.org/10.1137/S0036144503429121.
[28] A. Petras, L. Ling, C. Piret, and S. J. Ruuth, *A least-squares implicit RBF-FD closest point method and applications to PDEs on moving surfaces*, J. Comput. Phys., 381 (2019), pp. 146–161.
[29] A. Petras, L. Ling, and S. J. Ruuth, *An RBF-FD closest point method for solving PDEs on surfaces*, J. Comput. Phys., 370 (2018), pp. 43–57.
[30] C. Piret, *The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces*, J. Comput. Phys., 231 (2012), pp. 4662–4675.
[31] C. Piret and J. Dunn, *Fast RBF OGR for solving PDEs on arbitrary surfaces*, AIP Conference Proceedings, 1776 (2016).
[32] J. A. Reeger and B. Fornberg, *Numerical quadrature over the surface of a sphere*, Stud. Appl. Math., 137 (2016), pp. 174–188.
[33] J. A. Reeger and B. Fornberg, *Numerical quadrature over smooth surfaces with boundaries*, J. Comput. Phys., 355 (2018), pp. 176–190.

[34] J. A. Reeger, B. Fornberg, and M. L. Watts, *Numerical quadrature over smooth, closed surfaces*, Proc. Roy. Soc. Lond. A, 472 (2016), https://doi.org/10.1098/rspa.2016.0401, 472 (2016t).

[35] V. Shankar, *The overlapped radial basis function-finite difference (RBF-FD) method: A generalization of RBF-FD*, J. Comput. Phys., 342 (2017), pp. 211–228.

[36] V. Shankar and A. L. Fogelson, *Hyperviscosity-based stabilization for radial basis function-finite difference (RBF-FD) discretizations of advection-diffusion equations*, J. Comput. Phys., 372 (2018), pp. 616–639.

[37] V. Shankar, R. M. Kirby, and A. L. Fogelson, *Robust node generation for mesh-free discretizations on irregular domains and surfaces*, SIAM J. Sci. Comput., 40 (2018), pp. A2584–A2608.

[38] 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*, J. Comput. Phys., 373 (2018), pp. 722–735.

[39] V. Shankar and G. B. Wright, *Mesh-free semi-Lagrangian methods for transport on a sphere using radial basis functions*, J. Comput. Phys., 366 (2018), pp. 170–190.

[40] V. Shankar, G. B. Wright, R. M. Kirby, and A. L. Fogelson, *A radial basis function (RBF)-finite difference (FD) method for diffusion and reaction–diffusion equations on surfaces*, J. Sci. Comput., 63 (2014), pp. 745–768.

[41] E. Tadmor, *Convergence of spectral methods for nonlinear conservation laws*, SIAM J. Numer. Anal., 26 (1989), pp. 30–44.

[42] H. Wendland, *Scattered data approximation*, Cambridge Monogr. Appl. Comput. Math. 17, Cambridge University Press, Cambridge, 2005.

[43] 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*, J. Comput. Phys., 102 (1992), pp. 211–224.

[44] 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.

[45] G. B. Wright and B. Fornberg, *Stable computations with flat radial basis functions using vector-valued rational approximations*, J. Comput. Phys., 331 (2017), pp. 13756.