Mesh-free Semi-Lagrangian Methods for Transport on a Sphere Using Radial Basis Functions

Varun Shankar[a,*], Grady B. Wright[b]

[a] *Department of Mathematics, University of Utah, UT, USA*
[b] *Department of Mathematics, Boise State University, ID, USA*

Abstract

We present three new semi-Lagrangian methods based on radial basis function (RBF) interpolation for numerically simulating transport on the surface of a sphere. The methods are mesh-free and are formulated entirely in Cartesian coordinates, thus avoiding any irregular clustering of nodes at artificial boundaries on the sphere and naturally bypassing any apparent artificial singularities associated with surface-based coordinate systems. The semi-Lagrangian framework allows these new methods to avoid the use of any stabilization terms (such as hyperviscosity) during time-integration, thus reducing the number of parameters that have to be tuned. Furthermore, time-steps that are several hundred times the CFL number can be used. The three new methods are based on interpolation using 1) global RBFs, 2) local RBF stencils, and 3) RBF partition of unity. For the latter two of these methods, we find that it is crucial to include some low degree spherical harmonics in the interpolants. Standard test cases consisting of solid body rotation and deformational flow are used to compare and contrast the methods in terms of their accuracy, efficiency, conservation properties, and dissipation/dispersion errors. For global RBFs, spectral spatial convergence is observed for smooth solutions on quasi-uniform nodes, while high-order accuracy is observed for the local RBF stencil and partition of unity approaches.

Keywords: RBF, Hyperbolic PDEs, Advection, meshless

1. Introduction

Radial basis function (RBFs) methods have been used for over a decade to solve partial differential equations (PDEs) on the surface of a sphere. These methods can broadly be classified into *global* RBF collocation methods [1–5], RBF-generated finite difference (RBF-FD) methods [6–10], and more recently RBF-partition of unity (RBF-PU) collocation methods [11]. Global RBF methods when used with infinitely-smooth RBFs show spectral convergence on smooth problems at the cost of *dense* differentiation matrices; in contrast, RBF-FD and RBF-PU methods produce sparse differentiation matrices and high-order algebraic convergence rates. All of these methods can use "scattered" nodes in their discretizations of a sphere, and have the benefit of being independent of any surface-based coordinate system. They thus avoid any unnatural grid clustering and do not suffer from any coordinate singularities.

In this paper, we present three new RBF methods with similar benefits for numerically solving the transport equation on the surface of a sphere in an incompressible velocity field. For the unit sphere $\mathbb{S}^2$, this PDE is given

$$\frac{Dq}{Dt} = 0, \quad \frac{D}{Dt} := \frac{\partial}{\partial t} + \boldsymbol{u} \cdot \nabla_{\mathbb{S}^2}, \tag{1}$$

where q is the scalar quantity being transported, $\boldsymbol{u}$ is a surface divergence-free vector field that is tangent to $\mathbb{S}^2$, and $\nabla_{\mathbb{S}^2}$ denotes the surface gradient operator on $\mathbb{S}^2$. Since global atmospheric flows are dominated

*Corresponding Author
Email addresses: vshankar@math.utah.edu (Varun Shankar), gradywright@boisestate.edu (Grady B. Wright)

by the horizontal advection process, the numerical solution to the transport problem is a fundamental part of any solver for these flows.

Currently, all RBF discretizations for the transport equation (and more general hyperbolic equations like the shallow water equations) on the surface of the sphere suffer from the same drawback: the eigenvalues of the differentiation matrices corresponding to the surface gradient operator may in general have positive real parts, leading to either a slow or rapid onset of instability during a numerical simulation [1]. As of this paper, the only known approach to rectify this problem is to add an artificial hyperviscosity of the form $(-1)^{(k+1)}\gamma\Delta^k$ in the right hand side of the PDE, where $k \geq 2$ is an integer and $\gamma > 0$ is some small real number that scales *inversely* with the total number of nodes N. The intuition here is that higher powers of the Laplacian Δ will damp out the eigenvectors associated with the rogue eigenvalues of the discretized surface gradient, while leaving the others essentially untouched [6, 7, 12, 13]. With global RBFs, the hyperviscosity operator typically takes the form of γA^{-1}, where A is the global RBF interpolation matrix whose inverse mimics the properties of high powers of the Laplacian [6]; a similar approach can be employed for the RBF-PU method [11]. Unfortunately, for a given PDE and node set on a sphere, the precise values of γ and k required to stabilize the numerical solution may need to be determined by trial and error, which can add to the computational expense of the method.

A common way to naturally stabilize local Eulerian methods for transport is to use "upwinding", which uses dynamic direction-dependent information about the flow field. However, this form of upwinding typically requires an underlying mesh and so is impractical for truly mesh-free local methods like RBF-FD and RBF-PU collocation. An important class of methods that naturally possess upwinding and the *automatic* addition of (nonlinear) stabilizing viscosity, are semi-Lagrangian (SL) methods. SL methods have successfully been used in conjunction with the finite element and spectral element methods for solving problems on planar domains [14]. Conservative SL schemes have also been used in conjunction with thin-plate spline RBF interpolation on Voronoi cells in planar domains [15]. These methods allow for time-steps that are *much* larger than the CFL number without sacrificing stability, unlike their Eulerian counterparts [16]. More relevant to this article, SL advection techniques have a rich history in numerical simulation of transport on the surface of the sphere [17]. However, such methods have generally used latitude-longitude grids and spherical coordinate systems (e.g. [18–20]) or other regular surfaced based grids and local surface-based coordinate systems (e.g. [21–23]). This can lead to issues because of singularities that arise in the mappings from the physical sphere to the surface based coordinate systems.

In this paper, we present three new high-order SL methods for transport on the sphere based on interpolation with global RBFs, local RBFs, and RBF-PU methods. These methods are mesh-free, allowing for scattered node discretizations, and are formulated entirely in Cartesian coordinates, so as to avoid any surface based coordinate singularities. The local RBF and RBF-PU methods also allow a type of "p-refinement" for increasing the accuracy for a given fixed set of discretization nodes. We demonstrate that the SL framework lends our new methods both accuracy and intrinsic stability, thereby eliminating the need for a hyperviscosity term. For the local RBF and RBF-PU methods, we propose using "scale-free" RBFs appended with spherical harmonics (an idea related to that of Flyer et al. [24] for planar domains) to further reduce the number of tuning parameters (i.e. the shape-parameter) and to bypass so-called error stagnation. We compare and contrast all three methods using three standard test cases from the literature—solid-body rotation of a cosine bell from [25] and deformational flow of two bells from [26]. The focus of these comparisons is on the overall accuracy, dissipation and dispersion properties, mass conservation, and computational cost. We find that the computational costs of these new methods are comparable to those of existing RBF collocation and finite-difference techniques, and, in particular, that for the local RBF and RBF-PU methods are highly scalable to large node sets. We note that RBFs have previously been used for SL transport in [27], but the focus here was on planar domains and global RBF methods. This is the first application of RBFs to SL transport on the sphere with accurate and scalable numerical methods.

The paper is organized as follows. In the next section, we review the global RBF, local RBF and RBF-PU methods in the context of interpolation. In Section 3, we review the SL advection technique and discuss how to use use the three RBF methods within the SL framework in an efficient fashion. Section 4 compare and contrast the new SL methods on three standard test cases for transport on the sphere. Finally, we conclude with a summary of our results and future research directions in Section 5.

2. Global, local, and partition of unity RBF interpolation on $\mathbb{S}^2$

RBFs are a well-established method for interpolating/approximating data over a set of "scattered" nodes $X = \{\boldsymbol{x}_j\}_{j=1}^N \subset \Omega \subseteq \mathbb{R}^d$. The standard method uses linear combinations of shifts of a kernel $\phi : \Omega \times \Omega \to \mathbb{R}$ with the property that $\phi(\boldsymbol{x}, \boldsymbol{y}) := \phi(\|\boldsymbol{x} - \boldsymbol{y}\|)$ for $\boldsymbol{x}, \boldsymbol{y} \in \Omega$, where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^d$. Kernels with this property are referred to as *radial kernels* or simply *radial functions*. In the case where $\Omega = \mathbb{S}^2$, these kernels are sometimes referred to as *spherical basis functions* since $\phi(\|\boldsymbol{x} - \boldsymbol{y}\|) = \phi(2 - 2\boldsymbol{x}^T\boldsymbol{y})$ when $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{S}^2$, i.e. ϕ will only depend on the cosine of the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$. We will use the term RBFs and not make use of this simplification since our numerical schemes use extrinsic coordinates.

Below we review the three interpolation methods used in this study. We give cursory details on the first two methods, as they have appeared in many other places in the literature; see the recent book by Fornberg & Flyer [28] for further details and applications of these two methods.

2.1. Global RBF interpolation

Given a set of nodes $X = \{\boldsymbol{x}_k\}_{k=1}^N \subset \mathbb{S}^2$ and a continuous target function $f : \mathbb{S}^2 \to \mathbb{R}$ sampled at the nodes in X, the standard *global* RBF interpolant to the data has the form

$$s(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \phi(\|\boldsymbol{x} - \boldsymbol{x}_k\|). \tag{2}$$

The expansion coefficients $\{c_k\}_{k=1}^N$ are determined by enforcing $s|_X = f|_X$, which can be expressed by the following linear system:

$$\underbrace{\begin{bmatrix} \phi(r_{1,1}) & \phi(r_{1,2}) & \dots & \phi(r_{1,N}) \\ \phi(r_{2,1}) & \phi(r_{2,2}) & \dots & \phi(r_{2,N}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(r_{N,1}) & \phi(r_{N,2}) & \dots & \phi(r_{N,N}) \end{bmatrix}}_{A_X} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}}_{\mathbf{c}_X} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}}_{\mathbf{f}_X}, \tag{3}$$

where $r_{i,j} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||$. If ϕ is, for example, a positive-definite radial kernel on $\mathbb{R}^3$, and all nodes in X are distinct, then the matrix A_X above is guaranteed to be positive definite, so that (2) is well-posed. Examples of various choices for ϕ, including relaxed conditions to guarantee the well-posedness of (2) can be found in [29, Ch. 3–12]. Results on the approximation power of RBF interpolants on the sphere for target functions of various smoothness can be found in, for example, [30–32]. A result from [32] is that for infinitely-smooth target functions, convergence rates that are faster than any polynomial order can be realized for various infinitely smooth ϕ. Additionally, global RBF interpolants based on various infinitely-smooth kernels have been shown to converge to spherical harmonic interpolants as the kernels are scaled to become flat (the so-called flat limit) [33].

One issue with global RBF interpolation is that the linear system for determining the interpolation coefficients (3) is dense, thereby leading to an $O(N^3)$ computational cost to solve it using a direct method. Another issue is that the matrices can become ill-conditioned. While the RBF-QR method [33] can be used to bypass the ill-conditioning associated with small shape parameters (i.e. flat radial kernels), algorithms for bypassing the $O(N^3)$ computational cost *and* ill-conditioning have not yet been developed. The next two subsections discuss techniques for addressing these issues. We note that an alternative to these methods, that is not pursued here, is to use compactly-supported kernels in a multilevel type framework [34].

2.2. Local RBF interpolation

In this method, RBF interpolants are used locally over a small collection of points surrounding each node in the set X (this technique forms the foundation for RBF-FD methods [35]). The method proceeds as follows. For each node $\boldsymbol{x}_k$, $k = 1, \dots, N$, in the set X, we select subsets of X that consist of $\boldsymbol{x}_k$ and its $n - 1$ nearest neighbors, where $n << N$. We refer to these subsets as *stencils* and denote them by X_k, and refer to node $\boldsymbol{x}_k$ as the *center point* of the stencil X_k; see Figure 1 for an illustration. The nearest neighbors are typically determined in a preprocessing step using a data structure such as a kd-tree, which

typically requires $O(N \log N)$ operations to construct. On each stencil, an RBF interpolant is constructed to the nodes in X_k and the corresponding function values. It is then used to reconstruct the underlying target function globally at all points within some neighborhood of the center point of the stencil. While this approach does not produce a globally smooth interpolant to the underlying target function, we find that it works well in the SL setting.

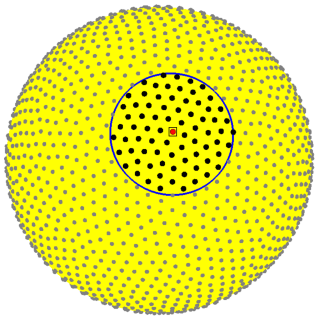


Figure 1: Illustration of the nodes in an example stencil for the local RBF method on the sphere. The global nodes X are marked as small greyed balls, while the nodes that make up the stencil X_k are marked as black balls, with the center point $\boldsymbol{x}_k$ marked in red an with a black square around it.

For the local RBF interpolants, rather than use (2) on each stencil, we follow an approach related to that proposed in Flyer et al. [24] for RBF-FD for increasing the approximation power of the interpolants and avoiding stagnation (or saturation) errors for interpolation in $\mathbb{R}^d$. Their approach augments the standard form of the interpolant (2) with a linear combination of polynomials up to a degree of roughly $n/2$ (in two-dimensions), together with a set of "moment conditions" on the RBF coefficients. A natural extension of their approach to the sphere is to augment the standard interpolant with spherical harmonics up to a degree L that grows with n (see Section 2.4), since these are polynomials in $\mathbb{R}^3$ that are linearly independent when restricted the sphere. The exact form of the augmented local RBF interpolants on each stencil X_k that we use is then given by

$$s_k(\boldsymbol{x}) = \sum_{j=\mathcal{I}_1^k}^{\mathcal{I}_n^k} c_j^k \phi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) + \sum_{i=1}^{(L+1)^2} d_i^k p_i(\boldsymbol{x}), \tag{4}$$

where $\mathcal{I}_1^k, \ldots, \mathcal{I}_n^k$ are the indices into the global node set X of the nodes contained X_k and $\{p_i\}_{i=1}^{(L+1)^2}$ are a basis for the space of spherical harmonics of degree L. To determine the unknown coefficients, we impose that s_k interpolates the data associated with the stencil X_k and that the coefficients c_j^k satisfy the following $(L+1)^2$ moment conditions:

$$\sum_{j=\mathcal{I}_1^k}^{\mathcal{I}_n^k} c_j^k p_i(\boldsymbol{x}_j) = 0, \quad i = 1, \ldots, (L+1)^2,$$

which are the standard conditions imposed in the literature for polynomials in $\mathbb{R}^d$ [24]. The interpolation and moment conditions lead to the following linear system:

$$\underbrace{\begin{bmatrix} A_k & P_k \\ P_k^T & 0 \end{bmatrix}}_{\widetilde{A}_k} \underbrace{\begin{bmatrix} \mathbf{c}_k \\ \mathbf{d}_k \end{bmatrix}}_{\widetilde{\mathbf{c}}_k} = \underbrace{\begin{bmatrix} \mathbf{f}_k \\ \mathbf{0} \end{bmatrix}}_{\widetilde{\mathbf{f}}_k}, \tag{5}$$

where A_k is the RBF interpolation matrix for the nodes in stencil X_k (see (3)) and P_k is a n-by-$(L+1)^2$ matrix with column i containing entries $p_i(\boldsymbol{x}_j)$, $j = \mathcal{I}_1^k, \ldots, \mathcal{I}_n^k$. These local interpolants can be used to approximate functions to high-order algebraic accuracy determined by the stencil size n and the degree of the spherical harmonics. Much like in [24], we find that augmenting the standard RBF interpolant with spherical harmonics significantly enhances the accuracy of the local RBF interpolants. We will henceforth use tilde's above variables that correspond to the augmented RBF system (5).

2.3. RBF-PU interpolation

The RBF-PU method is similar to the local RBF interpolation method. However, instead of forming and using different local interpolants for each node $\boldsymbol{x}_k$, local interpolants are constructed over a collection of patches that cover the sphere. These interpolants are then combined using compactly-supported weight functions on each patch that, all together, form a partition of unity. In this way, the method results in a smooth interpolant over the sphere, whereas the local approach does not. The RBF-PU method was first introduced for problems in the plane by Wendland [36] (see also [29, Ch. 29]) and for interpolation problems on a sphere by Cavoretto & De Rossi in [37]. It has since been extended to other domains and applications [11, 12, 38]. Specific details on the RBF-PU construction are provided below in the context of our application.

Let $\Omega_1, \ldots, \Omega_M$ be a collection of distinct spherical caps on $\mathbb{S}^2$ with the properties that the set of all patches provides an open covering of $\mathbb{S}^2$, i.e. $\cup_{\ell=1}^{M} \Omega_\ell = \mathbb{S}^2$, and each Ω_ℓ contains at least one node from X. These caps are the patches in the PU method. Figure 2 (a) illustrates a typical collection of patches for a quasi-uniformly distributed node set X on $\mathbb{S}^2$, which is what we use in this study. Let $\boldsymbol{\omega}_\ell \in \mathbb{S}^2$, $\ell = 1, \ldots, M$, denote the center of patch Ω_ℓ and $R_\ell > 0$ denote the radius of the patch, measured as the Euclidean distance from $\boldsymbol{\omega}_\ell$. For each patch, we define the following compactly-supported weight function:

$$\varphi_\ell(\boldsymbol{x}) = \varphi\left(\frac{\|\boldsymbol{x} - \boldsymbol{\omega}_\ell\|}{R_\ell}\right), \tag{6}$$

where $\varphi : \mathbb{R}^+ \to \mathbb{R}$ has compact support over the interval $[0, 1)$. In this study, we use the radially-symmetric cubic B-spline

$$\varphi(r) = \begin{cases} \frac{2}{3} + 4\,(r-1)\,r^2 & \text{if } 0 \le r < \frac{1}{2}, \\ -\frac{4}{3}(r-1)^3 & \text{if } \frac{1}{2} \le r < 1, \\ 0 & \text{if } 1 \le r. \end{cases} \tag{7}$$

Using (6), we define the PU weight functions for the collection of patches $\Omega_1, \ldots, \Omega_M$ as

$$w_\ell(\boldsymbol{x}) = \frac{\varphi_\ell(\boldsymbol{x})}{\sum_{j=1}^{M} \varphi_j(\boldsymbol{x})}, \ \ell = 1, \ldots, M. \tag{8}$$

Note that each w_ℓ is only supported over Ω_ℓ and that the summation on the bottom only involves terms that are non-zero over patch Ω_ℓ, which should be much smaller than M. Figure 2 (b) displays one of these weight functions for a quasi-uniform distribution of overlapping patches.

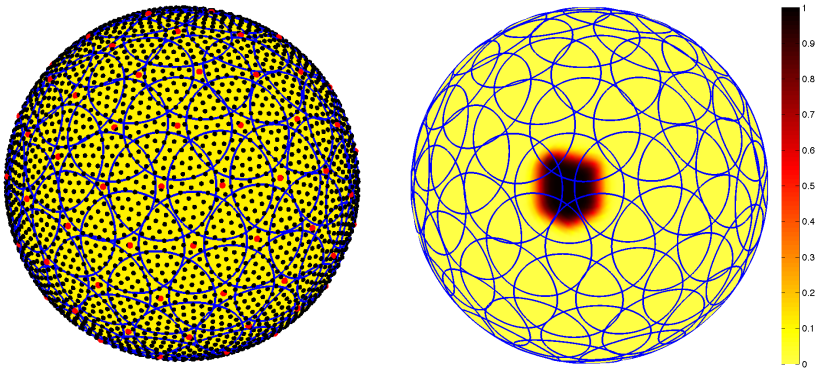
We use these PU weight functions to define the RBF-PU interpolant as follows. Let X_ℓ denote the nodes from X that belong to patch Ω_ℓ and let s_ℓ denote the RBF interpolant of the form (4) to the target function f over X_ℓ. Then the RBF-PU interpolant is given by

$$s(\boldsymbol{x}) = \sum_{\ell=1}^{M} w_\ell(\boldsymbol{x}) s_\ell(\boldsymbol{x}). \tag{9}$$

Since $\{w_\ell\}_{\ell=1}^{M}$ form a partition of unity, it is straightforward to show that s interpolates the target function at all nodes in X. The interpolant is also smooth over the entire sphere. Here we are using the augmented RBF interpolants (4) over each patch (which differs from [37]). We have found that this gives better accuracy than the standard interpolant (2). In Section 3.3.3, we discuss our strategy for choosing the patches.

2.4. Choosing the radial kernels

In the past, infinitely-smooth radial kernels, such as the inverse multiquadric (IMQ) kernel $\phi(r) = (1+(\varepsilon r)^2)^{-1/2}$, have been primarily used for RBF approximations to advection-dominated PDEs on a sphere, e.g. [1–3, 5–7]. When used with global RBFs, this results in convergence rates higher than any polynomial order for smooth solutions. These kernels have also been shown to be better than finitely-smooth kernels



(a) (b)

Figure 2: (a) Illustration of a typical collection of patches (outlined in blue) for a quasi-uniformly distributed node set of $N = 4096$ nodes (marked in black) on $\mathbb{S}^2$. The centers of the patches are marked in red and there are roughly 100 nodes per patch. (b) Illustration of one of the PU weight functions (8) for the patches from part (a).

even when the solutions are not smooth [2]. However, the advantages of using infinitely-smooth kernels over finitely-smooth ones with RBF-FD was recently called into question [13, 24]. Indeed, recent work on Euclidean geometries [13, 39] has demonstrated that it is more beneficial to use the finitely-smooth polyharmonic spline (PHS) kernels in the context of RBF-FD methods, augmented with a moderate-degree polynomial. These kernels take the form $\phi(r) = r^{2k} \log r$, $k \geq 1$, or $\phi(r) = r^{2k+1}$, $k \geq 0$. The choice between these two kernels is often made depending on the dimension of the interpolation problem for theoretical reasons, while the choice of k controls the smoothness of the kernels, and hence their accuracy [29].

In this work, we use the IMQ kernel for the global RBF method in order to benefit from the higher-than-polynomial convergence rates for smooth problems. For the local RBF and RBF-PU methods we follow the recent work of [13, 24, 39] and use the PHS kernel $\phi(r) = r^{2k+1}$ in conjunction with spherical harmonics, as given in (4). We choose the degree of the spherical harmonics as

$$L = \left\lfloor \frac{1}{2}(\sqrt{n} - 1) \right\rfloor,$$

and the order of the PHS as $k = L$. For the many test problems we considered, these parameters produced excellent results. Choosing $k = L$ also coincides with theoretical results on solvability of the PHS interpolation problem [29]. In contrast, recent work [13, 24, 39] has used polynomials of a much higher-degree than that required to prove solvability.

3. RBFs for SL advection on a sphere

As discussed in the introduction, SL advection is a technique for solving the transport equation (1). The primary idea of SL advection is to use the Lagrangian frame to find upwind directions for which q moves, and use an Eulerian set of nodes to perform interpolations of q. The upwinding step ensures that the numerical domain of dependence matches the physical domain of dependence, while the interpolation step implicitly adds a small amount of diffusion (which decreases as the Eulerian node set is refined) that stabilizes the solution. The presence of an Eulerian frame avoids spatial resolution issues and allows for high-order accuracy, a feature that is difficult to achieve in the Lagrangian frame without remeshing, or redistributing nodes.

The upwinding is done as follows: assume that a *fictitious* particle arrived at each of the Eulerian nodes. This fictitious particle carried with it (advected) some amount of the scalar q. If we can find the *departure point* of the fictitious particle, and find the value of q there, this must be the amount of q brought forward to the Eulerian node. To find the departure point, we trace back the fictitious particle along the flow field $\boldsymbol{u}$. To find q at the departure point, we interpolate the scalar field to the departure point; this interpolation is done to transfer information between the two frames. This procedure is laid out explicitly in Algorithm 1.

More details on the general SL advection procedure can be found, for example, in [16]. In the following subsections, we discuss the different aspects of the algorithm as it pertains to advection on $\mathbb{S}^2$. We note that the algorithm is easily adapted to advection on spheres of different radii.

Algorithm 1 Semi-Lagrangian advection on $\mathbb{S}^2$

Input: Velocity field $\boldsymbol{u}(\boldsymbol{x},t)$ tangent to $\mathbb{S}^2$; initial scalar field $q(\boldsymbol{x},0)$; $X=\{\boldsymbol{x}_j\}_{j=1}^N$, $X\subset\mathbb{S}^2$; final time t_f; time-step Δt.
Set $\mathbf{q}_X^0=\{q(\boldsymbol{x}_j,0)\}_{j=1}^N$, $t=0$, and $m=0$.
while $t\leq t_f$ **do**
  For $j=1,\ldots,N$, trace back $\boldsymbol{x}_j$ to time t to find departure point $\boldsymbol{\xi}_j^m$.
  Interpolate $\mathbf{q}_X^m$ to $\Xi^m=\{\boldsymbol{\xi}_j^m\}_{j=1}^N$ to obtain $\mathbf{q}_\Xi^m$.
  Set $\mathbf{q}_X^{m+1}=\mathbf{q}_\Xi^m$, $m=m+1$, and $t=m\Delta t$.
end while

3.1. Eulerian node sets

Central to all SL advection schemes is a fixed set of Eulerian nodes $X=\{\boldsymbol{x}_j\}_{j=1}^N$ over the given domain that is used to interpolate the advected quantity to the Lagrangian fictitious particles. The approximate solution to the PDE will ultimately be computed only at the set of Eulerian nodes X. Since the interpolation schemes used in this study are all based on RBFs, the Eulerian nodes are not required to live on a grid or mesh, and can be chosen however we wish for our application. In this study, we use node sets that provide near optimal resolution over $\mathbb{S}^2$. Since $N=20$ nodes is the maximum number that can be exactly equally distributed over $\mathbb{S}^2$, one is resigned to using node sets that are only quasi-uniformly distributed over $\mathbb{S}^2$. These node sets, which can be generated from a variety of algorithms [40], have the property that the average spacing between nodes, h, satisfies $h\sim 1/\sqrt{N}$. In the results presented here we have used maximum determinant nodes [41] and icosahedral nodes [42]. While the latter of these sets forms a natural grid, we do not use this fact in our algorithms. All of these node sets, including many others, are available in the `SpherePts` software package [43].

3.2. Trajectory reconstruction

The process of tracing fictitious particles back along the velocity field $\boldsymbol{u}$ is known as *trajectory reconstruction*. Let $\boldsymbol{\xi}_j(t)\in\mathbb{S}^2$ denote the position of a fictitious particle in the Lagrangian frame as a function of time such that $\boldsymbol{\xi}_j(t_{m+1})=\boldsymbol{x}_j$, for some time $t=t_{m+1}>0$. Here $\boldsymbol{x}_j$ is the j^{th} node in Eulerian node set $X=\{\boldsymbol{x}_i\}_{i=1}^N\subset\mathbb{S}^2$. Then we find the departure point of this particle at time $t=t_m=t_{m+1}-\Delta t$ by solving the simple ODE

$$\frac{d\boldsymbol{\xi}_j}{dt}=\boldsymbol{u},\ \boldsymbol{\xi}_j(t_{m+1})=\boldsymbol{x}_j, \tag{10}$$

backward in time over the interval $[t_m,t_{m+1}]$. Here Δt defines the interval over which we trace back the fictitious particle from $\boldsymbol{x}_j$, and it also defines the time-step for solving the advection equation (1).

When the velocity field $\boldsymbol{u}$ is known for all time and space (as we assume in this study) the trajectory ODE (10) can be solved in a straightforward manner using a variety of techniques. However, since we use the Cartesian coordinate representation of the velocity field $\boldsymbol{u}$ in the trajectory reconstruction, the particles may not reside on the sphere at time $t=t_m$. Similar to [44], which also uses Cartesian coordinates (albeit in a fully Lagrangian scheme), we have found that the particles remain very close to the sphere when a high-order integrator is used. In our experiments, we found that both the standard fourth-order Runge-Kutta (RK4) method and Fehlberg's fifth-order Runge-Kutta (RK5) scheme [45], worked well. We ultimately decided on the RK5 scheme since we found that it allowed for larger time-steps than RK4. We also found that orthogonally projecting the fictitious particles exactly back to the surface of the sphere at each stage of the RK scheme improved the overall accuracy of the method.

The trajectory reconstruction procedure is repeated for $j=1,\ldots,N$ giving a set of N departure points at time $t=t_m$, which we denote by $\Xi^m=\{\boldsymbol{\xi}_j^m\}_{j=1}^N$ and refer to as the *Lagrangian point set*. It is at these points that we need to compute q. However, q is only (approximately) known at the nodes in X and in general $X\neq\Xi^m$, for any $m>0$. Thus, we need to interpolate q from X to Ξ^m.

3.3. Interpolation

Let $\mathbf{q}_X^m$ denote the vector containing the approximate solution to (1) at the Eulerian node set X (with the same ordering as the nodes in X) at time $t = t_m$. Below we discuss the details of interpolating $\mathbf{q}_X^m$ to the Lagrangian point set Ξ^m using the three different RBF interpolation methods presented in Section 2. These methods can all be used in the interpolation phase of Algorithm 1.

3.3.1. Global RBF interpolation

The most straightforward interpolation scheme is built on the global RBF interpolant (2). To compute this interpolant, we first solve the linear system (3), with the right hand side set equal to $\mathbf{q}_X^m$, to determine the interpolation coefficients $\mathbf{c}_X$. We then evaluate the interpolant at the points in Ξ^m, resulting in a vector of approximate values, which we denote by $\mathbf{q}_\Xi^m$. Finally, we set $\mathbf{q}_X^{m+1} = \mathbf{q}_\Xi^m$ to complete a time step.

The coefficient matrix A_X in (3) does not change throughout the simulation (since the nodes X are fixed for all time), thus allowing us to factorize A_X once as a preprocessing step. We use the inverse multiquadric (IMQ) radial kernel, $\phi(r) = (1 + (\varepsilon r)^2)^{-1/2}$, in this study, which is positive-definite, so that the matrix A_X can be factorized using Cholesky factorization. Since A_X is dense, this requires $O(N^3)$ operations initially. Each time step then requires $O(N^2)$ operations to solve (3) for each $\mathbf{q}_X^m$. The evaluation of (2) at Ξ^m requires an additional $O(N^2)$ computations. While this cost becomes prohibitive for large N, the global RBF method remains competitive for smooth solutions because of its high accuracy—faster than any algebraic order when the target function is sufficiently smooth [32]—as illustrated in the numerical results section.

3.3.2. Local RBF interpolation

The local RBF interpolation scheme is more complicated to implement than the global case. The first step of this scheme requires determining the n-point local stencil from X that will be used for interpolating to each of the points in Ξ^m. The approach we take is to determine the nearest neighbor in X for each $\boldsymbol{\xi}_j^m \in \Xi^m$, $j = 1, \ldots, N$. The nearest neighbor determines the center point for the stencil that will be used for the interpolation. Letting $\mathcal{K}_j$, denote the index of the nearest neighbor in X for $\boldsymbol{\xi}_j$, the interpolation stencil is then denoted as $X_{\mathcal{K}_j}$. After these stencils have been determined, we perform the interpolations to the points in Ξ^m. These interpolations are done, for each $\boldsymbol{\xi}_j^m$, $j = 1, \ldots, N$, using (4), with $k = \mathcal{K}_j$, which results in the vector of approximate values $\mathbf{q}_\Xi^m$. As in the global case, we then set $\mathbf{q}_X^{m+1} = \mathbf{q}_\Xi^m$ to complete the time step.

The computation of the local interpolant for $\boldsymbol{\xi}_j$ requires solving the linear system (5) with the right hand side set equal to $\tilde{\mathbf{q}}_{X_{\mathcal{K}_j}}^m$. While it may be possible for some points in Ξ^m to share the same stencil, in general there will be $O(N)$ of these linear systems to solve. The matrices for these linear systems can all be pre-computed and factorized as a pre-processing step since the points in X do not change and all the stencils can be determined *prior* to the simulation. Unlike the global case, we use LU-factorizations of the matrices, since they are no longer positive-definite due to the inclusion of the polynomial terms and the use of the PHS kernel. The method initially requires $O(n^3 N)$ operations to factorize the linear systems for each stencil, then the cost is $O(n^2 N)$ operations to determine the interpolation coefficients of the local interpolants for all the points in Ξ^m and to evaluate the local interpolants to obtain $\mathbf{q}_\Xi^m$. To determine the nearest neighbors for each $\boldsymbol{\xi}_j^m \in \Xi^m$ in X, we use a kd-tree of the nodes X. This requires $O(N \log N)$ operations to construct initially, and $O(\log N)$ operations for each nearest neighbor search on average. Since n is chosen independent of N and typically $n << N$, the method has an asymptotic computational cost of $O(N \log N)$ per time-step, with a one-time initial cost of $O(N \log N)$.

Even when including the cost of kd-tree searches for nearest neigbhors, the local method is much more computationally efficient than the global RBF method. With this approach, it is possible to obtain high-order methods with algebraic convergence rates that depend on the stencil size n and the degree of the appended spherical harmonics.

3.3.3. RBF-PU interpolation

The first step of the RBF-PU interpolation scheme is to construct the local interpolants s_ℓ in (9) over the patches (spherical caps) Ω_ℓ, $\ell = 1, \ldots, M$. This requires solving M linear systems of the form (5), where the node sets and right hand sides used are determined by which nodes from X are included in patch Ω_ℓ. Once all the local interpolants are constructed, they are combined into the globally-smooth interpolant (9),

which is then evaluated at all the points in Ξ^m. Note that for a given $\boldsymbol{\xi}_j^m \in \Xi^m$, the sum in (9) only needs to be taken over terms corresponding to the patches Ω_ℓ containing $\boldsymbol{\xi}_j$. As with the global and local methods, the result is a vector of approximate values $\mathbf{q}_\Xi^m$, which are assigned to $\mathbf{q}_X^{m+1}$ to complete the time step.

The computational complexity of the RBF-PU scheme is determined by how the patches Ω_ℓ, $\ell = 1, \ldots, M$, are distributed; this also directly effects the accuracy of the interpolants. Since the node sets X are assumed to be quasi-uniformly distributed, it makes sense to distribute the patches in a quasi-uniform manner to control the computational cost. Distributing the patches in a quasi-uniform manner is equivalent to distributing the set of patch centers $\{\boldsymbol{\omega}_\ell\}_{\ell=1}^M$ in a quasi-uniform manner. For these sets we use minimum energy (ME) points, which are computed by arranging the points in the set such that the Reisz energy (with a power of 2) of the set on $\mathbb{S}^2$ attains a minimum [40]. We use the pre-computed quasi-ME point sets from [43], which are available for $2 \leq M \leq 5000$.

What remains to determine the distribution of patches is M, the total number of patches (centers), and R_ℓ, $\ell = 1, \ldots, M$, the radii of the patches. We determine these quantities by setting 1) the number nodes, n, that each patch is to approximately contain and 2) the average number of patches, a, that a given node is to belong to. The first of these controls the computational cost for each patch and effects the accuracy of the local interpolants s_ℓ, as each interpolant will be based on approximately n nodes. The second value a controls the computational cost of evaluating the global RBF-PU interpolant (9), as it determines how much overlap there is amongst the patches. This value also controls the locality of the interpolant.

We use n to determine the radii of the patches as follows. If X contains N quasi-uniformly distributed nodes and there are to be approximately n nodes per patch, then the area per node over the entire sphere should approximately equal the area per node over patch Ω_ℓ, i.e. $4\pi/N \approx \pi R_\ell^2/n$, where R_ℓ is the radius of Ω_ℓ. This gives the estimate $R_\ell \approx 2\sqrt{n/N}$. Since the centers of the patches are also assumed to be quasi-uniformly distributed, the radii can all be chosen in the same way. We thus set one radius,

$$R = 2\sqrt{n/N}, \tag{11}$$

for every patch. We use a to determine the number of patches M as follows. Since the patch centers $\{\boldsymbol{\omega}_\ell\}_{\ell=1}^M$ are quasi-uniformly distributed, the same area arguments as above can be used to arrive at the estimate $4\pi/M \approx \pi R^2/a$. Solving for M in this equation and using (11), we obtain the following approximation (noting that M should be an integer)

$$M = \lceil aN/n \rceil. \tag{12}$$

Figure 2 illustrates the patches for $N = 4096$, $n = 100$, and $a = 2.5$.

We are now ready to estimate the computational cost of the RBF-PU interpolation scheme per time-step. Similar to the local RBF scheme, the matrices in the M linear systems (5) for determining s_ℓ, $\ell = 1, \ldots, M$ do not change per time step and can thus be LU decomposed as a preprocessing step. Since each linear system contains approximately n nodes and there are M linear systems, this cost is $O(n^3 M)$, which can be estimated as $O(an^2 N)$ from (12). Using the LU factorizations, the M linear systems (5) can be solved in $O(anN)$ operations per time step. The evaluation of any of the interpolants s_ℓ in (9) at a point on $\mathbb{S}^2$ requires $O(n^2)$ operations. Each point of Ξ^m will, on average, belong to a patches, so that the sum in (9) will, on average, only involve a non-zero terms. Thus, the computational cost of evaluating the RBF-PU interpolant (9) at Ξ^m is $O(an^2 N)$. To determine which patches a given point in Ξ^m belongs to, we use a kd-tree of the patch centers, which only has to be constructed once at a cost of $O(M \log M) = O(aN/n \log(aN/n))$ operations and can then be searched each time-step at a cost of $O(N \log(aN/n))$. Since n and a are chosen independent of N and typically $a << n << N$, the method has an asymptotic computational cost of $O(N \log N)$ per time-step, with a one time initial cost of $O(N \log N)$.

Remark 1. An important special case of SL advection is when the problem is restricted to the real line and forward Euler for the trajectory calculation. If we restrict $\Delta t \sim h$ where $h \propto \frac{1}{N}$ is the node spacing on the real line, and use linear interpolation in conjunction with a forward Euler backtrace to find $\mathbf{q}_\Xi^m$, the SL advection scheme reduces to the classical first-order upwind scheme. Even if one lifts the CFL restriction on Δt in this setting, the SL advection scheme with linear interpolation is unconditionally stable [16]. Unfortunately, it is unclear how to prove stability for either high-order polynomial interpolation or RBF interpolation. However, in practice, all of our RBF methods are stable for values of Δt much larger than the CFL condition dictates.

4. Results

We investigate various properties of the three new methods proposed in this article on three standard test cases for transport on the sphere from the literature: solid-body rotation of a cosine bell from [25] and deformational flow with two different initial conditions from [26]. We test our methods using quasi-uniformly distributed node sets X of various sizes N. For the global RBF method, we use maximum determinant node sets [41] of sizes $N = 3136$, 4096, 5041, 6084, 7744, 9025, 10000, 11881, 13689, and 15129. For the local RBF and RBF-PU methods, we use equidistributed icosahedral nodes [46] of sizes $N = 2562$, 5762, 10242, 23042, 40962, and 92162. All these node sets are available from [43]. In the results that follow, we estimate the convergence of our method verses $\sqrt{N}$ since this is roughly inversely proportional to the average spacing between the nodes (see Section 3.1 for a discussion). For the local RBF and RBF-PU methods we present results for $n = 17$, 31, 49, and 84. These are common values used in RBF-FD methods for advective PDEs on the sphere [6, 7], and thus provide easy comparisons with those methods.

The main focus of the investigation is on accuracy of the methods, which we measure using the relative ℓ_2 and ℓ_∞ norms. We also give results on the dissipation and dispersion errors using a visual inspection of the errors and the *a posteriori* quantitative measures derived in [47] (see also [48]). These measures are derived from a decomposition of the mean square error of the numerical solutions and are given as follows:

$$\text{Dissipation error: } \left[\sigma(q) - \sigma(q_X)\right]^2 + \left[\bar{q} - \bar{q_X}\right]^2, \tag{13}$$

$$\text{Dispersion error: } 2\left[\sigma(q)\sigma(q_X) - \frac{1}{4\pi}\int_{\mathbb{S}^2}(q-\bar{q})(q_X - \bar{q_X})dS\right], \tag{14}$$

where q is the exact solution and q_X is the approximate solution, and bars on these variables denote the mean, while σ denotes their standard deviation over $\mathbb{S}^2$. In the results, we divide these errors by the mean square error of the solution to give a relative measure. Finally, we present results on the conservation properties of the methods by plotting the absolute errors in the total mass: $\left|\frac{1}{4\pi}\int_{\mathbb{S}^2}(q - q_X)dS\right|$. All of the above quantities are computed using discrete approximations to the continuous operators defining them on $\mathbb{S}^2$, which requires computing numerical approximations to surface integrals over $\mathbb{S}^2$ using the nodes X. To do these computations we use the sixth-order kernel-based meshfree quadrature method from [49].

4.1. Solid body rotation of a cosine bell

As a first test problem, we consider the standard Test Case 1 from Williamson et al. [25]. The components of the steady velocity field for this test case in spherical coordinates ($-\pi \le \lambda \le \pi$, $-\pi/2 \le \theta \le \pi/2$) are given by

$$u(\lambda,\theta) = \sin(\theta)\sin(\lambda)\sin(\alpha) - \cos(\theta)\cos(\alpha), \quad v(\lambda,\theta) = \cos(\lambda)\sin(\alpha).$$

This velocity field results in solid body rotation at an angle of α with respect to the equator. In all of our tests, we use one of the standard choices of $\alpha = \pi/2$, which corresponds to advection of the initial condition directly over the north and south poles. Since our method works purely in Cartesian coordinates, we use a change of basis to obtain the same velocity field in these coordinates. The initial condition is taken as a compactly supported cosine bell centered at $(1, 0, 0)$:

$$q(\boldsymbol{x}, 0) = \begin{cases} \frac{1}{2}\left(1 + \cos\left(\frac{\pi r}{R_\mathrm{b}}\right)\right) & \text{if } r < R_\mathrm{b}, \\ 0 & \text{if } r \ge R_\mathrm{b}, \end{cases} \tag{15}$$

where $\boldsymbol{x} = (x, y, z)$, $r = \arccos(x)$, and the support is set as $R_\mathrm{b} = 1/3$. This initial condition has a jump in the second derivative at $r = R_\mathrm{b}$, which makes the test susceptible to both dispersive and diffusive errors. The test calls for simulating the advection of the initial condition to the final time of $T = 2\pi$, which corresponds to one full revolution of the bell over the sphere. For the convergence tests, we set the time step to $\Delta t = \pi/10$, which is necessary so that spatial errors dominate for all the finest node sets. On the finest node set for the global RBF method, this time-step gives a CFL number of approximately 12, while for the finest node set for the local RBF and RBF-PU methods this gives a CFL number of approximately 28.

Convergence results in relative ℓ_2 and ℓ_∞ norms for increasing N are shown in Figure 3 for the three methods, together with the estimated rates of convergence. From the top row of this plot, we see that the

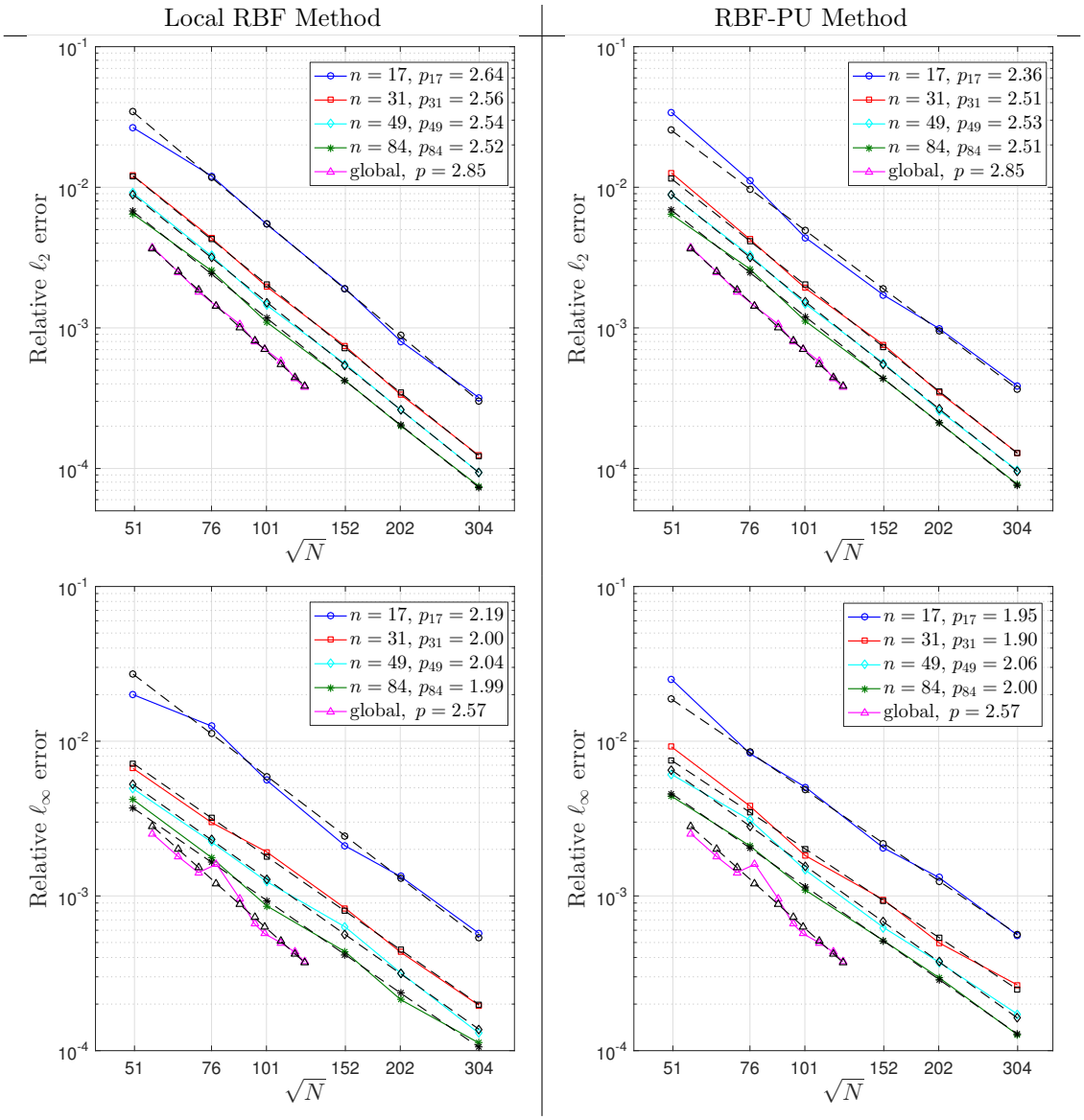


Figure 3: Convergence results of the relative ℓ_2 (top row) and ℓ_∞ (bottom row) errors for the solid body rotation of a cosine bell test case after one revolution. The results for the global RBF method are included in all plots for comparison purposes. The dashed lines are the lines of best fit to the data (without the first point included) of the form $C_n N^{-p_n/2}$). The values of p_n, which estimate the order of accuracy of the methods, are listed in the legend and the subscript on p is dropped for the global RBF method.

ℓ_2 error for the local RBF and RBF-PU methods appears to converges at a rate close to 2.5 for all n, while the convergence for the global RBF method appears closer to 3. The ℓ_∞ error is shown in the bottom row and we see that the convergence rates are now around 2 for the local RBF and RBF-PU methods and 2.5 for the global RBF method. These rates of convergence are dictated by the smoothness of the initial condition, which is only $C^1(\mathbb{S}^2)$, and are consistent with the convergence results for the global RBF collocation and RBF-FD methods for this same problem [1, 6]. For the local RBF and RBF-PU methods, we see that increasing n leads to a decrease in the error, but not in the convergence rates (again because of the limited smoothness of the solution). We also see that the errors for the global method are smaller for similar values of N.

Figure 4 displays the dissipation and dispersion errors (13)–(14). Here we have fixed N at 40962 for the local RBF and RBF-PU methods and plotted dissipation and dispersion errors against with n, the local

stencil/patch size. Also included in the plots are the results for the global RBF method (dashed line) with $N = 15129$. From the plots, we see that dispersion errors dominate the numerical solutions for all the methods, which can be expected since the initial condition is only $C^1(\mathbb{S}^2)$. We also see that for the local RBF and RBF-PU methods increasing n (which increases the order of accuracy of these methods) leads to a decrease in both dissipation and dispersion errors, with the decrease in the former being much more pronounced.

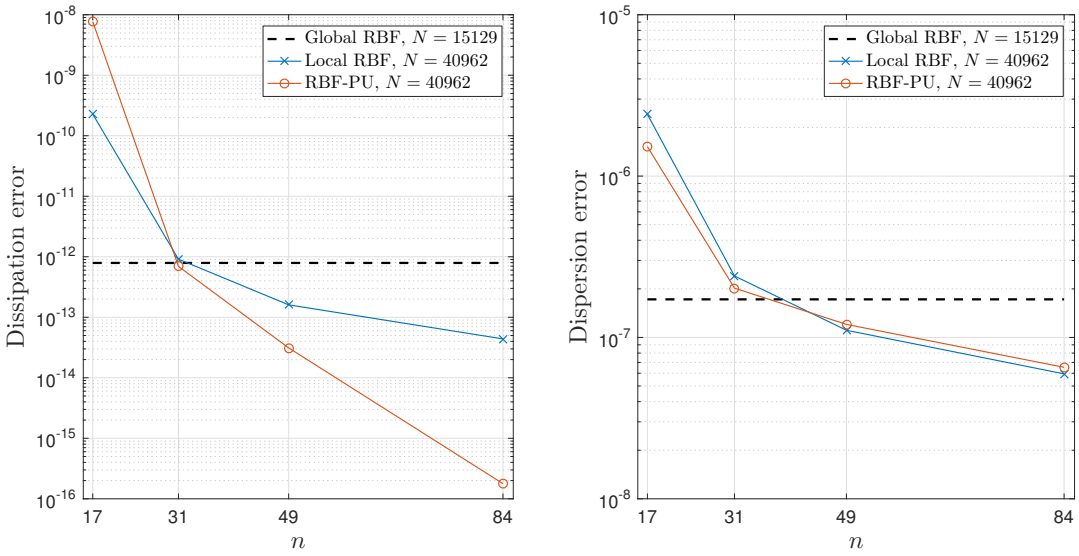
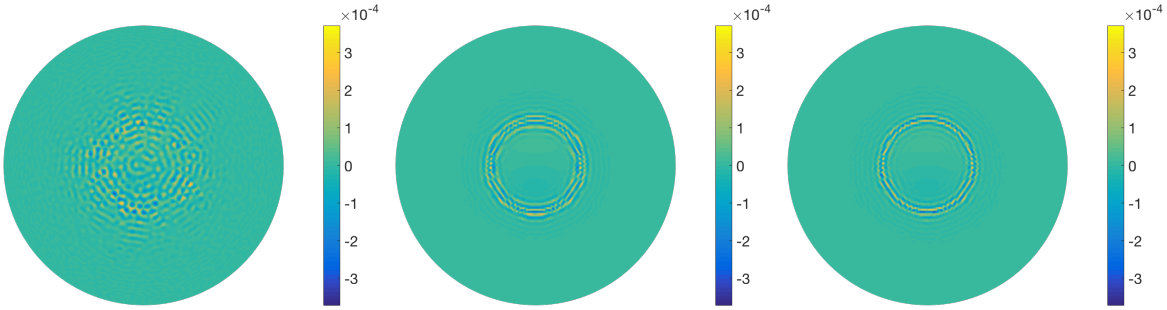


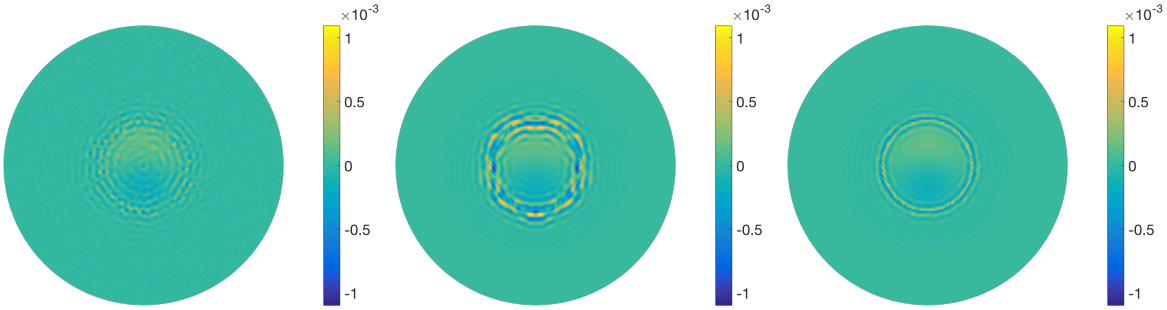
Figure 4: Relative dissipation (left) and dispersion (right) errors (13)–(14) for the solid body rotation of a cosine bell test case after one revolution as n increases in the local RBF and RBF-PU methods. The global RBF method does not have a dependence on n and is included as a dashed line for reference.

We further explore the dissipation/dispersion errors of the methods by plotting the difference in the exact and approximate solutions after one ($T = 2\pi$) and ten ($T = 20\pi$) revolutions of the cosine bell. This test was suggested in [6] to examine whether the errors remain well-localized to the support of the bells over time, and to give a visual indication of the dissipation/dispersion errors. Figure 5 displays the results for the global RBF method for the case of N=15129 and the local RBF and RBF-PU methods, both for the case of $N = 40962$ and $n = 49$. These values were selected since the errors were of similar magnitude. We see from the first row of the figure (one revolution) that the errors for the local RBF and RBF-PU methods are localized around the discontinuities in the second derivative of the solution, with the error being lower and more localized for the RBF-PU method. The errors for the global RBF method are spread more over the support of the entire bell. None of the methods display a dispersive wave-train over the whole sphere. After ten revolutions (second row of the figure) the errors for the local RBF and RBF-PU methods increase by about a factor of 4, but still remain localized around the discontinuities in the solution. The RBF-PU method again displays better localization of the errors, or lower dispersive-type errors. The dominant errors for the global method also remain restricted to the support of the bell, but only increase by about a factor of 1.5. The results for the local RBF and RBF-PU methods are qualitatively similar to those in [6] for the RBF-FD method with hyperviscosity stabilization.

Time traces of the absolute value of the mass conservation errors over ten revolutions ($T = 20\pi$) of the cosine bell are displayed in Figure 6. We again use $N = 40962$ and $n = 49$ for the local RBF and RBF-PU methods and $N = 15129$ for the global method. From this figure we see that the conservation errors for the local RBF and RBF-PU methods increase slightly over the integration time, with the growth of the local method being larger, while the errors in the global method remain consistent over the integration interval. While not presented here, we found that increasing n to 84 for the local RBF and RBF-PU methods had a marginal effect on decreasing the conservation errors.



(a) Error global RBF: $T = 2\pi$ (b) Error local RBF: $T = 2\pi$ (c) Error RBF-PU: $T = 2\pi$



(d) Error global RBF: $T = 20\pi$ (e) Error local RBF: $T = 20\pi$ (f) RBF-PU: $T = 20\pi$

Figure 5: Pseudocolor plots of the errors, $q_X - q$, on the sphere for solid body rotation test case of the cosine bell. The top row shows the errors after one revolution ($T = 2\pi$), while the bottom row shows the errors after ten revolutions ($T = 20\pi$). (a) & (d) show the results for the global RBF method using $N = 15129$, (b) & (d) show the results for the local RBF method using $N = 40962$ and $n = 49$, and (c) & (e) show the results for the RBF-PU method using $N = 40962$ and $n = 49$. Note the color scales differ for the two values of T. The view for all plots is from the initial center of the bell.

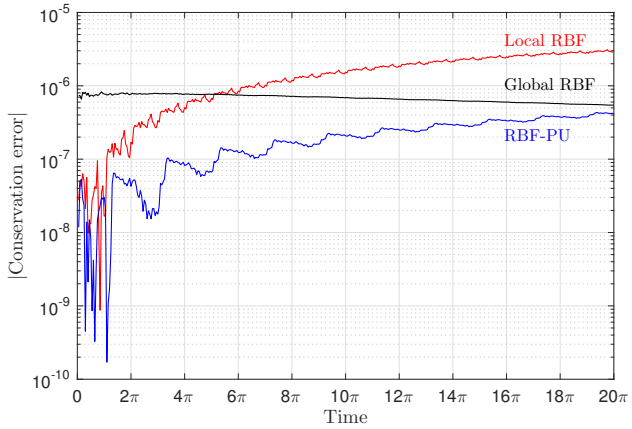


Figure 6: Time traces of the mass conservation errors over ten revolutions of the cosine bell in the solid body rotation test case. For the local RBF and RBF-PU methods $N = 40962$ and $n = 49$, while for the global method $N = 15129$. For this test we used $\Delta t = \pi/20$.

4.2. Deformational flow

For the second test problem, we consider the deformational flow test case from [50]. The components of the velocity field for this test are given in spherical coordinates as

$$u(\lambda, \theta, t) = \frac{10}{T} \cos\left(\frac{\pi t}{T}\right) \sin^2\left(\lambda - \frac{2\pi t}{T}\right) \sin(2\theta) + \frac{2\pi}{T} \cos\theta, \tag{16}$$

$$v(\lambda, \theta, t) = \frac{10}{T} \cos\left(\frac{\pi t}{T}\right) \sin\left(2\lambda - \frac{2\pi t}{T}\right) \cos(\theta), \tag{17}$$

where $T = 5$. This flow field is designed to deform the initial condition up to time $t = 2.5$ and then reverse so that the solution is returned to its initial position and value at time $t = 5$. This value serves as the final time for the simulation. As before, we use a simple change of basis to convert this velocity field into Cartesian coordinates.

The following two initial conditions are considered:

1. Two cosine bells: $q(\boldsymbol{x}, 0) = 0.1 + 0.9(q_1(\boldsymbol{x}, 0) + q_2(\boldsymbol{x}, 0))$, where for $j = 1, 2$

$$q_j(\boldsymbol{x}, 0) = \begin{cases} \frac{1}{2}\left(1 + \cos\left(2\pi \cos^{-1}\left(\boldsymbol{x} \cdot \boldsymbol{p}_j\right)\right)\right) & \text{if } \cos^{-1}\left(\boldsymbol{x} \cdot \boldsymbol{p}_j\right) < \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

2. Two Gaussian bells: $q(\boldsymbol{x}, 0) = 0.95\left(e^{-5\|\boldsymbol{x}-\boldsymbol{p}_1\|_2^2} + e^{-5\|\boldsymbol{x}-\boldsymbol{p}_2\|_2^2}\right)$.

In both cases, $\boldsymbol{p}_1 = \left(\frac{\sqrt{3}}{2}, \frac{1}{2}, 0\right)$ and $\boldsymbol{p}_2 = \left(\frac{\sqrt{3}}{2}, -\frac{1}{2}, 0\right)$. Like the previous example, the first initial condition is a good test of the sensitivity of the three methods to dispersion errors as it is only $C^1(\mathbb{S}^2)$. The second initial condition is $C^\infty(\mathbb{S}^2)$ and hence is a good test of the maximum convergence rates of the three methods. For all experiments involving the local RBF and RBF-PU methods, we set $\Delta t = 1/10$. This gives a CFL number of approximately 27 on the finest node set. For the global RBF method we use $\Delta t = 1/10$ for the cosine bells and $\Delta t = 1/40$ for the Gaussian bells, which results the respective CFL numbers of 12 and 3 on the finest node sets. All time-steps were chosen so that spatial errors dominate for all values of N.

4.2.1. Results for the cosine bells

Figure 7 displays the convergence results in relative ℓ_2 and ℓ_∞ norms for increasing N for the three methods, together with the estimated rates of convergence. We see that the convergence rates for this test case are a bit higher than the solid body rotation test, but that the errors are larger for a given N. The rates of convergence are again limited by the smoothness of the solution. The larger errors for this test are expected as the solution undergoes much more dramatic changes over the simulation period than the solid body rotation test. As in the solid body rotation test, the errors for the local RBF and RBF-PU methods are larger than the global method for a given N, but because the former methods are more computationally efficient, we can push them to larger N and reach smaller overall errors. At the end of this section, we compare the accuracy of all three methods to their runtime costs to get a better picture of their overall efficiency.

The relative dissipation and dispersion errors (13)–(14) are displayed in Figure 4. As in the previous test, N is fixed for the local RBF and RBF-PU methods and the errors are plotted against n. The global RBF method with $N = 15129$ is displayed as a dashed line. The figure shows that dispersion errors again dominate the numerical solutions for all the methods and that increasing n for the local RBF and RBF-PU methods leads to a decrease in these errors. Visual depictions of these dispersive errors are given in Figure 9. Part (a) shows the initial condition and final solution for this test case and parts (b)–(d) shows the difference between this solution and the numerical solutions at the final time for the three methods. The error for the global method (part (b)) are about 1.5 times higher than the local RBF and RBF-PU methods and appears to be much more dispersive in nature. Also, the error for the RBF-PU method is more localized to the discontinuities of the initial condition than both the global and local RBF methods.

Time traces of the mass conservation errors for the three methods over the simulation time are displayed in Figure 10 (a). For the local RBF and RBF-PU methods, we see that, after a relatively large initial growth, the errors level off around $t = 1$ and then start to grow very slowly towards the end of the simulation. The global RBF error is overall larger than the other two methods, but does not exhibit a discernible growth rate.

4.2.2. Results for the Gaussian bells

The convergence results for the relative ℓ_2 and ℓ_∞ errors for this test are displayed in Figure 11. We see from these plots that the norm of the errors for global RBF method appear to converge faster than any polynomial rate, which is expected since the solution is $C^\infty(\mathbb{S}^2)$ (see Section 2.1). The convergence rates of the errors for the local RBF and RBF-PU methods are also higher for this smooth test case. Unlike the

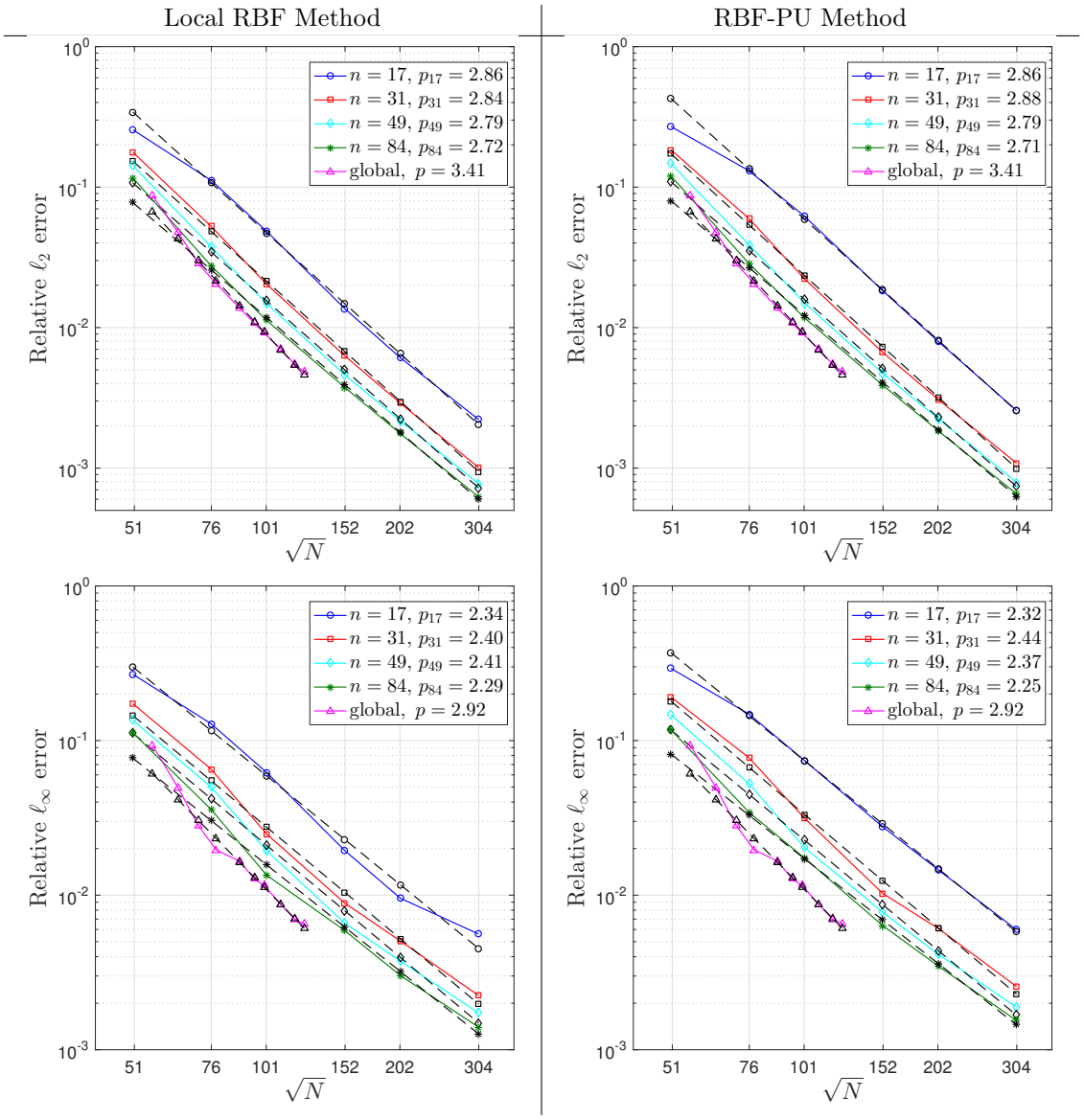


Figure 7: Convergence results of the relative ℓ_2 (top row) and ℓ_∞ (bottom row) errors the deformational flow of two cosine bells test case. The results for the global RBF method are included in all plots for comparison purposes. The dashed lines are the lines of best fit to the data (without the first point included) of the form $C_n N^{-p_n/2}$). The values of p_n, which estimate the order of accuracy of the methods, are listed in the legend and the subscript on p is dropped for the global RBF method.

two previous tests, we see that these convergence rates also increase as n increases. The RBF-PU method appears to have a higher convergence rate for the same n than the local RBF method, and the errors for the RBF-PU method are lower for each corresponding n and N value. This is likely due to the global smoothness of the RBF-PU interpolant.

Figure 12 displays the relative dissipation and dispersion errors just like the other test cases. We see that the dissipation and dispersion errors are smallest for the global method and largest for the local RBF method. We also see that increasing n for local and RBF-PU methods leads to a much larger decrease in both the dissipation and dispersion errors than the previous two test cases. We omit plots of the errors over the sphere since the dispersion errors are not a real issue for this test case.

Time traces of the mass conservation errors are displayed in Figure 10 (b). As with the dissipation/dispersion errors, the global method shows the best results followed by the RBF-PU method and then the local RBF method. All of these methods show a slight grow in the conservation errors towards the

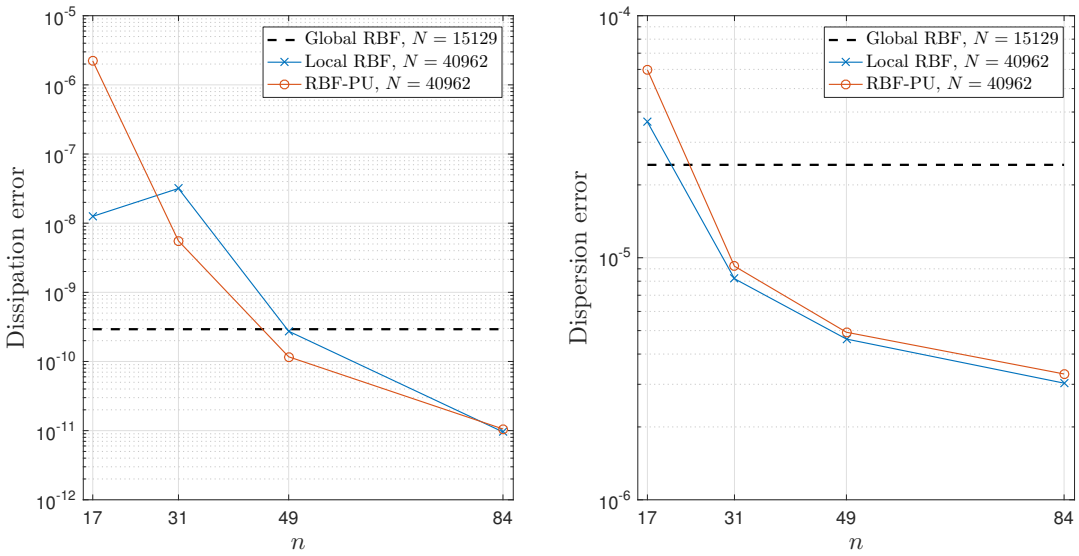
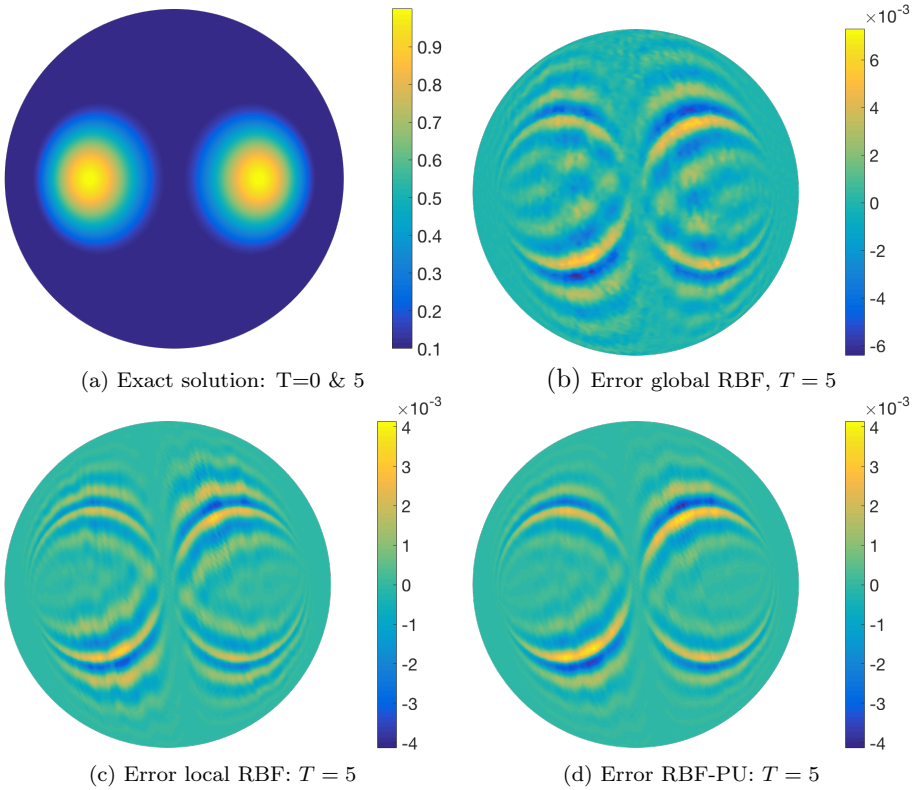


Figure 8: Relative dissipation (left) and dispersion (right) errors (13)–(14) for the deformational flow test case of two cosine bells after one revolution as n increases in the local RBF and RBF-PU methods. The global RBF method does not have a dependence on n and is included as a dashed line for reference.



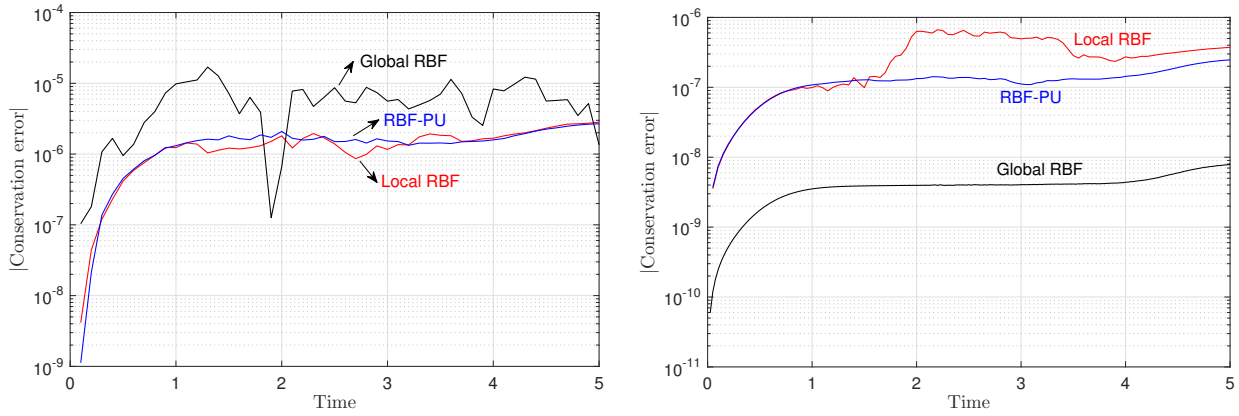
(a) Exact solution: T=0 & 5

(b) Error global RBF, $T = 5$

(c) Error local RBF: $T = 5$

(d) Error RBF-PU: $T = 5$

Figure 9: (a) Pseudocolor plots on the sphere of the initial and final solutions for the two cosine bells test. (b)–(c) Show pseudocolor plots of the error, $q_X - q$, at $T = 5$ for the global RBF, local RBF, and RBF-PU methods, respectively. For the global RBF method, $N = 15129$, while for the local and RBF-PU methods, $N = 40962$ and $n = 49$. Note that the color scale is about 1.5 times larger for the global RBF method. The view for all plots is from the midpoint between the centers of the two bells.



(a) Cosine bells (b) Gaussian bells

Figure 10: Time traces of the mass conservation errors over the simulation time for the deformational flow tests. For the local RBF and RBF-PU methods $N = 92162$ and $n = 49$ and $\Delta t = 1/10$ for both part (a) and (b), while for the global method $N = 15129$ and $\Delta t = 1/10$ for part (a) and $N = 15129$ and $\Delta t = 1/40$ for part (b).

end of the simulation time. While not reported here, increasing n in the local and RBF-PU methods does lead to a decrease in the conservation errors.

The convergence rates of the errors for the local RBF and RBF-PU methods are also higher for this smooth test case, as shown in first row Figure 11. Unlike the previous tests, we see that these convergence rates also increase as n increases. Also, the RBF-PU method has slightly higher convergence rates for the same n than the local RBF method, and the errors for the RBF-PU method are lower (up to an order of magnitude) for each corresponding n and N value. This is likely due to the global smoothness of the RBF-PU interpolant. The second and third rows of plots in Figure 11 show similar trends in the dissipation and dispersion errors. These results seem to indicate that for a smooth initial condition, having global smoothness in the interpolant can help decrease the overall error, in addition to the dissipation and dispersion errors. However, as we show next, this global smoothness comes with a higher computational cost.

4.3. Cost versus accuracy

To properly compare all three methods, it is important to look at their cost (measured in wall-clock time) versus accuracy. In this section we give such a comparison for the deformational flow test case using both the non-smooth and smooth initial conditions. We examine the simulation time (ignoring preprocessing costs) and accuracy of each method (in the relative ℓ_2 norm), and how these change with N and n. All tests were run on a Linux workstation with a 3 GHz Intel Core i7-3930K (12 logical cores) and 32 GB of RAM. All the codes were written and executed in MATLAB (version 2016a) in standard double precision, but without any explicit parallelization.

The results for the non-smooth cosine bells are displayed in Figure 13 (a). We see that for low accuracy, the global RBF method has a lower overall cost than both the local RBF and RBF-PU methods. For higher accuracies, the cost of the global method increases much more rapidly than the other two methods because of its $O(N^2)$ complexity and, for high enough accuracies, both the local RBF and RBF-PU methods are more efficient overall. Comparing the local RBF and RBF-PU methods, we see that, for a fixed n, the latter has a lower computational cost for a given accuracy. Also, for this non-smooth test case, it does not appear to offer much benefit in terms of cost to use large n with the local method, whereas the RBF-PU method shows slightly better efficiency with increasing n.

The results for the smooth Gaussian bells test case are given in Figure 13 (b). These plots clearly illustrate the advantage of the global RBF method with a smooth kernel over the local and RBF-PU methods when used on a problem with a smooth solution. Extrapolating out, we see that global method is able to reach a much smaller ℓ_2 error for same runtime. However, the global method will exhaust the memory resources of the machine much more quickly than the local and RBF-PU methods. The figure also shows that for a smooth initial condition, it pays to use larger values of n, with this being much more beneficial for the

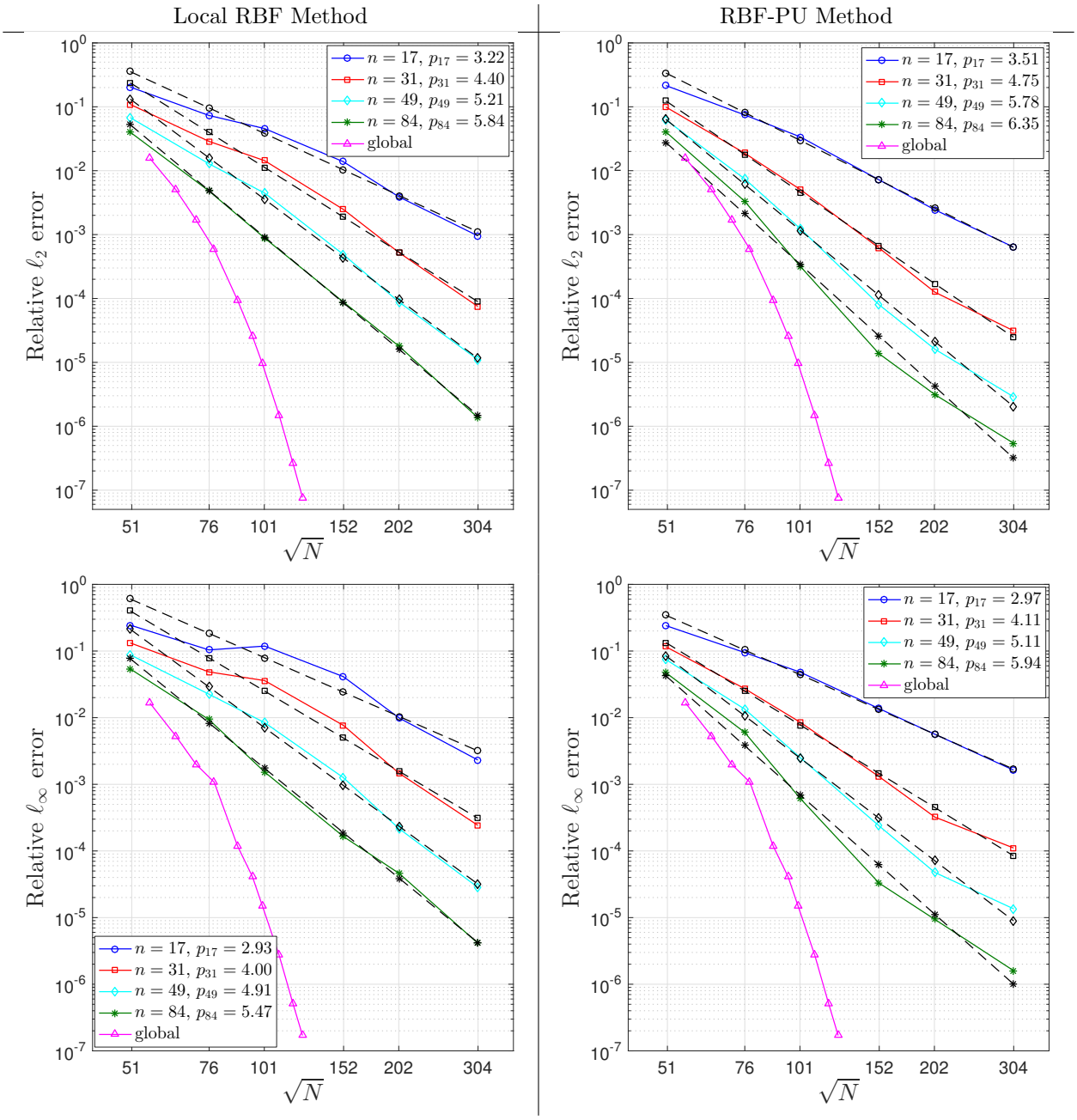


Figure 11: Convergence results of the relative ℓ_2 (top row) and ℓ_∞ (bottom row) errors the deformational flow of two Gaussian bells test case. The results for the global RBF method are included in all plots for comparison purposes. The dashed lines are the lines of best fit to the data (without the first point included) of the form $C_n N^{-p_n/2}$). The values of p_n, which estimate the order of accuracy of the methods, are listed in the legend. Lines of best fit are omitted for the global case since the convergence appears to be faster than any polynomial order.

RBF-PU method. Comparing the local and RBF-PU methods for this smooth initial condition, we see that overall, in our implementation, the RBF-PU method again gives higher accuracy for the same cost.

It is important to note that these are serial implementations of all the methods. The local RBF method is easily parallelized on SIMD architectures, allowing for speedups of 2-8 times over a serial implementation, as shown for the related RBF-FD method in [8, 9]. The RBF-PU method also has promising parallelization properties using its patch based structure. The global RBF method with infinitely-smooth kernels requires a domain-decomposition style approach to parallelization. We thus expect parallel versions of both the local

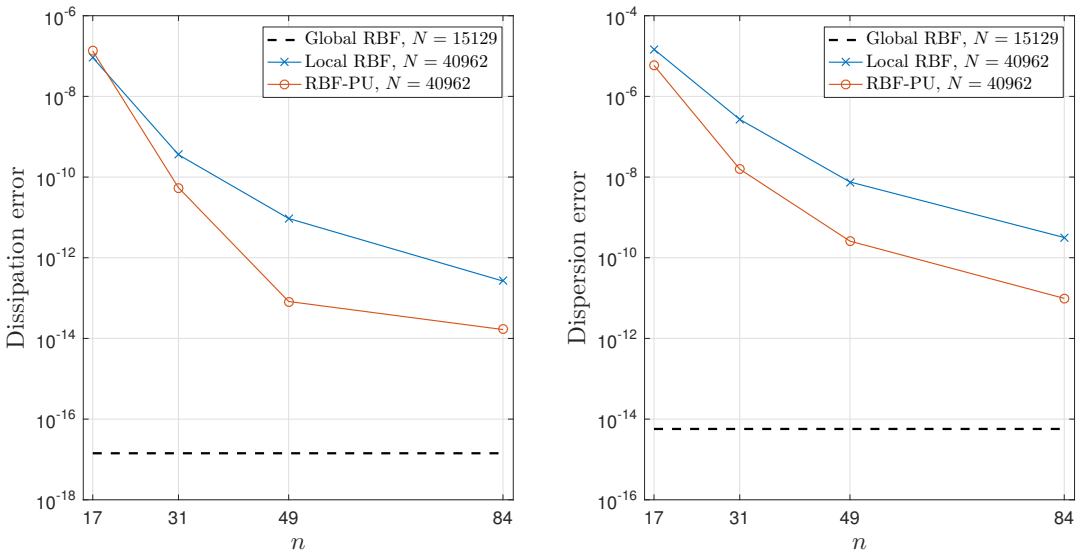


Figure 12: Relative dissipation (left) and dispersion (right) errors (13)–(14) for the deformational flow test case of two Gaussian bells after one revolution as n increases in the local RBF and RBF-PU methods. The global RBF method does not have a dependence on n and is included as a dashed line for reference.

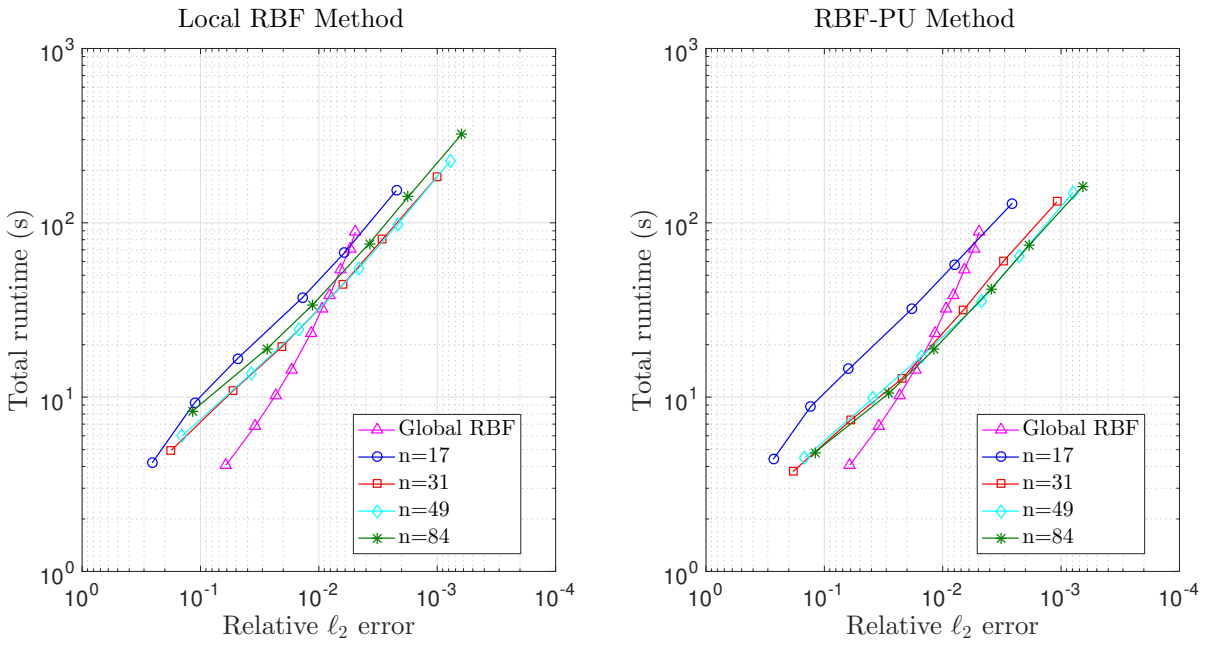
and RBF-PU methods to give much better results in cost vs. accuracy studies than presented here. We also note that, in all our tests, the global and RBF-PU methods were able to utilize Matlab's automatically multithreaded BLAS kernels more efficiently than the local RBF method, since these two methods inherently use more matrix-matrix multiplications (BLAS-3) operations.

5. Summary

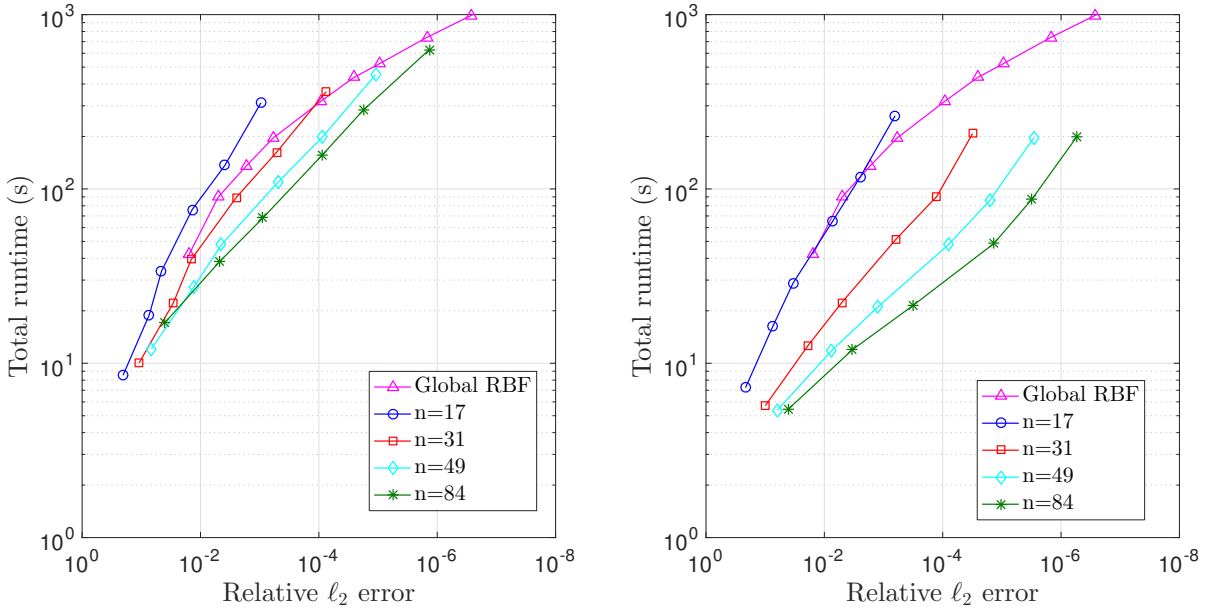
In this article, we presented three new SL methods for transport on the sphere based on interpolation with global RBFs, local RBFs, and RBF-PU. The RBF framework allowed us to obtain either high-order convergence rates for smooth problems using only scattered nodes and Cartesian coordinates. Using scale-free RBFs with the addition of spherical harmonics in the local and RBF-PU methods removes the need to choose a shape parameter and also avoids stagnation errors observed in other applications of these methods. Additionally, the SL framework appears to lend our methods intrinsic stability without the need for hand-tuned artificial hyperviscosity. We summarize the features of our methods below:

- The global RBF methods with smooth kernels appear to have the best cost-accuracy profile for problems with smooth solutions, but these methods scale poorly for problems with non-smooth solutions.
- The RBF-PU method gives smaller errors, lower dispersion, and better conservation properties than the local RBF method for a comparable number of degrees of freedom, regardless of the smoothness of the solutions.
- The local RBF method is easier to implement and may be more amenable to highly efficient implementations on SIMD architectures.

Our algorithms were designed for transport in an incompressible velocity field. However, compressible fields arise naturally in many biological and geological problems. We are currently working on extending our method to handle this case. In addition, there is a need for quasi-monotone (non-oscillatory) local RBF methods to ameliorate the numerical dispersion seen therein. We plan to address this issue in future work. Further, the local RBF method can obtain very large speedups even in a serial implementation if used with the overlapped RBF-FD framework [39]; we plan to explore this in future work as well. Finally, the RBF framework allows for a straightforward generalization of our methods to more general manifolds than the sphere [51–53], providing mathematicians with powerful tools to solve transport problems on more general manifolds.



(a) Deformational flow: Cosine Bells



(b) Deformational flow: Gaussian Bells

Figure 13: Computational cost (runtime) verses accuracy for the deformational flow test case using (a) Cosine Bells and (b) Gaussian bells. The left column shows the results for the local RBF method, while the right is for the RBF-PU method. The global RBF results are repeated in all plots for comparison purposes. The runtime is for the total simulation and does not include any pre-computational costs.

Acknowledgements

VS acknowledges support for this project under NSF DMS-1160432 and NSF DMS-1521748. GBW acknowledges funding support for this project under grants NSF-DMS 1160379 and NSF-ACI 1440638.

References

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

[2] B. Fornberg, C. Piret, On choosing a radial basis function and a shape parameter when solving a convective pde on a sphere, J. Comput. Phys. 227 (5) (2008) 2758–2780.

[3] N. Flyer, G. B. Wright, A radial basis function method for the shallow water equations on a sphere, Proc. Roy. Soc. A 465 (2009) 1949–1976.

[4] Q. T. L. Gia, Approximation of parabolic PDEs on spheres using spherical basis functions, Adv. Comput. Math. 22 (2005) 377–397.

[5] G. B. Wright, N. Flyer, D. A. Yuen, A hybrid radial basis function - pseudospectral method for thermal convection in a 3D spherical shell, Geophysics, Geochemistry, Geosystems 11 (7) (2010) Q07003.

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

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

[8] E. Bollig, N. Flyer, G. Erlebacher, Solution to PDEs using radial basis function finite-differences (RBF-FD) on multiple GPUs, J. Comput. Phys. 231 (2012) 7133–7151.

[9] M. Tillenius, E. Larsson, E. Lehto, N. Flyer, A scalable RBF-FD method for atmospheric flow, J. Comput. Phys. 298 (2015) 406–422.

[10] N. Flyer, G. B. Wright, B. Fornberg, Radial Basis Function-generated Finite Differences: A Mesh-free Method for Computational Geosciences, in: W. Freeden, M. Z. Nashed, T. Sonar (Eds.), Handbook of Geomathematics, 2nd Edition, Springer-Verlag, Berlin, 2014, pp. 1–30.

[11] K. A. Aiton, A Radial Basis Function Partition of Unity Method for Transport on the Sphere, Master's thesis, Boise State University, USA (2014).

[12] A. Safdari-Vaighani, A. Heryudono, E. Larsson, A radial basis function partition of unity collocation method for convection–diffusion equations arising in financial applications, J. Sci. Comput. 64 (2) (2015) 341–367.

[13] N. Flyer, G. A. Barnett, L. J. Wicker, Enhancing finite differences with radial basis functions: Experiments on the Navier-Stokes equations, J. Comput. Phys. 316 (2016) 39–62. doi:10.1016/j.jcp.2016.02.078.

[14] D. Xiu, G. E. Karniadakis, A semi-Lagrangian high-order method for Navier–Stokes equations, J. Comput. Phys. 172 (2) (2001) 658–684.

[15] J. Behrens, A. Iske, Grid-free adaptive semi-Lagrangian advection using radial basis functions, Comput. Math. Appl. 43 (3) (2002) 319–327.

[16] M. Falcone, R. Ferretti, Semi-Lagrangian Approximation Schemes for Linear and Hamilton-Jacobi Equations, SIAM, Philadelphia, PA, 2013. doi:10.1137/1.9781611973051.

[17] A. Staniforth, J. Côté, Semi-Lagrangian integration schemes for atmospheric models—a review, Monthly Weather Review 119 (9) (1991) 2206–2223.

[18] P. J. Rasch, D. L. Williamson, On shape-preserving interpolation and semi-Lagrangian transport, SIAM J. Sci. and Stat. Comput. 11 (4) (1990) 656–687.

[19] A. T. Layton, W. F. Spotz, A semi-Lagrangian double fourier method for the shallow water equations on the sphere, J. Comput. Phys. 189 (1) (2003) 180–196. doi:http://dx.doi.org/10.1016/S0021-9991(03)00207-9.

[20] M. A. Tolstykh, Vorticity-divergence semi-Lagrangian shallow-water model of the sphere based on compact finite differences, J. Comput. Phys. 179 (1) (2002) 180–200. doi:10.1006/jcph.2002.7050.

[21] R. Nair, J. Côté, A. Staniforth, Cascade interpolation for semi-Lagrangian advection over the sphere, Q. J. Roy. Meteor. Soc. 125 (556) (1999) 1445–1468.

[22] P. H. Lauritzen, R. D. Nair, P. A. Ullrich, A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid, J. Comput. Phys. 229 (5) (2010) 1401–1424. doi:10.1016/j.jcp.2009.10.036.

[23] M. F. Carfora, Semi-Lagrangian advection on a spherical geodesic grid, Int. J. Numer. Method. Fluid. 55 (2) (2007) 127–142. doi:10.1002/fld.1445.

[24] N. Flyer, B. Fornberg, V. Bayona, G. A. Barnett, On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy, J. Comput. Phys. 321 (2016) 21–38. doi:10.1016/j.jcp.2016.05.026.

[25] D. L. Williamson, J. B. Drake, J. J. Hack, R. Jakob, P. N. Swarztrauber, A standard test set for numerical approximations to the shallow water equations in spherical geometry, J. Comput. Phys. 102 (1992) 211–224.

[26] R. D. Nair, P. H. Lauritzen, A class of deformational flow test cases for linear transport problems on the sphere, J. Comput. Phys. 229 (23) (2010) 8868–8887.

[27] D. Y. LE ROUX, C. A. LIN, A. STANIFORTH, An accurate interpolating scheme for semi-Lagrangian advection on an unstructured mesh for ocean modelling, Tellus A 49 (1) (1997) 119–138.

[28] B. Fornberg, N. Flyer, A Primer on Radial Basis Functions with Applications to the Geosciences, SIAM, Philadelphia, 2014.

[29] G. E. Fasshauer, Meshfree Approximation Methods with MATLAB, Interdisciplinary Mathematical Sciences - Vol. 6, World Scientific Publishers, Singapore, 2007.

[30] S. Hubbert, T. M. Morton, L_p-error estimates for radial basis function interpolation on the sphere, J. Approx. Theory 129 (1) (2004) 58–77.

[31] F. J. Narcowich, X. Sun, J. D. Ward, H. Wendland, Direct and inverse Sobolev error estimates for scattered data interpolation via spherical basis functions, Found. Comput. Math. 7 (3) (2007) 369–390.

[32] K. Jetter, J. Stöckler, J. D. Ward, Error estimates for scattered data interpolation on spheres, Math. Comput. 68 (226) (1999) 733–747.

[33] B. Fornberg, C. Piret, A stable algorithm for flat radial basis functions on a sphere, SIAM J. Sci. Comput. 30 (2007) 60–80.

[34] Q. T. L. Gia, I. H. Sloan, H. Wendland, Multiscale analysis in Sobolev spaces on the sphere, SIAM J. Numer. Anal. 48 (6) (2010) 2065–2090. doi:10.1137/090774550.

[35] G. B. Wright, B. Fornberg, Scattered node compact finite difference-type formulas generated from radial basis functions, J. Comput. Phys. 212 (1) (2006) 99–123. doi:10.1016/j.jcp.2005.05.030.

[36] H. Wendland, Fast evaluation of radial basis functions: Methods based on partition of unity, in: Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt University Press, 2002, pp. 473–483.

[37] R. Cavoretto, A. DeRossi, Fast and accurate interpolation of large scattered data sets on the sphere, J. Comput. Appl. Math. 234 (2010) 1505–1521.

[38] R. Cavoretto, A. De Rossi, E. Perracchione, Partition of unity interpolation on multivariate convex domains, Int. J. Model. Simul. Sci. Comp. 06 (04) (2015) 1550034.

[39] V. Shankar, The overlapped radial basis function-finite difference (rbf-fd) method: A generalization of rbf-fd, Journal of Computational Physics 342 (2017) 211 – 228. doi:https://doi.org/10.1016/j.jcp.2017.04.037.
URL http://www.sciencedirect.com/science/article/pii/S0021999117303169

[40] D. P. Hardin, E. B. Saff, Discretizing manifolds via minimum energy points, Notices Amer. Math. Soc. 51 (2004) 1186–1194.

[41] R. S. Womersley, I. H. Sloan, How good can polynomial interpolation on the sphere be?, Adv. Comput. Math. 23 (2001) 195–226.

[42] J. R. Baumgardner, P. O. Frederickson, Icosahedral discretization of the two-sphere, SIAM J. Numer. Analy. 22 (6) (1985) 1107–1115. doi:10.1137/0722066.

[43] G. B. Wright, SpherePts, https://github.com/gradywright/spherepts/ (2016).

[44] P. A. Bosler, J. Kent, R. Krasny, C. Jablonowski, A Lagrangian particle method with remeshing for tracer transport on the sphere, J. Comput. Phys. 340 (2017) 639–654.

[45] E. Fehlberg, Classical fifth-, sixth-, seventh-, and eighth-order Runge-Kutta formulas with stepsize control, Tech. Rep. TR R-287, NASA (1968).

[46] T. Michaels, Equidistributed icosahedral configurations on the sphere, Comput. Math. Appl.In press. doi:http://dx.doi.org/10.1016/j.camwa.2017.04.007.

[47] L. L. Takacs, A two-step scheme for the advection equation with minimized dissipation and dispersion errors, Mon. Wea. Rev. 113 (6) (1985) 1050–1065.

[48] M. Restelli, L. Bonaventura, R. Sacco, A semi-Lagrangian discontinuous Galerkin method for scalar advection by incompressible flows, J. Comput. Phys. 216 (1) (2006) 195 – 215. doi:10.1016/j.jcp.2005.11.030.

[49] E. Fuselier, T. Hangelbroek, F. Narcowich, J. Ward, G. Wright, Kernel based quadrature on spheres and other homogeneous spaces, Numer. Math. Online (2014) 1–36. doi:10.1007/s00211-013-0581-1.

[50] R. D. Nair, P. H. Lauritzen, A class of deformational flow test cases for linear transport problems on the sphere, J. Comput. Phys. 229 (2010) 8868–8887.

[51] E. Fuselier, G. Wright, Order-preserving derivative approximation with periodic radial basis functions, Adv. Comput. Math. (2014) In Pressdoi:10.1007/s10444-014-9348-1.

[52] V. Shankar, G. B. Wright, R. M. Kirby, A. L. Fogelson, A radial basis function (RBF)-finite difference (FD) method for diffusion and reaction–diffusion equations on surfaces, J. Sci. Comput. 63 (3) (2014) 745–768.

[53] E. Lehto, V. Shankar, G. B. Wright, A radial basis function (RBF) compact finite difference (FD) scheme for reaction-diffusion equations on surfaces, SIAM J. Sci. Comput. (2017)To appear.