RBF-LOI: Augmenting Radial Basis Functions (RBFs) with Least Orthogonal Interpolation (LOI) for solving PDEs on surfaces

Varun Shankar [a,*], Akil Narayan [b], Robert M. Kirby [c]

[a] *Department of Mathematics, University of Utah, UT, USA*
[b] *Department of Mathematics, and Scientific Computing and Imaging Institute, University of Utah, UT, USA*
[c] *Scientific Computing and Imaging Institute, University of Utah, UT, USA*

ARTICLE INFO

ABSTRACT

We present a new method for the solution of PDEs on manifolds $\mathbb{M} \subset \mathbb{R}^d$ of co-dimension one using stable scale-free radial basis function (RBF) interpolation. Our method involves augmenting polyharmonic spline (PHS) RBFs with polynomials to generate RBF-finite difference (RBF-FD) formulas. These polynomial basis elements are obtained using the recently-developed *least orthogonal interpolation* technique (LOI) on each RBF-FD stencil to obtain *local* restrictions of polynomials in $\mathbb{R}^3$ to stencils on $\mathbb{M}$. The resulting RBF-LOI method uses Cartesian coordinates, does not require any intrinsic coordinate systems or projections of points onto tangent planes, and our tests illustrate robustness to stagnation errors. We show that our method produces high orders of convergence for PDEs on the sphere and torus, and present some applications to reaction–diffusion PDEs motivated by biology.

1. Introduction

Radial Basis Functions (RBFs) are a popular tool for scattered data interpolation and approximation. Much like polynomial approximation methods, RBFs can be used to generate numerical methods for the solution of partial differential equations (PDEs). However, unlike polynomial-based collocation methods, RBF collocation methods are very easily applied to solving PDEs on irregular domains using scattered node layouts [5,9,41]. 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. This ability has been leveraged to obtain four important classes of methods for solving PDEs on manifolds: global RBF methods [17,18,24,29], RBF-Finite Difference (RBF-FD) methods [16,20,30,38], RBF-Partition of Unity (RBF-PU) methods [1], and implicit/Hermite RBF-FD methods [25]. We will focus on RBF-FD methods for $\mathbb{M} \subset \mathbb{R}^3$ for the remainder of this article.

Historically, the primary drawback of RBF methods has been ill-conditioning in the RBF interpolation matrix [11,39]. This ill-conditioning manifests in algorithmic implementations as a *stagnation* in errors and resulting convergence rates as the number of nodes is increased. In Euclidean domains, this is easily remedied by a change of basis when using Gaussian RBFs [12,19,21,22], or enforcing reproduction of moderate to high-degree polynomials when using polyharmonic spline (PHS) RBFs [3,4,14,15]. Unfortunately, none of these methods to offset ill-conditioning appear to apply straightforwardly when the nodes lie on a submanifold $\mathbb{M} \subset \mathbb{R}^d$. Such manifolds appear to require specialized approaches. For instance, Reeger

---

Notation used throughout this article

$\boldsymbol{x}$	Point in $\mathbb{R}^d$
X	Collection of N points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ in $\mathbb{R}^d$
n	Local stencil size
P_k	The collection of the n nearest neighbors in X of $\boldsymbol{x}_k \in X$
$\mathcal{I}_j^k$	The index in X of the jth point in the set P_k, with $\mathcal{I}_1^k \equiv k$
$\mathcal{G}^x$	x-component of the surface gradient
ρ_k	Width of x_k-centered stencil P_k
δ	Overlap parameter
m	Polyharmonic spline degree
R_k	Subset of P_k dictated by overlap parameter δ
V_P	Least orthogonal interpolant polynomial subspace associated to point set P
M	Number of polynomial augmentation terms used in RBF-FD procedure
h_j^k	Ordered orthonormal basis elements for V_{P_k}
τ	LOI tolerance parameter
G_k^x	RBF-FD weights for the operator $\mathcal{G}^x$ on stencil k

---

and Fornberg [31–33] are able to overcome these issues on manifolds by using a triangle mesh, projecting the RBF-FD stencil nodes to the tangent plane, and computing the RBF-FD weights there. However, this is a different philosophy from that of using Cartesian coordinates and embedding space distances employed by previous RBF-FD methods on manifolds. We will not pursue this tangent plane approach further in this article. This choice of working purely on the node set introduces a specific stability issue, which is dealt with in Section 4.2. Yet another way to overcome ill-conditioning with RBF approximations is to view the RBF interpolant as a complex-valued function, and use contour integration [23] or vector-valued rational approximations [42] in the complex plane to avoid pole singularities. While these alternative techniques may be more applicable to manifolds than the change-of-basis techniques, we leave such an exploration for future work.

In recent work, however, it was shown that augmenting PHS RBFs with spherical harmonics ameliorated the ill-conditioning issue for interpolation on the sphere $\mathbb{S}^2$ [37]. Recalling that spherical harmonics are merely (orthonormal) restrictions of polynomials in $\mathbb{R}^3$ to $\mathbb{S}^2$, this opens up an alternative approach to overcoming ill-conditioning on manifolds: for an RBF-FD stencil on any manifold $\mathbb{M} \subset \mathbb{R}^3$, compute the (possibly orthonormal) restriction of a polynomial in $\mathbb{R}^3$ to that stencil, and use this polynomial in conjunction with PHS RBFs to enforce polynomial reproduction on that stencil. This is the approach we employ in this article. Of course, this raises the question of how to generate such a restriction. To generate these polynomial subspaces, we turn to Least Orthogonal Interpolation (LOI). LOI is a procedure for generating a minimum-degree polynomial subspace that achieves unisolvency of the interpolation problem associated to a point set in $\mathbb{R}^d$; the point set may have arbitrary finite size and arbitrary geometric configuration [28]. The LOI procedure requires as input a finite point set and a probability measure on $\mathbb{R}^d$, and outputs a polynomial subspace whose dimension equals the cardinality of the input point set. In addition, LOI also outputs an orthonormal basis for that polynomial subspace. We note that the LOI procedure is itself a generalization of de Boor and Ron's *least interpolation* [7]. Interestingly, RBFs with shape parameters have long been known to recover this least interpolant in a limiting case [34], further highlighting the connection between LOI and RBF methods.

We have observed in testing that using LOI polynomial approximations alone in collocation methods for PDEs on scattered nodes produces unstable or unsuitable results. However, this basis has advantageous use in enforcing polynomial reproduction in a PHS-based RBF-FD method, which is the approach we take in this paper. The resulting method, which we call RBF-LOI, retains the strengths of traditional RBF-FD methods on manifolds (Cartesian coordinates and embedding space distance calculations), while overcoming their weaknesses (ill-conditioning and stagnation errors) without the use of higher-precision arithmetic as in [25,38]. The LOI procedure does increase the cost of the RBF-FD method, as does the growth in stencil sizes required to support RBFs augmented with polynomials. The cost increases in assembling differentiation matrices corresponding to these larger stencils can be largely ameliorated by using the overlapped RBF-FD method, a generalization of the RBF-FD method where each local RBF interpolant is used to compute more than one set of RBF-FD weights, thereby drastically decreasing the total number of stencils for a given node set [35]. For this reason, we employ the overlapped RBF-FD method for this article. We will explore the impact of using overlapped RBF-FD on solution time in Section 5.3. We also remark that while the RBF-LOI procedure appears to be stable on a wide variety of manifolds of different genus, its use on more complicated point-cloud surfaces is a subject of future research.

The remainder of this paper is organized as follows. In the next section, we present the first application of the overlapped RBF-FD method on manifolds. Section 3 contains a brief description of the LOI procedure for selecting the polynomial subspaces required by the overlapped RBF-FD method. In Section 4, we discuss the stability of the RBF-LOI procedure, and present our approaches to stabilizing our technique. We then validate our methods on the forced heat equation in Section 5 by measuring errors and convergence rates on the sphere and torus. In Section 6, we present applications of our method to solving nonlinear reaction–diffusion equations on more complicated manifolds; more specifically, we solve the

Cahn–Hilliard, Fitzhugh–Nagumo, and Turing equations on some interesting manifolds. We conclude with a summary of our results and a discussion of future work in Section 7.

2. Overlapped RBF-FD on surfaces

2.1. Description

We first describe the overlapped RBF-FD method, recently developed by the first author [35], and its extension to interpolation on $\mathbb{M} \subset \mathbb{R}^3$. Let $X = \{\boldsymbol{x}_k\}_{k=1}^N$ be a global set of nodes on the manifold $\mathbb{M}$. Define the stencil P_k to be the set of nodes containing nodes $\boldsymbol{x}_{\mathcal{I}_1^k}$ and its $n-1$ nearest neighbors $\{\mathcal{I}_2^k, \dots, \mathcal{I}_n^k\}$; here, $\{\mathcal{I}_1^k, \dots, \mathcal{I}_n^k\}$ are indices that map into the global node set X. We defer discussion of the number of stencils to the end of this section. For the remainder of this discussion, we will focus without loss of generality on the stencil P_1. Assume further that 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 outward normal, and ∇ is the $\mathbb{R}^3$ gradient. We will first discuss approximating the surface gradient, then use this approximation to approximate the surface Laplacian (Laplace–Beltrami operator),

$$\Delta_{\mathbb{M}} = \nabla_{\mathbb{M}} \cdot \nabla_{\mathbb{M}},$$
$$\implies \Delta_{\mathbb{M}} = \mathcal{G}^x\mathcal{G}^x + \mathcal{G}^y\mathcal{G}^y + \mathcal{G}^z\mathcal{G}^z. \tag{2}$$

Before proceeding, we define the *stencil width* ρ_1 as

$$\rho_1 = \max_{j=1,\dots,n} \|\boldsymbol{x}_{\mathcal{I}_1^1} - \boldsymbol{x}_{\mathcal{I}_j^1}\|, \tag{3}$$

where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^3$. Given an *overlap parameter* $\delta \in (0, 1]$, we can now define the *stencil retention distance* r_1 to be

$$r_1 = (1 - \delta)\rho_1. \tag{4}$$

The parameters ρ_1 and δ define a ball $\mathbb{B}_1$ of radius r_1 centered at the node $\boldsymbol{x}_{\mathcal{I}_1^1}$. Let p_1 denote the number of nodes in P_1 that lie in $\mathbb{B}_1$. Now, let R_1 be the set of global indices of the p_1 nodes in the subset $\mathbb{B}_1 \subseteq P_1$:

$$R_1 = \{\mathcal{R}_1^1, \mathcal{R}_2^1, \dots, \mathcal{R}_{p_1}^1\}, \quad 1 \le p_1 \le n. \tag{5}$$

In general, R_1 is some subset of a permutation of the global node indices associated to P_1. The overlapped RBF-FD method involves computing RBF-FD weights for all the nodes whose indices are in R_1, and repeating this process for each stencil P_k. Focusing on the $\mathcal{G}^x$ component of $\nabla_{\mathbb{M}}$, the overlapped RBF-FD weights for all points $\boldsymbol{x} \in \mathbb{B}_1$ with indices in R_1 are computed using the following augmented local RBF interpolant on P_1:

$$s_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^n (g^x)_j^1(\boldsymbol{y})\|\boldsymbol{x} - \boldsymbol{x}_{\mathcal{I}_j^1}\|^m + \sum_{i=1}^M \lambda_i^1(\boldsymbol{y})h_i^1(\boldsymbol{x}), \tag{6}$$

where all superscripts "1" refer to the stencil index, and $\|\boldsymbol{x} - \boldsymbol{x}_{\mathcal{I}_j^1}\|^m$ is the polyharmonic spline (PHS) RBF of degree m (m is odd). A standard RBF-FD procedure would take $h_i^1(\boldsymbol{x})$ as the M monomials corresponding to a carefully chosen d-variate polynomial. The key to our technique is the selection of these polynomial basis functions using the least orthogonal interpolation (LOI) technique [28]. We will defer discussion on the selection of these basis functions to Section 3, and for now consider the h_i^1 functions as arbitrary polynomial functions.

The n overlapped RBF-FD weights $(g^x)_j^1(\boldsymbol{y})$ are written explicitly as functions of the evaluation point $\boldsymbol{y}$; in other words, each evaluation point $\boldsymbol{y}$ in the stencil has a set of n weights associated with it. Our ultimate goal is to compute the weights L_1 for the Laplace–Beltrami operator at all points with indices in the set R_1. To avoid differentiating normals, we will accomplish this using *iterated interpolation* [24,25,38]. This is done in two steps: first compute 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 in P_1); then, combine those RBF-FD weights in such a way that we only compute the weights for all nodes with indices in the set R_1. We will now show how these weights are computed for the operator $\mathcal{G}^x$. We impose the following two (sets of) conditions on the interpolant (6):

$$s_1(\boldsymbol{x}_{\mathcal{I}_j^1}, \boldsymbol{x}_{\mathcal{I}_i^1}) = \left(\mathcal{G}^x\|\boldsymbol{x} - \boldsymbol{x}_{\mathcal{I}_j^1}\|^m\right)\Big|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}_i^1}}, \qquad i=1,\dots,n, j=1,\dots,n, \tag{7a}$$

$$\sum_{j=1}^n (g^x)_j^1(\boldsymbol{x}_{\mathcal{I}_k^1})h_i^1(\boldsymbol{x}_{\mathcal{I}_j^1}) = \left(\mathcal{G}^x h_i^1(\boldsymbol{x})\right)\Big|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}_k^1}}, \quad k=1,\dots,n, i=1,\dots,M. \tag{7b}$$

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 the overlapped RBF-FD weights. If a degree-ℓ polynomial space is employed for $h^1(\boldsymbol{x})$, then $M = \binom{\ell+d}{d}$; for stability, we also require that $M \leq \lfloor \frac{n}{2} \rfloor$ [14,15,35]. The interpolant (6) and the two conditions (7a)–(7b) can be collected into the following block linear system:

$$\begin{bmatrix} A_1 & H_1 \\ H_1^T & O \end{bmatrix} \begin{bmatrix} G_1^x \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} B_{A_1} \\ B_{H_1} \end{bmatrix}, \tag{8}$$

where

$$(A_1)_{ij} = \|\boldsymbol{x}_{\mathcal{I}_i^1} - \boldsymbol{x}_{\mathcal{I}_j^1}\|^m, \qquad i, j = 1, \ldots, n, \tag{9a}$$

$$(H_1)_{ij} = h_j^1(\boldsymbol{x}_{\mathcal{I}_i^1}), \qquad i = 1, \ldots, n, \; j = 1, \ldots, M, \tag{9b}$$

$$(B_{A_1})_{ij} = \mathcal{G}^x \|\boldsymbol{x} - \boldsymbol{x}_{\mathcal{I}_j^1}\|^m \Big|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}_i^1}}, \; i, j = 1, \ldots, n, \tag{9c}$$

$$(B_{H_1})_{ij} = \mathcal{G}^x h_j^1(\boldsymbol{x}) \Big|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}_j^1}}, \qquad i = 1, \ldots, M, \; j = 1, \ldots, n, \tag{9d}$$

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

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 $\boldsymbol{x} \in P_1$. The linear system (8) has a unique solution if the nodes in P_1 are distinct [11,39]. More interestingly, (8) clearly shows that the $n \times M$ matrix of polynomial coefficients Λ_1 is a set of Lagrange multipliers that enforces the polynomial reproduction constraint (7b). The above procedure can be repeated with the operators $\mathcal{G}^y$ and $\mathcal{G}^z$ to obtain the local differentiation matrices G_1^y and G_1^z. Next, define the *truncated* matrix $\tilde{G}_1^x$ as:

$$\left(\tilde{G}_1^x \right)_{ij} = \left(G_1^x \right)_{ij}, i = 1, \ldots, n, j = 1, \ldots, p_1, \tag{10}$$

i.e., the $n \times p_1$ submatrix of G_1^x corresponding to the nodes in the ball $\mathbb{B}_1$. Similarly define the truncated matrices $\tilde{G}_1^y$ and $\tilde{G}_1^z$. Finally, we use iterated interpolation to obtain the differentiation matrix L_1 for the Laplace–Beltrami operator $\Delta_\mathbb{M}$ as:

$$L_1 = \left(\tilde{G}_1^x \right)^T \left(G_1^x \right)^T + \left(\tilde{G}_1^y \right)^T \left(G_1^y \right)^T + \left(\tilde{G}_1^z \right)^T \left(G_1^z \right)^T. \tag{11}$$

This construction using truncated matrices ensures that the $p_1 \times n$ matrix L_1 only contains RBF-FD weights for the nodes in P_1 whose indices are in the set R_1. By construction, the rows of L_1 populate the rows of a global differentiation matrix L, while the *columns* of $\tilde{G}_1^x$ and its counterparts populate the rows of the global differentiation matrices G^x, G^y, and G^z. If the weights for the Laplace–Beltrami operator are not required, it is straightforward to directly compute the truncated matrices $\tilde{G}_1^x$, $\tilde{G}_1^y$, and $\tilde{G}_1^y$ by modifying (7a)–(7b).

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 $P_i, i \neq k$. The entire procedure above must be performed for each stencil; this can be computationally onerous if the number of stencils is comparable to the total number of points N. Denote the total number of stencils by N_δ. For a quasi-uniform node set, $N_\delta = \frac{N}{p}$, where $p = \max\left((1-\delta)^d n, 1\right)$, and d is the dimension (in the above discussion, $d = 3$). If $\delta = 1$, this gives us $N_\delta = N$, recovering the standard RBF-FD method. However, if $\delta < 1$, then $N_\delta << N$, giving a significant speedup over the standard RBF-FD method. For a detailed complexity analysis, see [35].

2.2. Parameter selection

In this section, we describe parameter selection for our method. Given a linear operator $\mathcal{L}$ of order θ and an RBF-FD differentiation rule $(\boldsymbol{x}_j, w_j)_{j=1}^n$, we have the following error estimate for RBF-FD based formulas that reproduce a polynomial of degree ℓ [10]:

$$\left| \mathcal{L} f(\boldsymbol{x}) - \sum_{j=1}^n w_j f(\boldsymbol{x}_j) \right| \leq C(m, \boldsymbol{x}) h^{\ell+1-\theta}, \tag{12}$$

where h is the fill distance of the node set, and m is the degree of the RBF PHS used. While a derivation of such a formula for RBF-FD on manifolds is pending, we nevertheless use this formula to guide our parameter selection. If we require an RBF-FD method with order of accuracy ξ, we set

$$\ell = \xi + \theta - 1. \tag{13}$$

It is important to note that in the context of the LOI procedure (Section 3), the input value of ℓ is merely a "requested" polynomial degree. In practice, the LOI procedure may output a polynomial of degree slightly lower than ℓ. Thus, while the number of polynomial basis functions is related to the input degree ℓ as $M = \binom{\ell+d}{d}$, the LOI procedure (and its accompanying stabilization techniques) may in practice result in a smaller M than requested. Regardless, since this only affects the polynomial reproduction, we select the stencil size as

$$n = 2M + 1 = 2\binom{\ell + d}{d} + 1. \tag{14}$$

The degree m of the PHS RBF can either be fixed [3,15,31–33], or increased with respect to ℓ [6,37]. In Euclidean domains, it appears beneficial to fix m [3,14,15,35]. On the other hand, the traditional scaling law $m = 2\ell + 1$, appears to give the greatest accuracy and stability on manifolds [37]. We have found that $m = 2\ell + 1$ was the most stable choice for all manifolds considered in this article.

Finally, we must also choose the overlap parameter $\delta \in (0, 1]$. In practice, we have observed that setting $\delta \leq 0.2$ typically completely decouples the stencils, resulting in ill-posed subproblems when solving PDEs. However, using $0.2 < \delta \leq 1$ appears to be perfectly stable, and δ can be chosen to be smaller for larger values of n [35]. Given these constraints, we use the following heuristic:

$$\delta = \begin{cases} 0.7 & \text{if } \ell \leq 4 \\ 0.5 & \text{if } 4 < \ell \leq 6 \\ 0.3 & \text{if } \ell > 6 \end{cases}$$

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

3. Least Orthogonal Interpolation

The polynomially-augmented RBF-FD procedure described above requires specification of the polynomial functions portion of the algorithm, i.e., specification of the functions $h_j^k(\mathbf{x})$, $j = 1, \ldots, M$. We define this polynomial basis in this section; for simplicity we omit all notational dependence on the stencil index k in this section.

As mentioned in Section 1, the LOI procedure outputs a polynomial subspace for a given finite input point set. In addition, the procedure also outputs a basis $h_j(\mathbf{x})$ for the polynomial subspace whose elements are orthonormal in a weighted L^2 space on $\mathbb{R}^d$, where the weight is given by the differential of a user-prescribed probability measure. This is the basis we will use to compute augmented RBF-FD weights in (8). We start our discussion by assuming that the probability measure is given and fixed (denoted μ below), and describe towards the end of this section our choice for this measure.

3.1. Notation

With $d \in \mathbb{N}$, a point $\mathbf{x} \in \mathbb{R}^d$ has Cartesian components

$$\mathbf{x} = \left\{ x^{(1)}, \ldots, x^{(d)} \right\}.$$

We use standard multi-index notation: given a multi-index $\alpha \in \mathbb{N}_0^d$, we have

$$\alpha = \left(\alpha^{(1)}, \ldots, \alpha^{(d)} \right) \in \mathbb{N}_0^d, \qquad |\alpha| = \sum_{q=1}^d \alpha^{(q)} \qquad \mathbf{x}^\alpha = \prod_{q=1}^d \left(x^{(q)} \right)^{\alpha^{(q)}}.$$

We use V_{n-1} to denote the space of polynomials of degree $n - 1$ or less in $\mathbb{R}^d$:

$$V_{n-1} = \text{span} \left\{ \mathbf{x}^\alpha \mid \alpha \in \mathbb{N}_0^d, \ |\alpha| \leq n - 1 \right\}, \qquad \dim V_{n-1} = \binom{n - 1 + d}{d}.$$

Let μ be a probability measure on $\mathbb{R}^d$, and let $L_\mu^2(\mathbb{R}^d)$ be the space of real-valued square-integrable functions with respect to the measure μ on $\mathbb{R}^d$. We assume that μ has finite polynomial moments of all orders and is not degenerate with respect to polynomials, i.e.,

$$0 < \int_{\mathbb{R}^d} \mathbf{x}^{2\alpha} \, d\mu(x) < \infty, \qquad \alpha \in \mathbb{N}_0^d \tag{15}$$

Formally, we require only finite moments up to a finite order for the procedure we discuss, and degeneracy is neither a mathematical nor a computational issue. However, the stronger assumptions above are sufficiently general for our presentation, and codifying the allowable relaxation of the above requirements involves unnecessary technical discussions.

Under the above conditions, there is a sequence of polynomials ϕ_α, $\alpha \in \mathbb{N}_0^d$, with $\deg \phi_\alpha = |\alpha|$, satisfying

$$\int_{\mathbb{R}^d} \phi_\alpha(\boldsymbol{x}) \phi_\beta(\boldsymbol{x}) \mathrm{d}\mu(\boldsymbol{x}) = \delta_{\alpha,\beta}, \qquad V_n = \mathrm{span}\left\{\phi_\alpha \mid \alpha \in \mathbb{N}_0^d, \ |\alpha| \leq n\right\}.$$

Note that μ may have compact support, in which case all integrals can be reduced to ones over this compact set. For each n, the space V_n is a finite-dimensional Hilbert space.

Assuming polynomials are complete in L_μ^2, any $f \in L_\mu^2$ has the Fourier-like expansion

$$f(\boldsymbol{x}) = \sum_{\alpha \in \mathbb{N}_0^d} \widetilde{f}_\alpha \phi_\alpha(\boldsymbol{x}) = \sum_{j=0}^{\infty} \sum_{|\alpha|=j} \widetilde{f}_\alpha \phi_\alpha(\boldsymbol{x}) =: \sum_{j=0}^{\infty} f_j(\boldsymbol{x}), \qquad \widetilde{f}_\alpha = \int_{\mathbb{R}^d} f(\boldsymbol{x}) \phi_\alpha(\boldsymbol{x}) \mathrm{d}\mu(\boldsymbol{x}),$$

where we have defined $f_j \in V_j$ in terms of the coefficients $\widetilde{f}_\alpha$. We define the operation $(\cdot)_\downarrow$ as follows:

$$f_\downarrow := f_r(\boldsymbol{x}), \qquad r = \min\left\{j \in \mathbb{N}_0 \mid f_j \neq 0\right\}.$$

Note that this operation depends on μ.

3.2. The Least Orthogonal Interpolant

Let $P = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$ be a point set of size $n \in \mathbb{N}$. With μ fixed, the least orthogonal interpolation procedure provides a polynomial subspace of dimension n associated to P.

The Riesz representor of the point-evaluation map $v \mapsto \delta_{\boldsymbol{x}_j}(v) = v(\boldsymbol{x}_j)$ in the finite-dimensional space V_{n-1} has the form

$$v_j(\boldsymbol{x}) = \sum_{|\alpha| \leq n-1} \phi_\alpha(\boldsymbol{x}_j) \phi_\alpha(\boldsymbol{x}). \tag{16}$$

With v_j, $j = 1, \ldots, n$, defined above in terms of the nodes in P, we can define the following space of polynomials:

$$V_P := \mathrm{span}\left\{v_\downarrow \mid v \in \mathrm{span}\{v_1, \ldots, v_n\}\right\}. \tag{17}$$

The main result from [28] is that the space V_P has dimension n and the interpolation problem on P in the space V_P is unisolvent. Therefore, we can always identify a unique polynomial in V_P given data on P; this polynomial is the least orthogonal interpolant, and V_P is the least orthogonal interpolant (polynomial) space associated to P. Because V_P has dimension n, there is an L_μ^2-orthonormal basis, $h_1(\cdot), \ldots, h_n(\cdot)$, for V_P. This basis can be computationally generated using linear algebra, and these are the basis elements that we use in the RBF-FD procedure (8) and (9). This algorithmic construction depends on detecting rank-deficiency of certain submatrices of a Vandermonde-like matrix. Like all numerical linear algebraic methods to detect rank, this in turn depends on user specification of a tunable tolerance parameter denoted τ. This tolerance parameter is related to numerical unisolvency of the interpolation problem, and we discuss it in more detail in section 4.1.

The LOI definition above is abstract but, given a point set P and data on P, the computation of the interpolant (and the basis h_j) involves only standard tools from numerical linear algebra, namely LU and QR factorizations [28]. In particular, the computational complexity of the procedure is comparable to that for a standard interpolation problem of size n.

3.3. Differentiation of the interpolant

Condition (9d) in the formation of local RBF-FD weights shows that we must have the ability to differentiate the polynomial basis h_j. The least orthogonal interpolant associated to the points set P of size n has the form

$$p(\boldsymbol{x}) = \sum_{j=1}^{n} c_j h_j(\boldsymbol{x}) \in V_P, \qquad \int_{\mathbb{R}^d} h_j(\boldsymbol{x}) h_\ell(\boldsymbol{x}) \mathrm{d}\mu(\boldsymbol{x}) = \delta_{j,\ell}$$

The orthonormal basis h_j is output from the LOI procedure. (The coefficients c_j may also be easily generated if data on P is given.) The basis elements are polynomials, each of degree $\deg h_j$, and have an expansion in terms of the original orthonormal basis ϕ_α:

$$h_j(\boldsymbol{x}) = \sum_{|\alpha|=\deg h_j} d_{j,\alpha} \phi_\alpha(\boldsymbol{x}),$$

for some constants $d_{j,\alpha}$ that are output from the LOI procedure. (Note above that we only need take α satisfying $|\alpha| = \deg h_j$, not $|\alpha| \leq \deg h_j$.) For the purposes of solving PDEs, we are particularly interested in differentiating the interpolant $p(x)$, say with respect to coordinate ℓ. This is given by

$$\frac{\partial p}{\partial x^{(\ell)}} = \sum_{j=1}^{n} c_j \sum_{|\alpha|=\deg h_j} d_{j,\alpha} \frac{\partial \phi_\alpha}{\partial x^{(\ell)}}.$$

Therefore in order to compute derivatives, we need only the ability to construct the interpolant and to differentiate the original basis ϕ_α. The Cartesian derivatives of the basis ϕ_α can then be combined appropriately to give the surface gradient $\nabla_\mathbb{M}\phi_\alpha$.

3.4. The measure μ

The LOI procedure is a well-defined map from a point configuration to the sought basis h_j (and its derivatives). However, we have yet to make a specification for the probability measure μ. In principle any measure satisfying (15) will suffice, but in our quest for stable methods, it seems more prudent to choose μ so that the input orthonormal basis ϕ_α is well-behaved.

For simplicity, we choose μ to be the tensor-product Chebyshev measure over the smallest bounding box for the nodal set P. With this, we can generate the basis ϕ_α as

$$\phi_\alpha(\boldsymbol{x}) = \prod_{j=1}^{d} T_{\alpha^{(j)}}\left(x^{(j)}\right),$$

where $\{T_q\}_{q=0}^{\infty}$ are the univariate orthonormal Chebyshev polynomials. In this way, the basis ϕ_α along with its partial derivatives are easily computed.

In order to apply all of the above to the RBF problem, for each stencil (nodal set) P_k we compute LOI basis functions h_j^k for use on that stencil, using the prescription of μ above.

4. Eigenvalue stability

Eigenvalue stability is essential for stable time integration of PDEs. In this context, a discrete version of a diffusive (elliptic) differential operator can be regarded as "stable" if its spectrum contains no eigenvalue with positive real parts. In [38], this stability was achieved by performing a nonlinear optimization for the shape parameter on each stencil, constrained so that the RBF interpolation matrices on each stencil should have approximately the same condition number. In [25], this stability was achieved by encouraging diagonal dominance in RBF-HFD differentiation matrices using a stencil selection algorithm.

In contrast, stability in the RBF-LOI procedure is primarily achieved by picking a single tolerance parameter τ that is globally defined over the whole mesh. We also utilize an additional empirical s tabilization procedure based on avoiding "axis alignment", that is alignment of a stencil configuration with the global Cartesian coordinate system. This last correction is a pathology of our Cartesian representation of points on a manifold, and is needed only on stencils for which one or more points geometrically aligns with a Cartesian axis. We have observed that this stabilization is needed on very few stencils in all our tests.

We will now describe both techniques for stabilization. Throughout this section we use the same notation as in Section 3: P is a generic RBF-FD stencil containing the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. The principles discussed here are applied for each local RBF-FD stencil.

4.1. The LOI tolerance parameter

The tolerance parameter τ that we use is embedded in the LOI construction procedure. This tolerance parameter is a way of tuning selection of the polynomial degree so that the polynomial interpolation on P is numerically stable.

The mathematical LOI procedure described in Section 3 can be implemented as a sequence of QR decompositions [28], and the complexity of the entire procedure is asymptotically the same as a standard, square, interpolation problem of the same size, *i.e.*, $O(n^3)$. Recall that, given a finite point set P, the LOI polynomial space V_P has dimension $n = |P|$, and the elements $\{h_j\}_{j=1}^{n}$ are an orthonormal basis for V_P.

The algorithmic implementation of the LOI procedure builds the functions $h_1, \ldots, h_k$ sequentially by identifying them with points in P, *i.e.*, $h_1, \ldots, h_k$ are built and are each identified with a sequence $Y_k := \{\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}\} \subset P$, where $i_1, \ldots, i_k \in \{1, \ldots, n\}$. After $h_1, \ldots, h_k$ have been built, then h_{k+1} is identified and constructed with an associated point $\boldsymbol{x}_{i_{k+1}} \in P \backslash Y_k$. The identification and construction of h_{k+1} is based on the residual of a projection. Let $k < n$, and define $r := \deg h_k$; we must have that $r \le n - 1$. We require truncations of the summation in (16):

$$v_{j,r}(\boldsymbol{x}) = \sum_{|\alpha| \le r} \phi_\alpha(\boldsymbol{x}_j) \phi_\alpha(\boldsymbol{x}), \qquad r \le n - 1.$$

Now define $\Pi_r^\perp$ as the L_μ^2-orthogonal projector onto orthogonal complement of V_r, and let I_k denote the LOI interpolation operator associated with Y_k with the basis $h_1, \ldots, h_k$; i.e.,

$$I_k f(\pmb{x}) = \sum_{j=1}^{k} c_j h_j(\pmb{x}), \qquad I_k f(\pmb{x}_j) = f(\pmb{x}_j), \quad j = 1, \ldots, k.$$

We then define the degree r "residual" as

$$R = \max_{j \in \{1,\ldots,n\} \setminus \{i_1,\ldots,i_k\}} \left\| \Pi_r^{\perp} \left(v_{j,r} - I_k v_{j,r} \right) \right\|_{L_\mu^2}. \tag{18}$$

Now let $\tau > 0$ be a tolerance parameter. This tolerance parameter is a threshold for the allowable residual value R. If $R \geq \tau$, then we choose

$$i_{k+1} = arg\,max_{j \in \{1,\ldots,n\} \setminus \{i_1,\ldots,i_k\}} \left\| \Pi_r^{\perp} \left(v_{j,r} - I_k v_{j,r} \right) \right\|_{L_\mu^2},$$

and h_{k+1} is chosen as a normalized version of $\Pi_r^{\perp}(v_{i_{k+1},r} - I_k v_{i_{k+1},r})$. Otherwise, if $R < \tau$, we set $r \leftarrow r + 1$, recompute R from (18), and repeat the comparison of R to τ. When $\tau = 0$, only a pathological prescription of μ allows more than one increment of r for each k.

The tolerance parameter τ can now be understood in terms of R. The quantity R measures the ability of the point set P to resolve (with respect to the measure μ) a certain subspace of polynomials of degree r. The comparison of R with τ then enforces a desired threshold of resolvability for subspaces included in the LOI procedure. When this threshold is not met, the LOI interpolation operator will be (relatively) ill-conditioned on this subspace of polynomials. Therefore, instead of including this subspace, we simply increment the degree (generate a new subspace) in hopes of achieving a more stable polynomial approximation. This simple heuristic allows us, for a fixed τ, to achieve stable RBF-LOI approximations for general test cases without any optimization.

4.2. Axis misalignment

We empirically observe that some local stencils produce unstable results when the point set P has a very special configuration in space. This instability is not directly caused by our procedures, but instead by our choice of the alignment of a Cartesian coordinate system in d-dimensional space; it is plausible that this instability may not occur, say, for approximations on the tangent plane. We first describe the source of the instability, and then describe our simple computational strategy to circumvent the issue. To keep notational jargon at a minimum in this section, with $d = 3$ we use the notation $(x^{(1)}, x^{(2)}, x^{(3)}) = (x, y, z)$.

The stencil P contains a spatial configuration of points, and the LOI procedure outputs a basis $h_1, \ldots, h_M$ from these points. The instability we observe stems from situations where a special arrangement of points P results entries of the matrix defined in (9d) satisfying $\mathcal{G}^x h_j \equiv 0$. This "zero column" of the matrix B_{H_1} causes numerical instabilities when the corresponding global (sparse) differentiation matrix is used for the discretization of PDEs.

We give a brief account of why this "zero column" occurs: The functions h_j are arranged in order of increasing polynomial degree. E.g., in $d = 3$, h_1 is the constant polynomial, and h_2, h_3, and h_4 are all linear polynomials, except in pathological arrangements of P or for pathological measures μ. For simplicity of discussion, we assume in this section that $d = 3$ and that h_2, h_3, and h_4 are all linear polynomials.

Suppose that the outward pointing normal vector $\pmb{n}$ at the stencil center equals $(n_1, n_2, n_3)^T$. This implies that the x-component $\mathcal{G}^x$ of the surface gradient operator in (1) is given by

$$\begin{pmatrix} \mathcal{G}^x \\ \mathcal{G}^y \\ \mathcal{G}^z \end{pmatrix} = \left(\pmb{I} - \pmb{n}\pmb{n}^T \right) \nabla$$

A linear polynomial, say h_2, has the expansion

$$h_2(\pmb{x}) = \alpha x + \beta y + \gamma z + d,$$

where α, β, γ, and d are all constants. Then we can compute

$$\begin{pmatrix} \mathcal{G}^x \\ \mathcal{G}^y \\ \mathcal{G}^z \end{pmatrix} h_2 = \left(\pmb{I} - \pmb{n}\pmb{n}^T \right) \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}.$$

We can see then that one component of this vanishes when any row of $\left(\pmb{I} - \pmb{n}\pmb{n}^T \right)$ is orthogonal to $(\alpha, \beta, \gamma)^T$. While this situation happens rarely, it is not difficult to construct situations when such a condition is triggered. Indeed, by constructing a stencil arranged with a normal vector $\pmb{n} = (1, 0, 0)^T$, then certain configurations of the stencil P cause $(\alpha, \beta, \gamma)^T = (\alpha, 0, 0)^T$, which then results in $\mathcal{G}^x h_2 \equiv 0$.

We observe in practice that stencils centered on points that are "axis aligned" with the Cartesian grid generate an LOI basis that causes one surface gradient component for linear polynomials to vanish. Our simple fix to circumvent this alignment issue for those stencils is as follows:

1. Check if any column of the LOI $\mathbb{R}^3$ gradient-component matrices on the stencil P_k (other than the first) contains only zeros. Mark the columns.
2. Eliminate the basis functions $h_j^k(\boldsymbol{x})$ (and their derivatives) corresponding to these marked columns from the matrices H_k and B_{H_k}.

5. Results

We have completed a full description of the RBF-LOI procedure and in this section we test the convergence rates of this method on the sphere and torus, where explicit expressions for surface differential operators are known. We then present timing results for our methods as a function of error. In all cases, the relative errors are measured in the embedding space and no quadrature is used. This may affect the constants involved, but should not affect the convergence rates.

5.1. Advection on the sphere $\mathbb{S}^2$

In this test, we solve the surface advection equation on the sphere; in conservative form, this is given by:

$$\frac{\partial c}{\partial t} + \nabla_{\mathbb{M}} \cdot (\boldsymbol{u} c) = 0, \tag{19}$$

where $c(\boldsymbol{x}, t)$ is some scalar field being advected on the surface $\mathbb{M}$ by the velocity field $\boldsymbol{u}(\boldsymbol{x}, t)$. Of course, if $\nabla_{\mathbb{M}} \cdot \boldsymbol{u} = 0$, this equation can be simplified to $\frac{\partial c}{\partial t} + \boldsymbol{u} \cdot \nabla_{\mathbb{M}} c = 0$. However, we opt to discretize (19) directly as this appears to produce lower errors than the alternative approach. Unfortunately, the RBF-FD differentiation matrices corresponding to the components of the surface divergence operator $\nabla_{\mathbb{M}}$ typically contain spurious eigenvalues in their spectra, often with (small) positive real parts. This can cause instabilities especially when an explicit time discretization is used. The remedy for this issue is to add a small amount of artificial hyperviscosity for stabilization [16,20]. This transforms (19) to

$$\frac{\partial c}{\partial t} + \nabla_{\mathbb{M}} \cdot (\boldsymbol{u} c) = \gamma \Delta_{\mathbb{M}}^k, \tag{20}$$

where $\gamma \in \mathbb{R}$ and $k \in \mathbb{N}$ must be tuned. Typically, γ is a small number that goes to zero as $N \to \infty$, and k is gently increased as the order of the method is increased [16]. For simplicity, we compute the discrete surface hyperviscosity operator by simply computing L^k, where L is the discrete surface Laplacian.

While formulas for γ can be found in the literature for Euclidean domains in the context of PHS-based RBF-FD [3,14], there are no such formulas for the PHS-based RBF-FD on the sphere (to the best of our knowledge). In this work, we use the following formulas for γ and k:

$$\gamma = (-1)^{1+k} 2^{2-2k} \left(\sqrt{N} \right)^{2-2k} \Lambda_{\max} \|\boldsymbol{u}\|_{\max}, \tag{21}$$

$$k = \lfloor \ln n \rfloor, \tag{22}$$

where $\Lambda_{\max}$ is the *real part* of the eigenvalue with largest real part of the sparse differentiation matrices G^x, G^y, and G^z. $\|\boldsymbol{u}\|_{\max}$ is shorthand for the maximum of the pointwise ℓ_2 norms of the velocities evaluated on the node set. A derivation of this formula is beyond the scope of this paper, but will be detailed in future work. For this article, we always have $\|\boldsymbol{u}\|_{\max} = 1$. $\Lambda_{\max}$ is estimated by Matlab calls to eigs(.,1,'LR'), with a very loose tolerance of $8e-2$; this estimation is a preprocessing step for a given node set and stencil size. Our chosen test problem is the deformational flow test case from [27]. The components of the velocity field (in spherical coordinates) are

$$u(\phi_1, \phi_2, t) = \frac{10}{T} \cos\left(\frac{\pi t}{T}\right) \sin^2\left(\phi_1 - \frac{2\pi t}{T}\right) \sin(2\phi_2) + \frac{2\pi}{T} \cos(\phi_2), \tag{23}$$

$$v(\phi_1, \phi_2, t) = \frac{10}{T} \cos\left(\frac{\pi t}{T}\right) \sin\left(2\phi_1 - \frac{2\pi t}{T}\right) \cos(\phi_2), \tag{24}$$

where $-\pi \leq \phi_1 \leq \pi$, $-\pi/2 \leq \phi_2 \leq \pi/2$, and $T = 5$. The flow field deforms the initial condition up to time $t = 2.5$ and then reverses to return the solution to its initial position at $t = 5$, which serves as the final time for the simulation. A simple change of basis is used to convert the velocity field into Cartesian coordinates. To test the convergence behavior of RBF-LOI under refinement, we use a smooth initial condition in the form of two Gaussian bells, given by

$$c(\boldsymbol{x}, 0) = 0.95 \left(e^{-5\|\boldsymbol{x}-\mathbf{p}_1\|_2^2} + e^{-5\|\boldsymbol{x}-\mathbf{p}_2\|_2^2} \right), \tag{25}$$

where $\mathbf{p}_1 = \left(\sqrt{3}/2, 1/2, 0 \right)$ and $\mathbf{p}_2 = \left(\sqrt{3}/2, -1/2, 0 \right)$. Following [1], we use a time-step of $\Delta t = \frac{5}{2400}$ for this test. The time-stepping is done using the classical fourth-order explicit Runge Kutta method (RK4). The node sets were standard icosahedral nodes from the Spherepts package [40]. The results are shown in Fig. 1(left). Fig. 1 (left) shows that though
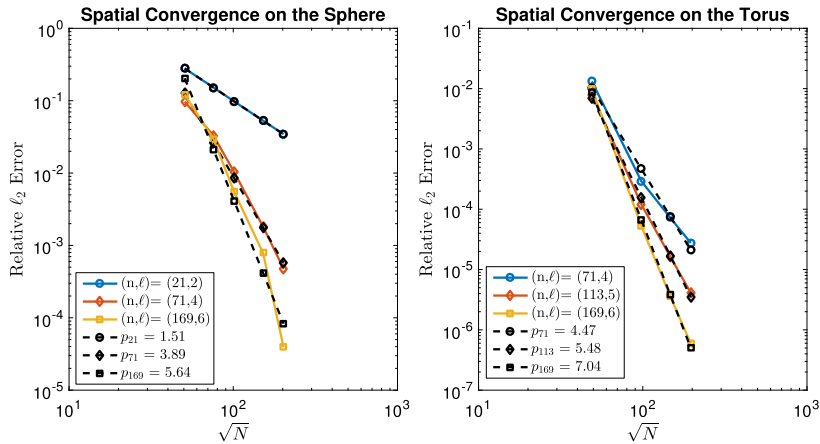
Fig. 1. Convergence on the sphere for the surface advection equation (left) and torus for the forced surface diffusion equation (right). The figure shows relative ℓ_2 error as a function of $\sqrt{N}$ for different values of stencil n and polynomial degree ℓ.

one expects a convergence rate of $\xi = \ell$, we appear to obtain slightly lower rates. The LOI tolerance parameter τ was set to $\tau = 1e - 2$ for $\ell = 2$, $\tau = 1e - 3$ for $\ell = 3$, and $\tau = 1e - 4$ for $\ell = 4$. In [16], a shape parameter was carefully tuned to avoid stagnation errors, and the parameters γ and k were numerically computed so as to avoid instabilities. In contrast, our approach only involves setting τ and estimating $\Lambda_{\max}$ very crudely (which is done rapidly). No shape parameters or tuning were required to obtain stability, and no extended precision arithmetic was required. The goal of this article is simply demonstrate the feasibility of RBF-LOI for PDEs on surfaces, and we hence defer a deeper investigation of RBF-LOI for advection on the sphere to future work. We note that while the stencil sizes in our work are larger than those used in [16], the increase in computational cost is largely ameliorated by the use of the overlapped RBF-FD method.

5.2. Diffusion on a torus $\mathbb{T}$

We consider the torus from [38] given by

$$\mathbb{T} = \left\{ \boldsymbol{X} = (x, y, z) \in \mathbb{R}^3 \ \middle| \ \left(1 - \sqrt{x^2 + y^2}\right)^2 + z^2 - \frac{1}{9} = 0 \right\}. \tag{26}$$

Our goal is to solve the forced diffusion equation given by

$$\frac{\partial c}{\partial t} = \Delta_{\mathbb{M}} c + f. \tag{27}$$

In all cases, we use the method of manufactured solutions, i.e., we prescribe a solution $c(\boldsymbol{x}, t)$ and calculate the forcing term $f(\boldsymbol{x}, t)$ that makes the solution hold. We use the BDF4 time-stepping scheme [2] for advancing the solution in time. This time-stepping scheme is fully implicit, and requires the solution of a sparse linear system every time-step. We set the time-step to $\Delta t = 10^{-3}$ for this test, and use Matlab's built-in sparse direct solver to solve the sparse linear systems obtained from overlapped RBF-FD. The manufactured solution in this case is given by

$$c(t, \phi, \lambda) = e^{-5t} \sum_{k=1}^{23} e^{-81(1-\cos(\lambda-\lambda_k))-9(1-\cos(\phi-\phi_k))}, \tag{28}$$

where the intrinsic coordinates $-\pi \le \phi, \lambda \le \pi$ parameterize the torus $\mathbb{T}$ in the usual way [38]. The solution is $C^\infty(\mathbb{T})$. The LOI tolerance was fixed at $\tau = 10^{-3}$ for $\ell = 4, 5$, and decreased to $\tau = 10^{-4}$ for $\ell = 6$. The results are shown in Fig. 1 (right). We obtain a convergence rate of approximately $\ell + 1$ on the torus. Once again, no extended-precision arithmetic was used; all calculations are in double precision, with the cost of forming the larger differentiation matrices being almost completely ameliorated by the large speedup obtained from overlapped RBF-FD. Similar results were obtained for forced diffusion on the sphere as well (not shown).

5.3. Cost vs accuracy

To better understand the costs involved in both overlapped RBF-FD and its use within the RBF-LOI algorithm, we now study computational cost (measured in wall-clock time) as a function of accuracy for different values of ξ (and therefore n and ℓ). Since the overlapped RBF-FD method is used only to speed up the assembly of the differentiation matrices, we
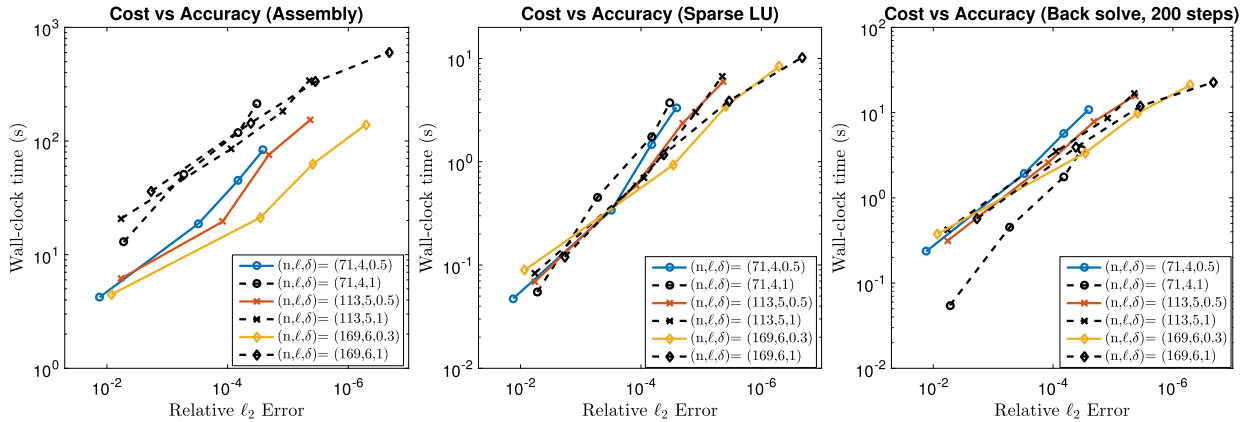
Fig. 2. Cost versus accuracy on the torus for the forced surface diffusion equation. The figures show wall-clock time as a function of relative ℓ_2 error for different values of stencil n, polynomial degree ℓ, and overlap parameter δ. $\delta = 1$ corresponds to the standard RBF-FD method (dashed lines). The figures compare costs for the assembly stage (left), sparse LU factorization stage (middle), and the actual back-solve stage (right). In all cases, the ℓ_2 error is a function of $\sqrt{N}$, which is increasing left to right.

present separate cost versus accuracy results for the assembly stage, the sparse LU factorization stage (preprocessing for solving the linear system), and the actual solution of the PDE using back solves, all with and without the use of overlapped RBF-FD. To ensure that the cost measured reflects matrix inversion rather than just matrix multiplies, we focus on the case of diffusion on the torus, which requires a sparse linear solve every step. The results are shown in Fig. 2.

Fig. 2 (left) shows that the overlapped RBF-FD method results in significant cost savings at the assembly stage for comparable accuracy; the method lags behind the standard RBF-FD method ($\delta = 1$) in accuracy only on the finest node set for $\ell = 6$. Fig. 2 (middle) shows that the LU factorization costs are all comparable across different values of δ, with the costs appearing to increase slowly with n and ℓ. Finally, Fig. 2 (right) shows that the back solve costs are also mostly comparable across all methods, with $\ell = 4$ being the exception; in this case, $\delta = 1$ appears to be cheaper, possibly due to slightly different matrix structure. It is important to note that if one sums up costs across the subfigures of Fig. 2 for $\delta = 1$, the assembly cost dominates all other costs. In contrast, for the overlapped RBF-FD method, the assembly cost is comparable to the back-solve cost. This feature would be beneficial when solving a problem on a moving domain.

6. Applications

Having validated our the RBF-LOI method on standard test cases, we now turn our attention to some applications. Our goal here is demonstrate that the RBF-LOI method is stable on different point cloud surfaces *and* PDEs that are more complicated than the forced diffusion equation. To that end, we test on three manifolds of increasing genus:

1. The red blood cell (genus 0), a parametric surface with node sets and normals obtained using the techniques outlined in [36];
2. Dupin's cyclide (genus 1), an implicit surface with node sets and normals obtained using Meshlab [8]; and
3. The double torus (genus 2), another implicit surface with node sets and normals again obtained using Meshlab.

Unfortunately, unlike in [25,38], we were unable to find stable parameters for point cloud models of more complicated manifolds such as frogs and bunnies. It is likely that such surfaces would require an adaptive tolerance selection for the LOI procedure, which we leave for future work. The problem of advection on arbitrary surfaces also requires a very careful derivation of hyperviscosity parameters γ and k, which is likewise beyond the scope of this article. We instead focus on biologically-motivated reaction–diffusion models involving nonlinear terms. We believe that these applications serve as a convincing demonstration of the simplicity and effectiveness of the RBF-LOI method. In all cases, we use the following parameters for RBF-LOI: $\ell = 4$, $m = 2\ell + 1$, tolerance of $\tau = 10^{-3}$. All simulations used the SBDF2 method for time-stepping, with corresponding linear systems being solved by the built-in Matlab sparse direct solver.

6.1. Cahn–Hilliard on a red blood cell

Our first application is the simulation of the Cahn–Hilliard equation on an idealized red blood cell [24]. The Cahn–Hilliard equation is a nonlinear PDE governing phase separation, and is given by

$$\frac{\partial c}{\partial t} = \nu \Delta_{\mathbb{M}} c^3 - \nu \Delta_{\mathbb{M}} c - \nu \gamma \Delta_{\mathbb{M}}^2 c, \tag{29}$$
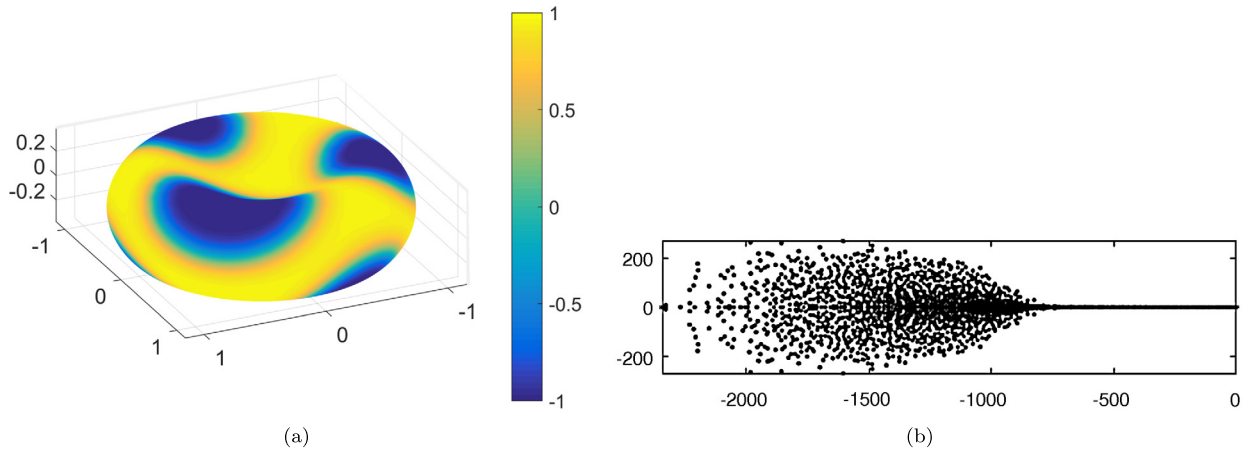
Fig. 3. Solution of the Cahn–Hilliard equation on the red blood cell at time $T = 2$ (left) and eigenvalues of the Laplacian (right). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

where $\Delta_\mathbb{M}^2$ is the surface bilaplacian. The solutions $c = 1$ and $c = -1$ both constitute critical points of this reaction–diffusion system, and any initial condition will be separated over time into these two phases. We simulate the above PDE on the red blood cell to time $t = 2$ using a random initial condition. To approximate the surface bilaplacian, we first form the discrete surface Laplacian L, then simply compute the discrete surface bilaplacian B as $B = L.L$. This has the effect of increasing the fill-in of B when compared to L, but our goal here is to simply demonstrate effectiveness. We use $\gamma = 0.006$ and $\nu = 0.5$, and a time-step of $\Delta t = 10^{-4}$. This small step is primarily due to the stiff nonlinear term $\nu \Delta_\mathbb{M} c^3$ being stepped explicitly in time. The results for $N = 2553$ nodes are shown in Fig. 3(a), and the spectrum of the discrete surface Laplacian is shown in Fig. 3(b). Clearly, our solutions are exhibiting the correct qualitative behavior, and the spectrum of L contains no spurious eigenvalues. To obtain the same behavior without LOI, careful tuning of the shape parameter on a stencil-by-stencil basis was needed in [38], and a stencil selection algorithm was needed in [25].

6.2. Fitzhugh–Nagumo waves on Dupin's cyclide

Following [24], we simulate the Fitzhugh–Nagumo reaction–diffusion system on Dupin's cyclide. The reaction–diffusion model is given by:

$$\frac{\partial c_1}{\partial t} = \delta_1 \Delta_\mathbb{M} c_1 + \frac{1}{0.02} c_1 (1 - c_1) \left(c_1 - \frac{c_2 + 0.02}{0.75} \right), \tag{30}$$

$$\frac{\partial c_2}{\partial t} = \delta_1 \Delta_\mathbb{M} c_2 + c_1 - c_2, \tag{31}$$

where c_1 and c_2 are typically viewed as chemical concentrations or densities corresponding to a membrane potential and a current, respectively. The above system is a simple model for the dynamics of excitable media, and is often viewed as a simplification of the Hodgkin-Huxley model for the dynamics of neurons [13,26]. Our initial condition on Dupin's cyclide is given by $c_1 = \frac{1}{2} (1 + \tanh(5x + y))$ and $c_2 = \frac{1}{2} (1 - \tanh(10z))$, where $\boldsymbol{x} = (x, y, z)$. The node sets on the cyclide are the same as those used in [24]; we use $N = 11884$ of these nodes. The results of the simulation with the SBDF2 method at time $t = 100$ are shown in Fig. 4(a), and the spectrum of the discrete Laplacian is shown in Fig. 4(b). As expected, the simulation results in spiral waves that scroll over the manifold.

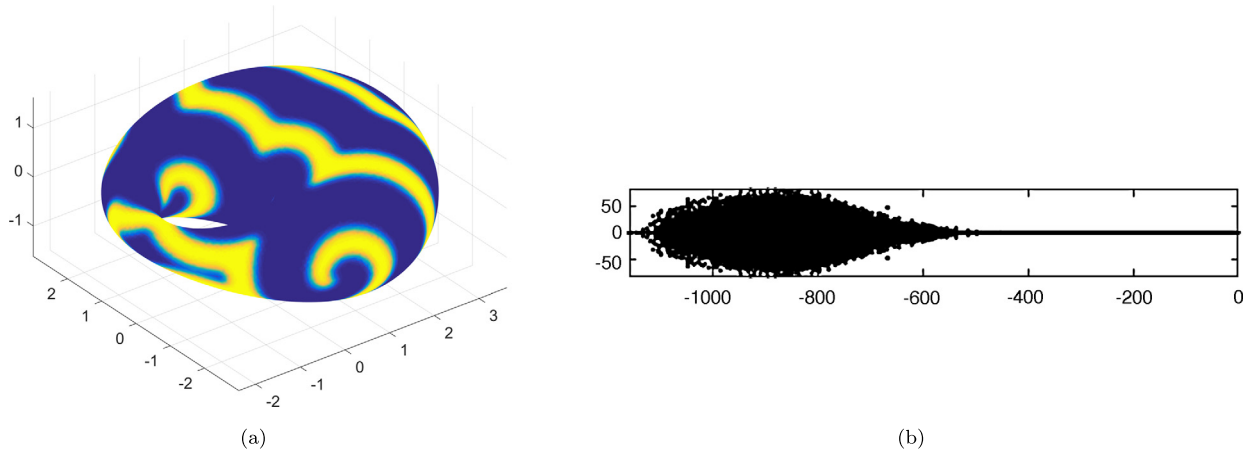### 6.3. Turing spots on the double torus

Our final application involves solving another coupled reaction–diffusion system on the double torus

$$\mathbb{T}^2 = \{\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3 \mid (x^2(1 - x^2) - y^2)^2 + 0.5z^2 = \frac{1}{40}\}, \tag{32}$$
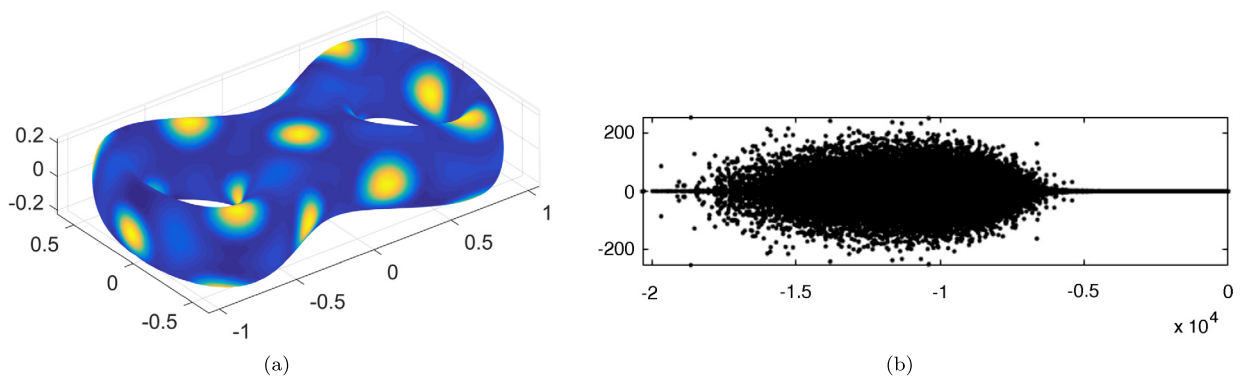
which a genus-2 surface obtained as the *join* of two genus-1 tori. On this surface, we simulate the Turing system given by

$$\frac{\partial c_1}{\partial t} = \delta_1 \Delta_\mathbb{M} c_1 + \alpha c_1 \left(1 - \tau_1 c_2^2 \right) + c_2 (1 - \tau_2 c_1), \tag{33}$$

$$\frac{\partial c_2}{\partial t} = \delta_2 \Delta_\mathbb{M} c_2 + \beta c_2 \left(1 + \frac{\alpha \tau_1}{\beta} c_1 c_2 \right) + c_1 (\gamma_1 + \tau_2 c_2), \tag{34}$$

Fig. 4. Solution of the Fitzhugh–Nagumo equations on Dupin's cyclide at time $T = 800$ (left) and eigenvalues of the Laplacian (right).

Fig. 5. Solution of the Turing equations on the double torus at time $T = 800$ (left) and eigenvalues of the Laplacian (right).

where 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 $\gamma_1 = -\alpha$. We use a time-step of $\Delta t = 0.01$ and simulate to a final time of $t = 800$ on $N = 12100$ nodes. The results are shown in Fig. 5a, and the spectrum of the discrete Laplacian L is shown in Fig. 5b. Fig. 5a shows that spot patterns have formed on the double torus despite the relatively coarse spatial discretization.

7. Summary and future work

We have proposed and demonstrated numerical solutions to PDEs on manifolds via RBF-LOI: a polynomially-augmented RBF-FD procedure. The major novel contribution of our work has been demonstration that a well-chosen polynomial basis (here, the Least Orthogonal Interpolant) along with efficient stencil overlap techniques for RBF-FD approximation can yield a stable, robust, efficient, and accurate PDE solver on manifolds. Our algorithm relies on specification of only two global parameters: an overlap parameter (which affects cost of local discrete operator construction), and a LOI tolerance parameter (which affects stability of polynomial approximations).

To tackle more general point clouds, we would like the LOI tolerance parameter to be automatically tunable on a per-stencil basis. Ongoing work revolves around devising an automated approach for this parameter, and application of the RBF-LOI to PDE solutions on more intricate manifolds. Future work would involve rigorously deriving a hyperviscosity formulation for PDEs on arbitrary orientable manifolds to facilitate the solution of hyperbolic problems with RBF-LOI.

Acknowledgements

References

[1] K.A. Aiton, A Radial Basis Function Partition of Unity Method for Transport on the Sphere, Master's thesis, Boise State University, USA, 2014.
[2] U.M. Ascher, S.J. Ruuth, B.T.R. Wetton, Implicit-explicit methods for time-dependent PDEs, SIAM J. Numer. Anal. 32 (1997) 797–823.
[3] G.A. Barnett, A Robust RBF-FD Formulation Based on Polyharmonic Splines and Polynomials, PhD thesis, University of Colorado Boulder, 2015.
[4] V. Bayona, N. Flyer, B. Fornberg, G.A. Barnett, On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic pdes, J. Comput. Phys. 332 (Supplement) (2017) C:257–273.
[5] V. Bayona, M. Moscoso, M. Carretero, M. Kindelan, RBF-FD formulas and convergence properties, J. Comput. Phys. 229 (22) (2010) 8281–8295.
[6] J. Behrens, A. Iske, Grid-free adaptive semi-Lagrangian advection using radial basis functions, Comput. Math. Appl. 43 (3) (2002) 319–327.
[7] C.D. Boor, A. Ron, Computational aspects of polynomial interpolation in several variables, Math. Comput. 58 (198) (1992) 705–727.
[8] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, MeshLab: an open-source mesh processing tool, in: V. Scarano, R.D. Chiara, U. Erra (Eds.), Eurographics Italian Chapter Conference. The Eurographics Association, 2008.
[9] O. Davydov, D.T. Oanh, Adaptive meshless centres and RBF stencils for Poisson equation, J. Comput. Phys. 230 (2) (2011) 287–304.
[10] O. Davydov, R. Schaback, Optimal stencils in Sobolev spaces, 2017, submitted for publication.
[11] G.E. Fasshauer, Meshfree Approximation Methods with MATLAB, Interdisciplinary Mathematical Sciences, vol. 6, World Scientific Publishers, Singapore, 2007.
[12] G.E. Fasshauer, M.J. McCourt, Stable evaluation of Gaussian radial basis function interpolants, SIAM J. Sci. Comput. 34 (2012) A737–A762.
[13] R. FitzHugh, Impulses and physiological states in theoretical models of nerve membrane, Biophys. J. 1 (6) (1961) 445–466.
[14] 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.
[15] 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.
[16] 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.
[17] N. Flyer, G.B. Wright, Transport schemes on a sphere using radial basis functions, J. Comput. Phys. 226 (2007) 1059–1084.
[18] N. Flyer, G.B. Wright, A radial basis function method for the shallow water equations on a sphere, Proc. R. Soc. A 465 (2009) 1949–1976.
[19] B. Fornberg, E. Larsson, N. Flyer, Stable computations with Gaussian radial basis functions, SIAM J. Sci. Comput. 33 (2) (2011) 869–892.
[20] B. Fornberg, E. Lehto, Stabilization of RBF-generated finite difference methods for convective PDEs, J. Comput. Phys. 230 (2011) 2270–2285.
[21] B. Fornberg, E. Lehto, C. Powell, Stable calculation of Gaussian-based RBF-FD stencils, Comput. Math. Appl. 65 (2013) 627–637.
[22] B. Fornberg, C. Piret, A stable algorithm for flat radial basis functions on a sphere, SIAM J. Sci. Comput. 30 (2007) 60–80.
[23] B. Fornberg, G. Wright, Stable computation of multiquadric interpolants for all values of the shape parameter, Comput. Math. Appl. 48 (2004) 853–867.
[24] E.J. Fuselier, G.B. Wright, A high-order kernel method for diffusion and reaction–diffusion equations on surfaces, J. Sci. Comput. 56 (3) (2013) 535–565.
[25] 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. 39 (5) (2017) A2129–A2151.
[26] J. Nagumo, S. Arimoto, S. Yoshizawa, An active pulse transmission line simulating nerve axon, Proc. IRE 50 (10) (1962) 2061–2070.
[27] 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.
[28] A. Narayan, D. Xiu, Stochastic collocation methods on unstructured grids in high dimensions via interpolation, SIAM J. Sci. Comput. 34 (3) (2012) A1729–A1752.
[29] C. Piret, The orthogonal gradients method: a radial basis functions method for solving partial differential equations on arbitrary surfaces, J. Comput. Phys. 231 (20) (2012) 4662–4675.
[30] C. Piret, J. Dunn, Fast rbf ogr for solving pdes on arbitrary surfaces, AIP Conf. Proc. 1776 (1) (2016).
[31] J.A. Reeger, B. Fornberg, Numerical quadrature over the surface of a sphere, Stud. Appl. Math. 137 (2) (2016) 174–188.
[32] J.A. Reeger, B. Fornberg, Numerical quadrature over smooth surfaces with boundaries, J. Comput. Phys. 355 (Supplement C) (2018) 176–190.
[33] J.A. Reeger, B. Fornberg, M.L. Watts, Numerical quadrature over smooth, closed surfaces, Proc. R. Soc. Lond. A: Math. Phys. Eng. Sci. 472 (2194) (2016).
[34] R. Schaback, Multivariate interpolation by polynomials and radial basis functions, Constr. Approx. 21 (3) (2005) 293–317.
[35] V. Shankar, The overlapped radial basis function-finite difference (RBF-FD) method: a generalization of RBF-FD, J. Comput. Phys. 342 (2017) 211–228.
[36] V. Shankar, R.M. Kirby, A.L. Fogelson, Robust node generation for meshfree discretizations on irregular domains and surfaces, SIAM J. Sci. Comput. (2018), in press.
[37] V. Shankar, G.B. Wright, Mesh-free semi-Lagrangian methods for transport on a sphere using radial basis functions, J. Comput. Phys. 366(C) (2018) 170–190.
[38] 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.
[39] H. Wendland, Scattered Data Approximation, Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge University Press, Cambridge, 2005.
[40] G.B. Wright, SpherePts, https://github.com/gradywright/spherepts/, 2018.
[41] 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.
[42] G.B. Wright, B. Fornberg, Stable computations with flat radial basis functions using vector-valued rational approximations, J. Comput. Phys. 331 (2017) 137–156.