A RADIAL BASIS FUNCTION (RBF) COMPACT FINITE DIFFERENCE (FD) SCHEME FOR REACTION-DIFFUSION EQUATIONS ON SURFACES

ERIK LEHTO*, VARUN SHANKAR†, AND GRADY B. WRIGHT‡

Abstract. We present a new high-order, local meshfree method for numerically solving reaction diffusion equations on smooth surfaces of co-dimension one embedded in $\mathbb{R}^d$. The novelty of the method is in the approximation of the Laplace-Beltrami operator for a given surface using Hermite radial basis function (RBF) interpolation over local node sets on the surface. This leads to compact (or implicit) RBF generated finite difference (RBF-FD) formulas for the Laplace-Beltrami operator, which gives rise to sparse differentiation matrices. The method only requires a set of (scattered) nodes on the surface and an approximation to the surface normal vectors at these nodes. Additionally, the method is based on Cartesian coordinates and thus does not suffer from any coordinate singularities. We also present an algorithm for selecting the nodes used to construct the compact RBF-FD formulas that can guarantee the resulting differentiation matrices have desirable stability properties. The improved accuracy and computational cost that can be achieved with this method over the standard (explicit) RBF-FD method are demonstrated with a series of numerical examples. We also illustrate the flexibility and general applicability of the method by solving two different reaction diffusion equations on surfaces that are defined implicitly and only by point clouds.

Key words. RBF-FD, RBF-HFD, manifolds, reaction diffusion

AMS subject classifications. 41A21, 65D05, 65E05, 65F22

1. Introduction.

Global radial basis function (RBF) methods are quite popular for the numerical solution of various partial differential equations (PDEs) due to their ability to handle scattered node layouts, their simplicity of implementation, and their potential for spectral accuracy for smooth problems. These methods have been successfully applied to the solution of PDEs in various geometries in $\mathbb{R}^2$ and $\mathbb{R}^3$ (e.g., [11,16]), including spherical domains (e.g. [15,26,41]), and more general surfaces embedded in $\mathbb{R}^3$(e.g. [25,35]).

When high orders of algebraic accuracy are sufficient for a given problem, or if the solutions to the problem are expected to only have finite-smoothness, RBF generated finite difference (RBF-FD) formulas are an attractive alternative to global RBFs as they perform better in terms of accuracy per computational cost [16]. These formulas are generated from RBF interpolation over *local* sets of nodes (stencils) so that the resulting differentiation matrices are sparse like in the standard FD method. In contrast to standard FD methods, however, the RBF-FD method can naturally handle irregular geometries and scattered node layouts. Additionally, their locality makes them more flexibility in terms of local refinement strategies than global RBF methods. The strength of the RBF-FD method has been leveraged to solve problems on planar domains, e.g., [5,6,38,39,42], the surface of a sphere [14,18], and more recently, very general surfaces represented solely by point clouds and normal vectors [37].

It is natural to view these two classes of RBF methods as extensions of classical methods to scattered nodes and irregular geometries. The global RBF method for surface PDEs in [25] may be viewed as an extension of polynomial based (or Fourier based) pseudospectral methods to surfaces, while the RBF-FD method presented

*Department of Mathematics, KTH Royal Institute of Technology, Sweden (elehto@kth.se). Corresponding author.

†Department of Mathematics, University of Utah, USA (vshankar@math.utah.edu).

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

in [37] may be viewed as an extension of standard, polynomial based FD methods to surfaces. In this work, we turn our attention to the extension of a third important class of classical methods to surfaces: the so-called compact, implicit, or Hermite FD methods, first introduced by Collatz [8]. We use the acronym HFD for these schemes to avoid the obvious confusion with CFD, and because they will ultimately be based on Hermite interpolation.

The goal of HFD methods is to solve a given PDE numerically by computing more accurate approximations to the differential operators in the PDE. In these schemes, this improved accuracy is obtained by using additional information from the PDE itself, rather than increasing the stencil size, as is the usual way to increase the accuracy with standard (or explicit) FD methods. HFD schemes are thus typically more computationally efficient than standard FD schemes, as they can obtain higher accuracy and resolution for the same stencil size [30]. Further, the differentiation matrices obtained from HFD formulas often have desirable properties such as diagonal dominance, leading to both enhanced numerical stability and faster convergence of iterative methods used in solving the sparse linear systems that arise when using these matrices to discretize a PDE. While HFD schemes have already been successfully generalized to scattered node layouts [42], the application of these schemes to the solution of PDEs on surfaces presents significant challenges due to the presence of surface differential operators. In this article, we overcome those challenges and present a new RBF-HFD scheme for the solution of reaction-diffusion equations on surfaces.[1] The resulting method uses Cartesian coordinates, thereby avoiding the singularities typically associated with intrinsic coordinate systems. Further, our new method only uses nodes on the surface in consideration, making it more computationally efficient than embedded narrow-band methods that solve the PDE in a narrow-band in the embedding space (e.g., [31, 35]). Finally, the RBF-HFD formulas require fewer nodes than the RBF-FD method presented in [37] for the same accuracy, while also possessing improved stability properties.

The remainder of the paper is organized as follows. In Section 2, we briefly review the formulation of surface differential operators in Cartesian coordinates. Section 3 outlines Hermite RBF interpolation on scattered node sets in $\mathbb{R}^d$. Section 4 describes how to use approximations to the Hermite RBF interpolants to generate RBF-HFD weights for approximating the surface Laplacian, and also how these can be arranged into *sparse* differentiation matrices. We follow this in Section 5 with a brief discussion of how to use these differentiation matrices in a method-of-lines formulation for numerically solving forced diffusion equations on surfaces. In Section 6, we discuss the stability of our method by studying the Gershgorin sets associated with the eigenvalues of our differentiation matrices. In Section 7, we numerically demonstrate the accuracy and efficiency of our method for the forced scalar diffusion equation on two different surfaces. We also present a few applications of our method to two species reaction diffusion equations on implicitly defined surfaces and surfaces defined by point clouds, which have relevant biological applications. We conclude our paper with a summary and discussion of future research directions in Section 8.

2. Review of Differential Geometry. While the standard way of expressing differential operators on surfaces is through the use of intrinsic coordinates, covariant derivatives and metric tensors [4], we instead choose to formulate these operators entirely in Cartesian (or extrinsic) coordinates, as this avoids any singularities asso-

[1] Throughout this paper, we will use the terms surface or manifold to refer to embedded submanifolds of codimension one in $\mathbb{R}^d$ with no boundary, and focus on the specific case of $d = 3$.

ciated with intrinsic coordinate systems. Consider the standard gradient operator in $\mathbb{R}^3$, $\nabla = \begin{bmatrix}\partial_x & \partial_y & \partial_z\end{bmatrix}^T$. If we apply this to a differentiable function f at a point $\mathbf{x} = (x, y, z)$ on the surface $\mathbb{M}$ and then project the resulting vector into the tangent space of the surface, then this gives the surface gradient of f, which we denote as $\nabla_{\mathbb{M}} f$. Mathematically, this can be accomplished as follows. Let $\mathbf{n} = \begin{bmatrix}n^x & n^y & n^z\end{bmatrix}^T$ be the *unit* normal vector to $\mathbb{M}$ at $\mathbf{x}$, then

$$\nabla_{\mathbb{M}} f = \nabla f - \mathbf{n}(\mathbf{n} \cdot \nabla f) = \nabla f - \mathbf{n}\mathbf{n}^T(\nabla f).$$

Thus, the surface gradient operator can written entirely in Cartesian coordinates as

$$\nabla_{\mathbb{M}} := \nabla - \mathbf{n}\mathbf{n}^T\nabla = \underbrace{(\mathcal{I} - \mathbf{n}\mathbf{n}^T)}_{\mathcal{P}}\nabla,$$

where $\mathcal{I}$ is the 3-by-3 identity matrix. $\mathcal{P}$ is a *projection operator* that takes a vector field in $\mathbb{R}^3$ sampled at a point $\mathbf{x}$ on the surface and projects it onto the tangent plane to the surface at $\mathbf{x}$. An explicit expression for this operator is given by

$$\mathcal{P} = \begin{bmatrix}(1 - n^x n^x) & -n^x n^y & -n^x n^z \\ -n^x n^y & (1 - n^y n^y) & -n^y n^z \\ -n^x n^z & -n^y n^z & (1 - n^z n^z)\end{bmatrix} = \begin{bmatrix}\mathbf{p}^x & \mathbf{p}^y & \mathbf{p}^z\end{bmatrix}, \tag{1}$$

where $\mathbf{p}^x$, $\mathbf{p}^y$ and $\mathbf{p}^z$ are vectors representing the projection operators in the x, y and z directions, respectively. We can now use $\mathbf{p}^x$, $\mathbf{p}^y$ and $\mathbf{p}^z$ to obtain the following (more convenient) expression for $\nabla_{\mathbb{M}}$:

$$\nabla_{\mathbb{M}} := \mathcal{P}\nabla = \begin{bmatrix}\mathbf{p}^x \cdot \nabla \\ \mathbf{p}^y \cdot \nabla \\ \mathbf{p}^z \cdot \nabla\end{bmatrix} = \begin{bmatrix}\mathcal{G}^x \\ \mathcal{G}^y \\ \mathcal{G}^z\end{bmatrix}, \tag{2}$$

where $\mathcal{G}^x$, $\mathcal{G}^y$ and $\mathcal{G}^z$ are the components of the surface gradient along each of the coordinate directions in $\mathbb{R}^3$. Now, the surface Laplace (Laplace-Beltrami) operator $\Delta_{\mathbb{M}}$ can be obtained by applying the surface divergence to the surface gradient [25]. This can naturally be expressed using $\mathcal{G}^x$, $\mathcal{G}^y$, and $\mathcal{G}^z$ as

$$\Delta_{\mathbb{M}} := \nabla_{\mathbb{M}} \cdot \nabla_{\mathbb{M}} = (\mathcal{P}\nabla) \cdot \mathcal{P}\nabla = \mathcal{G}^x\mathcal{G}^x + \mathcal{G}^y\mathcal{G}^y + \mathcal{G}^z\mathcal{G}^z. \tag{3}$$

This gives an explicit expression for the surface Laplace operator entirely in Cartesian components. We will use this expression in our numerical approximation to the surface Laplacian.

3. Hermite Interpolation with RBFs. We now review Hermite interpolation with RBFs, a technique essential to deriving the new RBF-HFD scheme outlined in the next section. Let $\Omega \subseteq \mathbb{R}^d$, and $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a scalar-valued radial kernel, i.e., $\phi(\mathbf{x}, \mathbf{y}) := \phi(\|\mathbf{x} - \mathbf{y}\|)$ for $\mathbf{x}, \mathbf{y} \in \Omega$, where $\|\cdot\|$ is the standard Euclidean norm in $\mathbb{R}^d$. Let $\mathcal{L}$ be a linear functional and suppose we are given samples of a continuous target function f at a set of distinct nodes $X = \{\mathbf{x}_i\}_{i=1}^n \subset \Omega$ and samples of $\mathcal{L}f$ at a set of distinct nodes $\tilde{X} = \{\tilde{\mathbf{x}}_j\}_{j=1}^m \subset \Omega$. Then we consider the following Hermite RBF interpolant to the this data, proposed first by Wu [43]:

$$I_\phi f(\mathbf{x}) = \sum_{i=1}^{n} c_i \phi(\mathbf{x}, \mathbf{x}_i) + \sum_{j=1}^{m} d_j \mathcal{L}_2 \phi(\mathbf{x}, \tilde{\mathbf{x}}_j) + \alpha. \tag{4}$$

Here we have used of notation $\mathcal{L}_2$ to mean that $\mathcal{L}$ is applied to ϕ with respect to its second argument. Later we will similarly use $\mathcal{L}_1$ to mean that $\mathcal{L}$ is applied to ϕ with respect to its first argument. The expansion coefficients $\{c_i\}_{i=1}^n$ and $\{d_j\}_{j=1}^m$ in (4) are determined by enforcing the (Hermite) interpolation conditions

$$I_\phi f|_X = f|_X\,, \tag{5}$$

$$\mathcal{L}\,(I_\phi f)|_{\tilde{X}} = (\mathcal{L} f)|_{\tilde{X}}\,, \tag{6}$$

while the constant α is obtained by enforcing the moment condition $\sum_{i=1}^n c_i = 0$. These conditions can be represented as the following block linear system:

$$\underbrace{\begin{bmatrix} A & B_2 & \mathbf{e} \\ B_1 & C & \mathbf{0} \\ \mathbf{e}^T & \mathbf{0}^T & 0 \end{bmatrix}}_{A_{\mathrm{H}}} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{d} \\ \alpha \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\mathcal{L}f} \\ 0 \end{bmatrix}, \tag{7}$$

where

$$\begin{aligned} A_{i,j} &= \phi(\mathbf{x}_i, \mathbf{x}_j), i, j = 1, \ldots, n, \\ (B_2)_{i,j} &= \mathcal{L}_2\phi(\mathbf{x}_i, \tilde{\mathbf{x}}_j), i = 1, \ldots, n, j = 1, \ldots, m, \\ (B_1)_{i,j} &= \mathcal{L}_1\phi(\tilde{\mathbf{x}}_i, \mathbf{x}_j), i = 1, \ldots, m, j = 1, \ldots, n, \\ C_{i,j} &= \mathcal{L}_1\mathcal{L}_2\phi(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j), i, j = 1, \ldots, m, \\ \mathbf{e}_i &= 1, i = 1, \ldots, n. \end{aligned}$$

Because ϕ is radially symmetric we have that $\phi(\mathbf{x}, \tilde{\mathbf{x}}) = \phi(\tilde{\mathbf{x}}, \mathbf{x})$, so that $\mathcal{L}_2\phi(\mathbf{x}, \tilde{\mathbf{x}}) = \mathcal{L}_1\phi(\tilde{\mathbf{x}}, \mathbf{x})$. This means that $A = A^T$, $C = C^T$, and $B_2 = B_1^T$ so that the matrix A_{H} is symmetric. If ϕ is, for example, positive definite or order one conditionally positive definite, then under very mild conditions on $\mathcal{L}$, the linear system (7) is non-singular [32, 43].

We will also make use of regular RBF interpolation, which consists only of interpolating function values, in the subsequent section. These interpolants are simply given by (4) with m set equal to zero and the constant α omitted, i.e.,

$$I_\phi f(\mathbf{x}) = \sum_{i=1}^n c_i \phi(\mathbf{x}, \mathbf{x}_i). \tag{8}$$

In this case, we only enforce the conditions (5), which can be represented by the linear system

$$A\boldsymbol{c} = \boldsymbol{f}. \tag{9}$$

We will use the subscript R to denote the linear system for the *regular* interpolant as opposed to the subscript H in (7) for the linear system associated with the *Hermite* interpolant.

In this study, the interpolation nodes X and "functional nodes" $\tilde{X}$ lie on an embedded lower dimensional surface $\Omega = \mathbb{M}$ in $\mathbb{R}^d$. However, we will still use the standard Euclidean distance in $\mathbb{R}^d$ for computing $\phi(\mathbf{x}, \tilde{\mathbf{x}}) = \phi(\|\mathbf{x} - \tilde{\mathbf{x}}\|)$ in Equation (4) (i.e., straight line distances rather than distances intrinsic to the surface). A theoretical foundation for RBF interpolation on surfaces with the straight-line distance measure is given in [23], along with proofs of favorable error estimates.

While it is possible to use any (conditionally) positive-definite kernel within the RBF-FD and RBF-HFD method (e.g., [3, 9, 14, 38, 42]), we use the Gaussian (GA) kernel, which is positive definite in $\mathbb{R}^d$, for any d. All infinitely smooth kernels feature a shape parameter ε such that large values of ε make the kernels peaked, while smaller ε values make them flat. In the limit as $\varepsilon \to 0$, Gaussian RBF interpolants to data converge to (multivariate) polynomial interpolants in $\mathbb{R}^d$ [10, 28, 36], and to spherical harmonic interpolants on the sphere $\mathbb{S}^2$ [20]. While smaller values of ε generally lead to greater accuracy for smooth target functions, [21, 28], the interpolation matrix in Equation (7) becomes increasingly ill-conditioned as $\varepsilon \to 0$ (see, e.g., [22]). Some stable algorithms have been developed for bypassing this ill-conditioning [12,17,19–21], but these algorithms typically break down when the data sites lie on a submanifold $\mathbb{M} \subset \mathbb{R}^d$, as in the present study, due to nodes being non-unisolvent with respect to polynomials in $\mathbb{R}^d$. While some strategies have recently been undertaken to resolve them in $\mathbb{R}^d$ [29], a robust approach is not yet available for surfaces.

4. RBF-HFD formulas for the surface Laplacian. Let $\Xi = \{\boldsymbol{\xi}_k\}_{k=1}^N$ denote a set of (scattered) node locations on a surface $\mathbb{M}$ of dimension two embedded in $\mathbb{R}^3$ and suppose $f : \mathbb{M} \to \mathbb{R}$ is some differentiable function sampled on Ξ. Our goal is to approximate $\Delta_{\mathbb{M}} f|_{\Xi}$ with HFD-style local approximations to the operator $\Delta_{\mathbb{M}}$. Without loss of generality, let the node where we want to approximate $\Delta_{\mathbb{M}} f$ at be $\boldsymbol{\xi}_1$, and let $\boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_p$ be the $p-1$ nearest neighbors to $\boldsymbol{\xi}_1$, measured by Euclidean distance in $\mathbb{R}^3$. We refer to $\boldsymbol{\xi}_1$ and its $p-1$ nearest neighbors as the neighborhood of $\boldsymbol{\xi}_1$ on the surface and denote this neighborhood as $S_1 = \{\boldsymbol{\xi}_\ell\}_{\ell=1}^p$; this neighborhood will comprise the candidate nodes that make up the HFD stencil for $\boldsymbol{\xi}_1$. We seek an approximation to $\Delta_{\mathbb{M}} f$ at $\boldsymbol{\xi}_1$ that involves a linear combination of the values of f and $\Delta_{\mathbb{M}} f$ over some subset of nodes from S_1 of the form

$$(\Delta_{\mathbb{M}} f)\big|_{\mathbf{x}=\boldsymbol{\xi}_1} \approx \sum_{i \in J} w_i f(\boldsymbol{\xi}_i) + \sum_{j \in \tilde{J}} \tilde{w}_j (\Delta_{\mathbb{M}} f)\big|_{\mathbf{x}=\boldsymbol{\xi}_j}, \tag{10}$$

where J and $\tilde{J}$ denote index sets of size $n \leq p$ and $m < p$, respectively, into S_1 for the explicit and the implicit (or Hermite) part of the stencil, respectively. We will assume that $1 \in J$, but $1 \notin \tilde{J}$ (otherwise a trivial solution would exist). Using the notation of the previous section, we will let the n nodes indicated by J be denoted by $X = \{\mathbf{x}_i\}_{i=1}^n$ and the m nodes indicated by $\tilde{J}$ be denoted by $\tilde{X} = \{\tilde{\mathbf{x}}_j\}_{j=1}^m$. Additionally, we always set $\mathbf{x}_1 = \boldsymbol{\xi}_1$. Using this notation we can rewrite (10) as

$$(\Delta_{\mathbb{M}} f)\big|_{\mathbf{x}=\mathbf{x}_1} \approx \sum_{i=1}^{n} w_i f(\mathbf{x}_i) + \sum_{j=1}^{m} \tilde{w}_j (\Delta_{\mathbb{M}} f)\big|_{\mathbf{x}=\tilde{\mathbf{x}}_j}. \tag{11}$$

The weights $\{w_i\}_{i=1}^n$ and $\{\tilde{w}_j\}_{j=1}^m$ in this approximation will be computed using RBFs, and will be referred to as RBF-HFD weights.

4.1. Computation of the weights from the Hermite interpolant. The method from [42] determines the RBF-HFD weights in (11) from the Hermite RBF interpolant (4) constructed with $\mathcal{L} = \Delta_{\mathbb{M}}$. To compute the weights consider the problem of applying $\Delta_{\mathbb{M}}$ to the interpolant (4) and evaluating it at $\mathbf{x}_1$ to approximate $\Delta_{\mathbb{M}} f\big|_{\mathbf{x}=\mathbf{x}_1}$. The resulting approximation would be exact whenever f is any of the functions $\phi(\mathbf{x}, \mathbf{x}_i)$, $i = 1, \ldots, n$, $\mathcal{L}_2 \phi(\mathbf{x}, \tilde{\mathbf{x}}_j) = \Delta_{\mathbb{M},2} \phi(\mathbf{x}, \tilde{\mathbf{x}}_j)$, $j = 1, \ldots, m$, or a non-zero constant (since the interpolant is exact for these f). Thus, the weights $\{w_i\}_{i=1}^n$

and $\{\tilde{w}_j\}_{j=1}^m$ are the values that make (11) exact for these values of f. This can be written as the following linear system

$$\underbrace{\begin{bmatrix} A & B & \mathbf{e} \\ B^T & C & \mathbf{0} \\ \mathbf{e}^T & \mathbf{0}^T & 0 \end{bmatrix}}_{A_{\mathrm{H}}} \begin{bmatrix} w \\ \tilde{w} \\ \alpha \end{bmatrix} = \begin{bmatrix} \Delta_{\mathbb{M},1}\phi(\mathbf{x},\mathbf{x}_i)\big|_{\mathbf{x}=\mathbf{x}_1} \\ \Delta_{\mathbb{M},1}\Delta_{\mathbb{M},2}\phi(\mathbf{x},\tilde{\mathbf{x}}_j)\big|_{\mathbf{x}=\mathbf{x}_1} \\ 0 \end{bmatrix}, \tag{12}$$

where the block matrices A and C are the same as those given in the Hermite interpolation matrix (7), $B = B_2 = B_1^T$ in this same matrix (recall that the matrix in (7) is symmetric), $\Delta_{\mathbb{M},1} = \mathcal{L}_1$ and $\Delta_{\mathbb{M},2} = \mathcal{L}_2$, and $i = 1, \ldots, n$ and $j = 1, \ldots, m$ to form block vectors of length n and m in the right hand side. Note that the constant α is not used for anything in the actual RBF-HFD formula.

The issue with using (12) for determining the RBF-FD weights is that one has to explicitly compute $\Delta_{\mathbb{M},1}\phi(\mathbf{x},\tilde{\mathbf{x}}_j)$ and $\Delta_{\mathbb{M},1}\Delta_{\mathbb{M},2}\phi(\mathbf{x},\tilde{\mathbf{x}}_j)$. As discussed in Section 2, constructing $\Delta_{\mathbb{M}}$ requires having explicit information about the underlying surface, such as an analytical expression for the surface normal vectors. Even in cases where these are known, the resulting formulas for computing $\Delta_{\mathbb{M},1}\phi$ and $\Delta_{\mathbb{M},1}\Delta_{\mathbb{M},2}\phi$ are likely to be quite complex. Moreover, we are interested in surfaces that are defined by point clouds and where only numerical representations of the normal vectors are available. Thus, constructing the system (12) analytically will not be possible. However, it is possible to construct an approximation to the entries of this system using the regular RBF-FD method from [37], which is based on iterated differentiation (see also [24]). This is the approach we take.

4.2. Computation of the weights from iterated differentiation. The first goal is to compute approximations of the entries in B^T in (12) and the entries of the first vector block in the right hand side of this equation. We state the entries of B^T explicitly as it will help elucidate the discussion of their approximation:

$$B^T = \begin{bmatrix} \Delta_{\mathbb{M},1}\phi(\tilde{\mathbf{x}}_1,\mathbf{x}_1) & \cdots & \Delta_{\mathbb{M},1}\phi(\tilde{\mathbf{x}}_1,\mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ \Delta_{\mathbb{M},1}\phi(\tilde{\mathbf{x}}_m,\mathbf{x}_1) & \cdots & \Delta_{\mathbb{M},1}\phi(\tilde{\mathbf{x}}_m,\mathbf{x}_n) \end{bmatrix}. \tag{13}$$

We compute these approximations by constructing an approximation to $\Delta_{\mathbb{M}}$ using discrete approximations to $\mathcal{G}^x$, $\mathcal{G}^y$, and $\mathcal{G}^z$ in (2) computed from the standard RBF interpolant (8) over the candidate stencil nodes $S_1 = \{\boldsymbol{\xi}_k\}_{k=1}^p$. To this end, consider, for example, applying $\mathcal{G}^x$ to the interpolant $I_\phi f$ in (8) based on the nodes in S_1 (here the target function f is not important) and then evaluating it at S_1:

$$(\mathcal{G}^x I_\phi f(\mathbf{x}))\big|_{\mathbf{x}=\boldsymbol{\xi}_i} = \sum_{j=1}^{p} c_j \underbrace{\left(\mathcal{G}^x\phi(\mathbf{x},\boldsymbol{\xi}_j)\right)\big|_{\mathbf{x}=\boldsymbol{\xi}_i}}_{D^x_{ij}}, \; i = 1, \ldots, p. \tag{14}$$

We can rewrite (14) so that it explicitly depends only on the vector of samples $f\big|_{S_1}$ using (9) as follows:

$$(\mathcal{G}^x I_\phi f)\big|_{S_1} = D^x c_f = D^x A_{\mathrm{R}}^{-1} f\big|_{S_1} =: G^x f\big|_{S_1}. \tag{15}$$

Here G^x is a p-by-p differentiation matrix that represents the RBF approximation to the x-component of the surface gradient operator over the set of nodes in S_1. Now,

letting

$$D^y_{i,j} = \left(\mathcal{G}^y \phi(\mathbf{x}, \boldsymbol{\xi}_j)\right)\Big|_{\mathbf{x}=\boldsymbol{\xi}_i} \text{ and } D^z_{i,j} = \left(\mathcal{G}^z \phi(\mathbf{x}, \boldsymbol{\xi}_j)\right)\Big|_{\mathbf{x}=\boldsymbol{\xi}_i},\ i,j = 1,\ldots,p, \tag{16}$$

we can obtain similar approximations to the y- and z-components of the surface gradient operator on S_1 as

$$\left(\mathcal{G}^y I_\phi f\right)\Big|_{S_1} = D^y A_{\mathrm{R}}^{-1} f\Big|_{S_1} =: G^y f\Big|_{S_1}, \tag{17}$$

$$\left(\mathcal{G}^z I_\phi f\right)\Big|_{S_1} = D^z A_{\mathrm{R}}^{-1} f\Big|_{S_1} =: G^z f\Big|_{S_1}. \tag{18}$$

To obtain an approximation to $\Delta_{\mathbb{M}}$ at the candidate stencil nodes S_1, we mimic the continuous formulation of the surface Laplacian in (3), replacing the continuous operators $\mathcal{G}^x$, $\mathcal{G}^y$, and $\mathcal{G}^z$ with the differentiation matrices G^x, G^y, and G^z, respectively. This gives the following differentiation matrix for approximating the surface Laplacian at the nodes S_1:

$$\begin{aligned} L_{\mathbb{M},1} &= G^x G^x + G^y G^y + G^z G^z \\ &= \underbrace{\left(D^x A_{\mathrm{R}}^{-1} D^x + D^y A_{\mathrm{R}}^{-1} D^y + D^z A_{\mathrm{R}}^{-1} D^z\right)}_{\hat{B}^T} A_{\mathrm{R}}^{-1}. \end{aligned} \tag{19}$$

When applying $L_{\mathbb{M},1}$ to a vector of samples of a target function f taken over S_1, this is equivalent to interpolating the target function with the regular RBF interpolant (8), computing the components of the surface gradient of the interpolant, then interpolating each of these components again using (8), applying the surface divergence, then evaluating this at the nodes in S_1. This is a type of iterated derivative approximation [24] and has the advantage of not needing the explicit formulas for the normal vectors (or their derivatives) to the surface $\mathbb{M}$.

Recall that the node sets X and $\tilde{X}$ are subsets of S_1 given by the index sets J and $\tilde{J}$, respectively (*cf.* (10)). Thus, to approximate the (i,ℓ) entry, $\Delta_{\mathbb{M},1}\phi(\tilde{\mathbf{x}}_i, \mathbf{x}_\ell)$, of B^T in (13), we can first apply $L_{\mathbb{M},1}$ to the vector of samples of $\phi(\mathbf{x}, \mathbf{x}_\ell)$ at S_1,

$$\begin{bmatrix}\phi(\boldsymbol{\xi}_1, \mathbf{x}_\ell) & \phi(\boldsymbol{\xi}_2, \mathbf{x}_\ell) & \cdots & \phi(\boldsymbol{\xi}_p, \mathbf{x}_\ell)\end{bmatrix}^T, \tag{20}$$

which gives a vector containing approximations to $\Delta_{\mathbb{M},1}\phi(\boldsymbol{\xi}_k, \mathbf{x}_\ell)$, $k = 1,\ldots,p$. The approximation to $\Delta_{\mathbb{M},1}\phi(\tilde{\mathbf{x}}_i, \mathbf{x}_\ell)$ is then given by the row in this vector corresponding to the i^{th} value in $\tilde{J}$ (which we denote by $\tilde{J}_i$). Note, however, that the vector (20) is just the J_ℓ column of A_{R} in (15), so that the approximation to $\Delta_{\mathbb{M},1}\phi(\tilde{\mathbf{x}}_i, \mathbf{x}_\ell)$ obtained from applying $L_{\mathbb{M},1}$ to (20) is just given by the $\tilde{J}_i$ and J_ℓ column of $\hat{B}^T$ in (19). Thus, all entries in B^T in (13) can be similarly obtained directly from the rows and columns or $\hat{B}^T$ using the index sets J and $\tilde{J}$. Additionally, the vector in the first block of the right hand side of (12) can be approximated from $\hat{B}^T$; in this case, from the first row of $\hat{B}^T$ and from the columns corresponding to J_i, $i = 1,\ldots,n$.

The second goal is to compute approximations to the entries of C in (12) and the entries of the second vector block in the right hand side of this equation. We give the entries of C explicitly to again elucidate the discussion:

$$C = \begin{bmatrix} \Delta_{\mathbb{M},1}\Delta_{\mathbb{M},2}\phi(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_1) & \cdots & \Delta_{\mathbb{M},1}\Delta_{\mathbb{M},2}\phi(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_m) \\ \vdots & \ddots & \vdots \\ \Delta_{\mathbb{M},1}\Delta_{\mathbb{M},2}\phi(\tilde{\mathbf{x}}_m, \tilde{\mathbf{x}}_1) & \cdots & \Delta_{\mathbb{M},1}\Delta_{\mathbb{M},2}\phi(\tilde{\mathbf{x}}_m, \tilde{\mathbf{x}}_m) \end{bmatrix}. \tag{21}$$

To approximate the operator $\Delta_{\mathbb{M},1}\Delta_{\mathbb{M},2}$ we again use iterated differentiation involving the differentiation matrices G^x, G^y, and G^z. Using the idea of (19), we can approximate $\Delta_{\mathbb{M},2}$ at the candidate stencil nodes S_1 using the differentiation matrix

$$L_{\mathbb{M},2} = \underbrace{\left((D^x)^T A_{\mathrm{R}}^{-1}(D^x)^T + (D^y)^T A_{\mathrm{R}}^{-1}(D^y)^T + (D^z)^T A_{\mathrm{R}}^{-1}(D^z)^T\right)}_{\hat{B}} A_{\mathrm{R}}^{-1}, \tag{22}$$

where D^x, D^y, and D^z are given by (14) and (16). We then approximate $\Delta_{\mathbb{M},1}\Delta_{\mathbb{M},2}$ at the nodes in S_1 as

$$L_{\mathbb{M},1}L_{\mathbb{M},2} = \underbrace{\left(\hat{B}^T A_{\mathrm{R}}^{-1}\hat{B}\right)}_{\hat{C}} A_{\mathrm{R}}^{-1}. \tag{23}$$

Using similar arguments as above for extracting approximations to the elements of B^T from $\hat{B}^T$, we can extract approximations to the elements of C from $\hat{C}$. For example, entry $C_{i,\ell}$ can be approximated by the entry in the $\tilde{J}_i$ row and $\tilde{J}_\ell$ column of $\hat{C}$. The elements in the second block vector of the right hand side of (12) can similarly be extracted from $\hat{C}$. In practice, $\hat{B}$ and $\hat{C}$ are formed by solving linear systems using the Cholesky factorization of A_{R}, instead of computing A_{R}^{-1}. Note that this ensures that $\hat{C}$ is symmetric so that the approximation to C will also be symmetric.

Upon obtaining approximations to the entries of B^T and C and the vector in the right hand of (12), we substitute these into the system (12) and solve it to obtain iterated RBF-HFD weights $\{w\}_{i=1}^n$ and $\{\tilde{w}\}_{j=1}^m$ to be used in (11).

For each node $\boldsymbol{\xi}_k \in \Xi$, $k = 1, \ldots, N$, we repeat the above procedure of finding its $p-1$ nearest neighbors (candidate stencil nodes S_k), selecting index sets for the explicit and implicit stencils, computing approximations to the p-by-p submatrices $\hat{B}^T$ and $\hat{C}$, and extracting the entries from these matrices to use for solving for the weights in (12). These weights are then arranged into two *sparse* N-by-N matrices L_Ξ and $\tilde{L}_\Xi$ for approximating the surface Laplacian over all the nodes in Ξ (see Section 5 for how L_Ξ and $\tilde{L}_\Xi$ are used for solving a PDE). Each row of L_Ξ has n non-zero entries and each row of $\tilde{L}_\Xi$ has m non-zero entries.

The computational cost of computing the weights for node $\boldsymbol{\xi}_k$ is $O(p^3)$, and there are N such stencils, so that the total cost of computing the entries of L_Ξ and $\tilde{L}_\Xi$ is $O(p^3 N)$. In our application of the RBF-HFD method, the dominant $O(p^3)$ cost for each $\boldsymbol{\xi}_k$ can also depend on m and n as we use a Greedy algorithm to select the index sets J and $\tilde{J}$ that give weights with desirable properties as discussed in Section 4.4. In practice, $p \ll N$ and would typically be fixed as N increases, so that the total cost scales like $O(N)$. Furthermore, the weights for one node can be computed independently from the others and is thus an embarrassingly parallel computation. In contrast, the method from [25], requires $O(N^3)$ operations and results in a dense differentiation matrix. However, the accuracy of this global method is better than the local RBF-HFD approach.

4.3. Choosing the candidate stencil nodes. Increasing the size p of the candidate stencil nodes in the iterated differentiation improves accuracy of the approximations to B^T and C described in the previous section, but also increases the computational cost and worsens the conditioning of the linear system in (19). Theoretically, the smallest possible candidate stencil would simply include every stencil node used in the RBF-HFD formula (11). However, this choice will not lead to accurate weights. Numerical experiments indicate that the candidate stencil should contain at least $n + m$ nodes in order to obtain stable and accurate weights.

In the flat basis limit, as the shape parameter goes to zero, RBF interpolants, in many cases, reproduce certain polynomial interpolants [10]. Wright and Fornberg [42] provided evidence that the weights obtained from Hermite RBF interpolants in this limit are identical to classical compact weights that are exact for polynomials. For the sphere, it is natural to suppose that the Hermite weights would become exact for the spherical harmonics. This is indeed the case, if the neighborhood size is sufficiently large, which is demonstrated in Fig. 1. Let the residual for an example stencil on the sphere be denoted

$$r = \sum_{i=1}^{n} w_i Y(\mathbf{x}_i) + \sum_{j=1}^{m} \tilde{w}_j (\Delta_{\mathbb{M}} Y)\big|_{\mathbf{x}=\tilde{\mathbf{x}}_j} - (\Delta_{\mathbb{M}} Y)\big|_{\mathbf{x}=\mathbf{x}_1}, \tag{24}$$

where Y is a spherical harmonic. Shown in the plot in Fig. 1 is the maximum absolute value of the residual, where the maximum is taken over the first $n+m$ spherical harmonics. For $p \ll n+m$, the weights incur errors of very large magnitude, but the error decreases rapidly as p increases. From this plot, we posit that p must consist of at least as many nodes as the number of spherical harmonics of one degree higher than the degree we wish the weights to be exact for. For instance, if we have $n+m=16$ and we wish the weights to be exact for all spherical harmonics up to third degree (of which there are 16), then p must at least be 25.

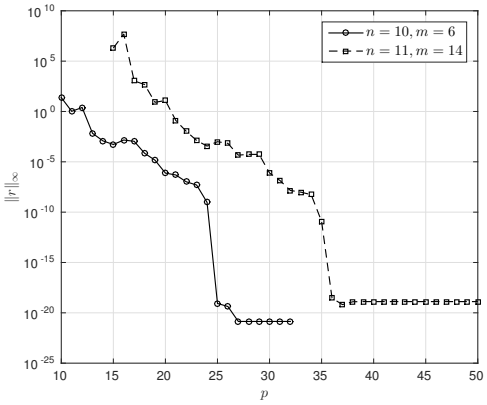


Fig. 1: The maximum absolute value of the residual for differentiating a spherical harmonic, with the maximum taken over the first $n+m$ spherical harmonics. In this figure, the weights are computed using variable precision arithmetic and $\varepsilon = 10^{-10}$. The levelling off of the residual at $\mathcal{O}(10^{-20})$ can be attributed to the choice of ε.

Whether the arguments above hold for other surfaces is uncertain, as this depends on the polynomial space spanned by the RBF basis in the limit as ε goes to zero. Additionally, it may not be of particular interest to explore the flat basis limit in practice, as such exploration requires multiple precision arithmetic or a stable method, such as RBF-GA [19, 29] or RBF-QR [17, 20], for computing the weights. The choice of p should rather be determined by accuracy and stability concerns.

4.4. Greedy Algorithm for Stencil Selection. If the node set is near-uniform, experiments have shown that a nearest neighbor approach to stencil selection is usually sound. However, compact stencils can provide additional properties if the stencil

nodes are chosen wisely. A simple greedy algorithm, similar to the one in [42], is used for this purpose. For small stencils ($n, m \leq 10$), that provide up to fourth order convergence, we enforce that all weights in $\tilde{L}_\Xi$ are positive, that $\tilde{L}_\Xi$ is diagonally dominant, and that all off-diagonal elements in L_Ξ are positive. By consistency, any row sum of L_Ξ is zero and thus the diagonal elements are negative. This property of L_Ξ, along with the diagonal dominance of $\tilde{L}_\Xi$, provides the stability properties outlined in Section 6, while imposing positivity of the $\tilde{L}_\Xi$-weights ensures that the compact weights mimic their lattice-based counterparts. For larger stencils, such weights cannot be found, and we must give up the last property.

The greedy algorithm proceeds in the following way:

1. For each node $\boldsymbol{\xi}_k$, determine the $p-1$ nearest neighbors to form S_k and compute all matrices necessary to form the approximation to the entries of A_H and the right hand side in the system (12) as described in Section 4.1.
2. Compute all combinations of choosing $n-1$ nodes from S_k and sort them by average distance to $\boldsymbol{\xi}_k$. Let $\{J^{(i)}\}_{i=1}^{i_{\max}}$ denote the set of index sets obtained.
3. Repeat step 2 with $n-1$ replaced by m and denote this set $\{\tilde{J}^{(j)}\}_{j=1}^{j_{\max}}$.
4. Let $i = j = 1$. Until stencils with suitable weights have been found, repeat the following two steps:
5. Compute w and $\tilde{w}$ from the approximation (12) using the stencils $J^{(i)}$ and $\tilde{J}^{(j)}$.
6. If the weights satisfy the conditions, or $i = i_{\max}$ and $j = j_{\max}$; go to step 1 and continue with $k = k+1$. Else if $j = j_{\max}$, or if $j = 1$ and w is not diagonally dominant; increase i and let $j = 1$. Else increase j.

The rationale behind the second condition in step 6 is that if w is not diagonally dominant, numerical experiments have shown that replacing the implicit stencil is unlikely to work. For near-uniform nodes and suitable values for the stencil and neighborhood sizes, the conditions are met for $i = j = 1$ for a majority of stencils, and the algorithm rarely requires more than 10 iterations in steps 5 and 6.

5. Using RBF-HFD weights with the method-of-lines. In Sections 7.1 and 7.2, we use the RBF-HFD discrete approximation to the surface Laplacian in the method-of-lines (MOL) to simulate diffusion and reaction-diffusion equations on surfaces. We briefly review this technique for the former equation, as its generalization to the latter follows naturally.

The diffusion of a scalar quantity u on a surface with a (non-linear) forcing term is given as

$$\frac{\partial u}{\partial t} = \delta \Delta_{\mathbb{M}} u + f(t, u), \tag{25}$$

where $\delta > 0$ is the diffusion coefficient, $f(t,u)$ is the forcing term, and an initial value of u at time $t = 0$ is given.

Our RBF-HFD method for (25) takes the form

$$\frac{d}{dt} u_\Xi = \delta \tilde{L}_\Xi^{-1} L_\Xi u_\Xi + f\left(t, u_\Xi\right), \tag{26}$$

where $\tilde{L}_\Xi^{-1} L_\Xi$ is an RBF-HFD discretization of $\Delta_{\mathbb{M}}$ over the nodes in Ξ, as described above. This is a system of N coupled ODEs and, provided it is stable (see Section 6), can be advanced in time with a suitably chosen time-integration method. In contrast to an explicit RBF-FD discretization, where $\tilde{L}_\Xi$ is the identity matrix, both explicit and implicit time discretizations will require solving a sparse linear system. The

diffusion term is typically treated implicitly in order to allow larger time steps, and we have chosen to use a semi-implicit BDF3 method [2], given by

$$\left(\frac{11}{6}I - \delta\Delta t\tilde{L}_\Xi^{-1}L_\Xi\right)u_X^{n+1} = 3u_X^n - \frac{3}{2}u_X^{n-1} + \frac{1}{3}u_X^{n-2} + \Delta t\left(3f(t^n,u_X^n) - 3f(t^{n-1},u_X^{n-1}) + f(t^{n-2},u_X^{n-2})\right), \tag{27}$$

where the superscript denotes time level. If δ and Δt are constant in time, we may multiply by $\tilde{L}_\Xi$ to obtain the system

$$\underbrace{\left(\frac{11}{6}\tilde{L}_\Xi - \delta\Delta tL_\Xi\right)}_{R_\Xi}u_X^{n+1} = \tilde{L}_\Xi\{\text{r.h.s. of (27)}\} \tag{28}$$

The matrix R_Ξ is sparse and well-conditioned, and may be factorized if a sufficient amount of memory is available. Another option is to use a Krylov solver with a suitable pre-conditioner, for instance an ILU decomposition. The latter might be preferable if the time step is adaptive, since the zero-fill ILU pre-conditioner is cheap to compute, at least compared to a full LU factorization. The performance of this approach is discussed in Section 7.

6. Stability and Gershgorin Sets. One important reason to favor high-order compact stencils over their explicit counterparts is eigenvalue stability. In the classical setting, where stencils are symmetric, the structure of the obtained generalized eigenvalue problem ensures stability when using any A-stable time stepping scheme. Consider Equation (25) with $f \equiv 0$ and the corresponding compact semi-discretization

$$\frac{d}{dt}u_\Xi = \tilde{L}_\Xi^{-1}L_\Xi u_\Xi. \tag{29}$$

We wish to prove that any eigenvalue of $\tilde{L}_\Xi^{-1}L_\Xi$ lies in the left half-plane, which is a necessary condition for stability. In the following, we will consider the equivalent problem of showing that all eigenvalues of the generalized eigenvalue system

$$A\boldsymbol{x} = \lambda B\boldsymbol{x}, \tag{30}$$

where $A = -L_\Xi$ and $B = \tilde{L}_\Xi$, lie in the right half-plane.

A common way to prove stability is to use the Gershgorin circle theorem. For instance, if A has zero row sum, positive diagonal and non-positive off-diagonal elements, then any eigenvalue of A must have a non-negative real part. We will assume that A has these properties, and that B is a strictly diagonally dominant matrix with positive diagonal elements. If A and B were Hermitian, these properties would be sufficient for the eigenvalues of the generalized eigenvalue problem to have non-negative real parts. The situation is somewhat more complicated in the non-symmetric case.

Stewart [40] extended the Gershgorin circle theorem to generalized eigenvalues, and proved that any eigenvalue must lie in $\bigcup_i \Gamma_i$, where Γ_i is given by $z \in \mathbb{C}$ such that

$$|zb_{ii} - a_{ii}| \leq \sum_{j\neq i}|zb_{ij} - a_{ij}|, \tag{31}$$

where a_{ij} and b_{ij} denote the elements of A and B, respectively. In contrast to the regular Gershgorin theorem, it is quite difficult to determine the values of z that

fulfill this inequality. By cleverly applying the the triangle inequality, Kostić and co-workers [27] provided an approximate Gershgorin set that can be easily computed. Let $r_i(A)$ denote the absolute sum of the ith row of A with the diagonal element zeroed out. The ith approximate Gershgorin set $\hat{\Gamma}_i$ is given by $z \in \mathbb{C}$ such that

$$|zb_{ii} - a_{ii}| \leq |z| r_i(B) + r_i(A).$$

By dividing by b_{ii}, which is positive by assumption, we obtain

$$|z - \frac{a_{ii}}{b_{ii}}| \leq |z| \frac{r_i(B)}{b_{ii}} + \frac{r_i(A)}{b_{ii}}. \tag{32}$$

We will let $\alpha = \frac{a_{ii}}{b_{ii}}$ and $\beta = \frac{r_i(B)}{b_{ii}}$, and note that we have $r_i(A) = a_{ii}$ from the matrix properties we assumed. Note also that $\beta < 1$ since B is strictly diagonally dominant.

Approximate Gershgorin sets for $\alpha = 1$ and various β are shown in Figure 2. In particular, note that none of the sets contain any part of the negative real axis. For small values of β, the only part of the negative real half-plane that is included in the approximate Gershgorin set is a narrow segment along the imaginary axis. In the special case where B has non-negative elements, it is typically possible to find stencils that provide $\beta < 0.5$, which makes the unstable part of the Gershgorin set practically insignificant.

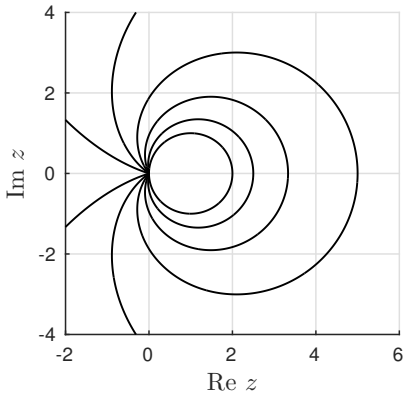
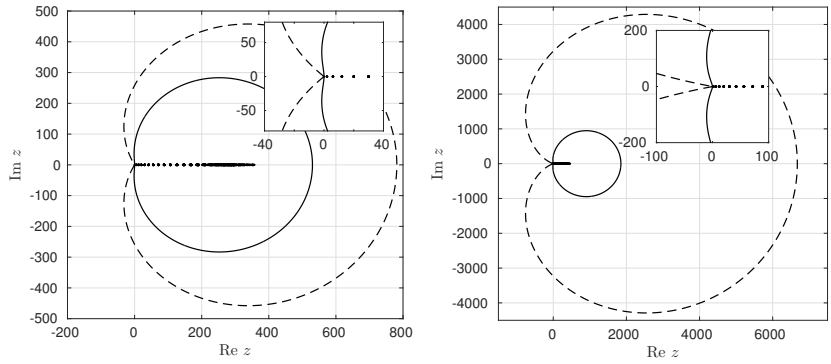


Fig. 2: Approximate Gershgorin sets for $\alpha = 1$ and various values of β. The circle corresponds to $\beta = 0$, and additional curves are given by $\beta = 0.2$, 0.4, 0.6 and 0.8, starting from the innermost to the outermost curve.

In practice, the exact Gershgorin sets turn out to be much smaller than the approximate ones for the discretizations considered here. Examples are shown in Figure 3, where fourth order and sixth order approximations of the Laplace–Beltrami operator on the sphere are considered. Note that the regions shown are not the (approximate) Gershgorin set, but rather the ith (approximate) Gershgorin set, where i is chosen as the row that gives the largest extent in the left half-plane. For the fourth order method, the Gershgorin set shown is essentially completely contained in the right half-plane.

7. Numerical Results. The numerical experiments in this section were performed in MATLAB, using node sets generated by DistMesh [33, 34], unless otherwise



(a) Fourth order method. (b) Sixth order method.

Fig. 3: Generalized eigenvalues of (A, B), and corresponding Gershgorin sets. The ith approximate Gershgorin set is outlined by a dashed line, and the respective exact one is given by a solid line, where the index i is chosen to give the largest extent in the left half-plane of each individual set. The inset shows a magnification about the origin.

Table 1: The manifolds in the experiments are given by $\{x, y, z\}$ satisfying $F(x, y, z) = 0$. For the Red Blood Cell, the parameters are $c_0 = \frac{0.81}{a}$, $c_2 = \frac{7.83}{a}$, $c_4 = \frac{-4.39}{a}$ and $r_0 = \frac{3.91}{a}$ with $a = 3.39$. The parameters for Dupin's Cyclide are $c_1 = 2$, $c_2 = 1.9$, $c_3 = \sqrt{0.39}$ and $c_4 = 1$.

Surface	$F(x, y, z)$
Sphere	$x^2 + y^2 + z^2 - 1$
Torus	$(1 - \sqrt{x^2 + y^2})^2 + z^2 - \frac{1}{9}$
Red Blood Cell	$\left(1 - \frac{x^2+y^2}{r_0^2}\right)\left(c_0 + c_2\left(\frac{x^2+y^2}{r_0^2}\right) + c_4\left(\frac{x^2+y^2}{r_0^2}\right)^2\right)^2 - 4z^2$
Dupin's Cyclide	$(x^2 + y^2 + z^2 - c_4^2 + c_2^2)^2 - 4(c_1 x + c_3 c_4)^2 - 4c_2^2 y^2$
"Tooth"	$x^8 + y^8 + z^8 - (x^2 + y^2 + z^2)$

noted. For a mathematical description of the manifolds, see Table 1. Example node sets are presented in Fig. 4.

7.1. Parameter Studies. To verify the convergence rate and facilitate comparisons between explicit and implicit finite difference methods, we use the forced heat equation with a known analytic solution. We restrict our attention to the surface of the sphere and the torus, in order to be able to manufacture exact solutions.

As in [37], we take the exact solution for the sphere to be

$$u(t, \mathbf{x}) = e^{-5t} \sum_{k=1}^{23} e^{-10\cos^{-1}(\boldsymbol{y}_k \cdot \mathbf{x})}, \tag{33}$$

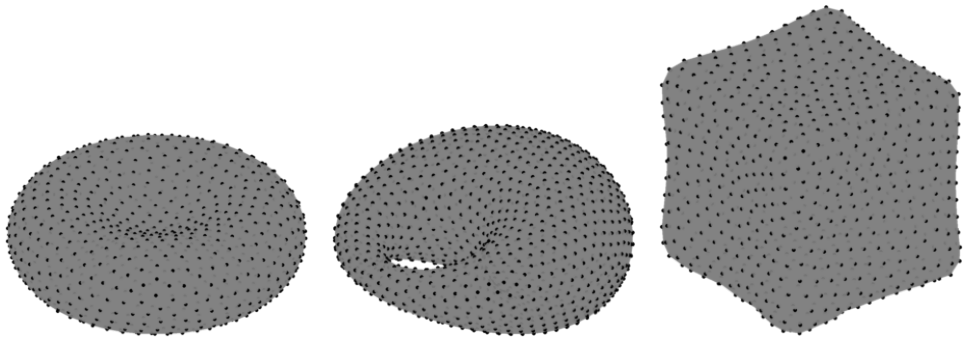


Fig. 4: Example node sets for the Red Blood Cell model, Dupin's Cyclide, and the "Tooth" model.

where $\mathbf{x} \in \mathbb{S}^2$ and $\boldsymbol{y}_k$, $k = 1, \ldots, 23$, randomly placed points on $\mathbb{S}^2$. For the torus, we also use the exact solution from [37]:

$$u(t, \lambda, \varphi) = e^{-5t} \sum_{k=1}^{30} e^{-20(1-\cos(\lambda-\lambda_k))-\frac{9}{4}(1-\cos(\varphi-\varphi_k))}. \tag{34}$$

Here the solution is stated in the parametric variables $(\lambda, \varphi) \in [-\pi, \pi]^2$ for the torus, and (λ_k, φ_k), $k = 1, \ldots, 30$, are taken as randomly chosen values in $[-\pi, \pi]^2$. The exact parameterization of the torus we use is

$$x = \left(1 + \frac{1}{3}\cos(\varphi)\right)\cos(\lambda),\ y = \left(1 + \frac{1}{3}\cos(\varphi)\right)\sin(\lambda),\ z = \frac{1}{3}\sin(\varphi).$$

The forcing terms for the diffusion equations on the two surfaces are computed from these exact solutions. Unless otherwise noted, the diffusion equations for both surfaces are simulated for $0 \leq t \leq 0.2$ and the time step is chosen such that spatial errors dominate. All time integrations are done using BDF3 and the forcing function is evaluated implicitly. We restrict the presentation to the relative ℓ_∞-norm of the error as the observed convergence rates were the same for the ℓ_1-, ℓ_2-, and ℓ_∞-norms. Finally, we let h denote the 'spacing' of the nodes in Ξ, and compute this as the average distance to the nearest neighbor.

Shape parameter. Two strategies for the scaling of the shape parameter are commonly used: inversely proportional to the node distance h, or fixed. The former choice keeps the condition number of the regular interpolation matrix, here denoted $\kappa(A_R)$, constant, but introduces a stationary interpolation error that does not decrease to zero in the limit as h goes to zero. A fixed ε gives convergence for all h, but the linear systems for computing the weights become ill-conditioned for small h, and convergence is lost due to round-off errors. There are workarounds for both of these problems. In the stationary interpolation case, it is possible to recover low order convergence by adding suitable polynomial terms to the interpolant [13]. On a surface, however, the polynomials may themselves introduce ill-conditioning.

If ε is kept fixed, there are currently two options to circumvent the problem of ill-conditioning: stable algorithms or variable-precision arithmetic. The algorithms of the former category, such as RBF-GA, RBF-QR, and RBF-CP [17, 19, 21], are unfortunately not easily adaptable to Hermite interpolation in the form introduced

here. Instead, we adopt quad-precision arithmetic, for instance using the Advanpix toolbox [1], which allows accurate determination of the weights for values of h corresponding to millions of nodes on the surfaces considered. Note that quad-precision is only needed for the computation of the weights, after which the results can be truncated back to double precision and the simulations fun without issues. Also note that computing the weights is an embarrassingly parallel task, and one that is only performed once for a given simulation. Optimization of ε is beyond the scope of this study, and the value is chosen such that weights can be computed in double precision for small to moderate node sets ($N \lesssim 50,000$), after which we switch to quad-precision.

Results for the forced heat equation of the sphere illustrating the effects of the two shape parameter strategies are presented in Figures 5a and 5b, in which the errors as a function of h are shown for a fourth order and a sixth order approximation, respectively. In the fourth order case, both strategies for selecting the shape parameter converge with the same rate for a large range of values of h, but keeping the condition number fixed eventually results in a stationary error as h decreases. For the sixth order stencil, the resulting order is slightly lower in the fixed condition number setting, and convergence is achieved only in a small range of values of h. Similar results for the torus were observed and have thus been omitted.

Stencil sizes. The choice $m = n-1$ and letting the index sets I and J coincide (bar the evaluation node) provides ideal sparsity of the matrix R_Ξ in Eq. (28). However, as the approximation order is increased, it gradually becomes more difficult to find weights w that satisfy the diagonal dominance criterion. On near-uniform nodes, it is typically possible to find stable weights for n up to 12. The stencil selection algorithm will also influence the sparsity of the matrices, and certain combinations of n and m are more likely to produce stable weights (e.g., by symmetry). Another consideration is the formal approximation order of the stencil, which we are not able to derive, and so instead determine it experimentally. The order appears to be primarily determined by the number of degrees of freedom of the Hermite system, i.e., $n + m$.

Figures 6a and 6b show the error as a function of h for the forced heat equation on the sphere and on the torus, respectively, with different values of m and n. In these figures, the shape parameter is kept fixed at $\varepsilon = 3$ for the smaller stencil sizes and $\varepsilon = 5$ for the larger ones. With these choices, quad precision arithmetic is only required for computing the weights for the two largest node sets, which range in size from $N \approx 1000$ to $N \approx 200,000$. Inevitable floating point cancellation errors from the finite difference scheme appear to limit the accuracy at around 10^{-8}. We summarize the observations from this experiment regarding observed order of convergence and stencil sizes in Table 2.

Performance comparison. In addition to the stability properties discussed in Section 6, compact stencils provide better sparsity for the same approximation order. This should lead to a smaller memory footprint and fewer floating-point operations for solving the system in Eq. (28). In this paper, we use BiCGSTAB with zero-fill ILU-preconditioner for solving the linear system. Table 3 shows the number of non-zero elements in R_Ξ and the CPU time for a simulation for some combinations of N, n and m. Also presented in this table is the average number of Krylov iterations per time step. It is interesting to note from this table that in terms of CPU time, the smaller implicit stencil barely outperforms the explicit stencil of the same order. This can be attributed to the larger number of iterations needed for the Krylov solver to converge. Increasing the stencil size to $n = 11$ and $m = 15$ reduces the number of iterations needed, plausibly due to the larger number of non-zeros in the incomplete

Table 2: Recommended stencil and neighborhood sizes for different approximation orders. The order was determined from numerical experiments on the sphere and the torus shown in Figure 6.

n	m	p	Observed order
10	6	19	4
11	15	32	6
17	0	17	4
31	0	31	6

LU-factorization. This results in a CPU time that is comparable to that of the smaller implicit stencil. In this comparison, the fourth order explicit stencil ($n = 17$, $m = 0$) is on par with the sixth order implicit stencil both in terms of memory requirements and computational cost. Note that the time step chosen, $\Delta t = 10^{-3}$, is rather small (which is need for spatial errors to dominate). Increasing the time step to $\Delta t = 0.1$ increases the number of Krylov iterations per time step roughly by a factor of 6.

Table 3: The table shows the number of non-zeros of the matrix R_Ξ, the average iteration count for the Krylov solver, and the CPU time for 200 time steps with $\Delta t = 10^{-3}$ for some choices of n, m and h. The surface in this experiment is the sphere, and the tolerance for the Krylov solver is 10^{-10}.

n	m	h	N	# of non-zeros	Avg. iter.	CPU time (s)
10	6	0.1	1806	18060	1.5	0.50
		0.05	7446	74462	2.5	1.6
		0.025	30054	300550	4.5	9.6
11	15	0.1	1806	28896	1.5	0.53
		0.05	7446	119136	2	1.7
		0.025	30054	480866	3.5	9.8
17	0	0.1	1806	30702	2	0.60
		0.05	7446	126582	2.5	1.9
		0.025	30054	510918	3.5	9.5
31	0	0.1	1806	55986	1.5	0.59
		0.05	7446	230826	2	2.7
		0.025	30054	931674	3.5	16.8

In most applications, Δt would be chosen proportional to h in order to reduce both spatial and temporal errors as the node set is refined. Figure 7 shows the CPU time as a function of h with $\Delta t = 10^{-2} \cdot h$. For the near-uniform node sets used throughout this paper, $N \propto 1/\sqrt{h}$, and thus the CPU time scales as $\mathcal{O}(N^{3/2})$.

7.2. Applications. We now present applications of the new compact scheme to solving reaction-diffusion equations on different surfaces. As in [37], we present results of simulations both on surfaces defined implicitly by algebraic expressions (see Table 1) and on more general point cloud surfaces. On the former, we simulate the same

two-species Turing system used in [37], given by:

$$\frac{\partial u}{\partial t} = \alpha u(1-\tau_1 v^2) + v(1-\tau_2 u) + \delta_u \Delta_{\mathbb{M}} u, \tag{35}$$

$$\frac{\partial v}{\partial t} = \beta v\left(1 + \frac{\alpha\tau_1}{\beta} uv\right) + u(\gamma + \tau_2 v) + \delta_v \Delta_{\mathbb{M}} v. \tag{36}$$

A visualization of the solutions of this equation on the various surfaces with the parameters selected from Table 4 are shown in Fig. 8. On the more general point

Table 4: The table shows the values of the parameters of Equations (35) and (36) used in the numerical experiments shown in Figures 8. We set $\delta_u = 0.516\delta_v$ for the Red Blood Cell and Tooth models, and use $\delta_u = 5.16\delta_v$ for Dupin's Cyclide.

Pattern	δ_v	α	β	γ	τ_1	τ_2	Final time
Spots	4.5×10^{-3}	0.899	-0.91	-0.899	0.02	0.2	200
Stripes	2.1×10^{-3}	0.899	-0.91	-0.899	3.5	0	4000

cloud surfaces, we simulate the Fitzhugh–Nagumo-type model used in [25]:

$$\frac{\partial u}{\partial t} = \delta_u \Delta_{\mathbb{M}} u + \frac{1}{\alpha} u(1-u)\left(u - \frac{v+b}{a}\right), \tag{37}$$

$$\frac{\partial v}{\partial t} = \delta_v \Delta_{\mathbb{M}} v + u - v. \tag{38}$$

For both the Bumpy Sphere[2] and Bunny[3] models, we set $a = 0.75$, $b = \alpha = 0.02$, $\delta_v = 0$, and $\delta_u = 1.5(\frac{2\pi}{50})^2$. With these parameters, the model generates dynamic spiral wave solutions. Snapshots of these solutions computed with the RBF-HFD method for the two surfaces are shown in Fig. 9. We note that the normal vectors on these point cloud models can be generated by any appropriate method. In this work, the node sets and normal vectors were created using MeshLab [7], utilizing the Poisson surface reconstruction algorithm with some additional smoothing and the Poisson disk sampling method.

7.3. Curvature, node spacing, and the stencil selection algorithm. In some experiments, the greedy algorithm failed to find suitable stencils for some node points. A common characteristic for these nodes were that they were located in areas of large curvature, e.g., around the ears of the Bunny. Using nearest neighbor stencils for these cases did not cause any instabilities, however. Two simple remedies for this issue are increasing ε, and refining the node set. The former has previously been noted to improve stability (see, e.g., [15]). To illustrate the effect of curvature on the failure of the stencil selection algorithm we carried out an experiment generating the surface Laplacian on the a prolate spheroid of varying curvatures and determining the number of rejected stencils. The results are plotted in Fig. 10 in terms of the number of rejected stencils, i.e., stencils where the stability constraints could not be met, as a function of both the node distance h and the maximum mean curvature H. Note

[2]Available from the Aim@Shape Shape Repository (http://visionair.ge.imati.cnr.it/).

[3]Available from the Stanford Computer Graphics Laboratory (http://graphics.stanford.edu/data/3Dscanrep/).

that $i_{\max}$ and $j_{\max}$ in the greedy algorithm were set to five in this experiment so that only a small number of stencils were attempted for each node, to emphasize the effect of the curvature.

8. Discussion. The compact RBF-HFD scheme improves on previous RBF-FD schemes for diffusion on surfaces both in terms of efficiency and stability. The proposed greedy stencil selection algorithm ensures eigenvalue stability on surfaces without large (or rapid) changes in curvature. In numerical experiments of the forced diffusion equation on the sphere and the torus, the new scheme provided accuracy similar to the previous non-compact RBF-FD method, but with a smaller memory footprint and higher accuracy. The linear systems generated from the semi-implicit BDF3 time discretization were also shown to be efficiently solvable using BiCGStab with a standard zero-fill ILU-preconditioner. In addition to illustrating the having good stability, accuracy, and efficiency properties of the scheme, we showed how it can be easily adaptable to reaction-diffusion equations on both implicitly defined surfaces, and surfaces defined by a point cloud. The method can be naturally generalized to a smooth orientable surface that is discretized with a set of roughly uniform nodes and with approximations to the normal to the surface at each of the nodes.

While stencil sizes generating fourth and sixth order convergence in numerical experiments on the sphere and the torus are provided, no investigation of the theoretical convergence rates have been given. This is clearly an avenue of future investigation. Another future topic of research is the influence of the curvature and the shape parameter on the computed RBF-HFD weights. Experiments suggest that large local curvature makes it impossible to find weights that satisfy the conditions that ensure eigenvalue stability, although no issues with temporal integration were encountered when this stability was not insured. Refining the node set appears to provide an easy way to alleviate the problem, and an extensive investigation of the relationship between nodal distance and curvature would be of value.

Finally, we note that an extension to convection-diffusion problems would allow the method to be used in various applications, *e.g.* chemical transport on thin membranes and shells, biomechanical modeling of cells. This is currently being pursued by the first author.

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

REFERENCES

[1] *Advanpix Multiprecision Computing Toolbox for MATLAB*. http://www.advanpix.com/. Accessed: 2016-06-09.
[2] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton, *Implicit-explicit methods for time-dependent PDEs*, SIAM J. Numer. Anal, 32 (1997), pp. 797–823.
[3] V. Bayona, M. Moscoso, M. Carretero, and M. Kindelan, *RBF-FD formulas and convergence properties*, J. Comput. Phys., 229 (2010), pp. 8281–8295.
[4] R. L. Bishop and S. I. Goldberg, *Tensor Analysis on Manifolds*, Macmillan, New York, 1968.
[5] T. Cecil, J. Qian, and S. Osher, *Numerical methods for high dimensional Hamilton-Jacobi equations using radial basis functions*, J. Comput. Phys., 196 (2004), pp. 327–347.
[6] G. Chandhini and Y. Sanyasiraju, *Local RBF-FD solutions for steady convection–diffusion problems*, Int. J. Numer. Meth., 72 (2007), pp. 352–378.
[7] P. Cignoni, M. Corsini, and G. Ranzuglia, *Meshlab: an open-source 3D mesh processing system*, ERCIM News, (2008), pp. 45–46.
[8] L. Collatz, *The numerical treatment of differential equations*, Berlin: Springer, 1966, 3rd ed., (1966).

[9] O. Davydov and D. T. Oanh, *Adaptive meshless centres and RBF stencils for Poisson equation*, J. Comput. Phys., 230 (2011), pp. 287–304.
[10] T. A. Driscoll and B. Fornberg, *Interpolation in the limit of increasingly flat radial basis functions*, Comput. Math. Appl., 43 (2002), pp. 413–422.
[11] G. E. Fasshauer, *Meshfree Approximation Methods with MATLAB*, Interdisciplinary Mathematical Sciences - Vol. 6, World Scientific Publishers, Singapore, 2007.
[12] G. E. Fasshauer and M. J. McCourt, *Stable evaluation of Gaussian radial basis function interpolants*, SIAM J. Sci. Comput., 34 (2012), pp. A737–A762.
[13] N. Flyer, B. Fornberg, V. Bayona, and G. A. Barnett, *On the role of polynomials in RBF-FD approximations I. Interpolation and accuracy*, J. Comput. Phys., 321 (2016), pp. 21–38.
[14] N. Flyer, E. Lehto, S. Blaise, G. B. Wright, and A. St-Cyr, *A guide to RBF-generated finite differences for nonlinear transport: shallow water simulations on a sphere*, J. Comput. Phys., 231 (2012), pp. 4078–4095.
[15] N. Flyer and G. B. Wright, *A radial basis function method for the shallow water equations on a sphere*, Proc. Roy. Soc. A, 465 (2009), pp. 1949–1976.
[16] B. Fornberg and N. Flyer, *A Primer on Radial Basis Functions with Applications to the Geosciences*, SIAM, Philadelphia, 2014.
[17] B. Fornberg, E. Larsson, and N. Flyer, *Stable computations with Gaussian radial basis functions*, SIAM J. Sci. Comput., 33(2) (2011), pp. 869–892.
[18] B. Fornberg and E. Lehto, *Stabilization of RBF-generated finite difference methods for convective PDEs*, J. Comput. Phys., 230 (2011), pp. 2270–2285.
[19] B. Fornberg, E. Lehto, and C. Powell, *Stable calculation of Gaussian-based RBF-FD stencils*, Comp. Math. Applic., 65 (2013), pp. 627–637.
[20] B. Fornberg and C. Piret, *A stable algorithm for flat radial basis functions on a sphere*, SIAM J. Sci. Comput., 30 (2007), pp. 60–80.
[21] B. Fornberg and G. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter*, Comput. Math. Appl., 48 (2004), pp. 853–867.
[22] B. Fornberg and J. Zuev, *The Runge phenomenon and spatially variable shape parameters in RBF interpolation*, Comput. Math. Appl., 54 (2007), pp. 379–398.
[23] E. Fuselier and G. Wright, *Scattered data interpolation on embedded submanifolds with restricted positive definite kernels: Sobolev error estimates*, SIAM J. Numer. Anal., 50 (2012), pp. 1753–1776.
[24] E. Fuselier and G. B. Wright, *Order-preserving derivative approximation with periodic radial basis functions*, SIAM J. Numer. Anal., 41 (2015), pp. 23–53.
[25] E. J. Fuselier and G. B. Wright, *A high-order kernel method for diffusion and reaction-diffusion equations on surfaces*, J. Sci. Comput., (2013), pp. 1–31.
[26] Q. T. L. Gia, *Approximation of parabolic pdes on spheres using spherical basis functions*, Adv. Comput. Math., 22 (2005), pp. 377–397.
[27] V. Kostić, R. S. Varga, and L. Cvetković, *Localization of generalized eigenvalues by Cartesian ovals*, Numer. Linear Algebra Appl., 19 (2012), pp. 728–741.
[28] E. Larsson and B. Fornberg, *Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions*, Comput. Math. Appl., 49 (2005), pp. 103–130.
[29] E. Larsson, E. Lehto, A. Heryudono, and B. Fornberg, *Stable computation of differentiation matrices and scattered node stencils based on Gaussian radial basis functions*, SIAM J. Sci. Comput., 35 (2013), pp. A2096–A2119.
[30] S. K. Lele, *Compact finite difference schemes with spectral-like resolution*, J. Comput. Phys., 103 (1992), pp. 16–42.
[31] C. Macdonald and S. Ruuth, *The implicit closest point method for the numerical solution of partial differential equations on surfaces*, SIAM J. Sci. Comput., 31 (2010), pp. 4330–4350.
[32] F. Narcowich and J. Ward, *Generalized hermite interpolation via matrix-valued conditionally positive definite functions*, Math. Comp., 63 (1994), pp. 661–688.
[33] P.-O. Persson, *DistMesh – a simple mesh generator in MATLAB*. http://persson.berkeley.edu/distmesh/. Accessed: 2016-06-09.
[34] P.-O. Persson and G. Strang, *A simple mesh generator in MATLAB*, SIAM Review, 46 (2004), pp. 329–345.
[35] C. Piret, *The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces*, J. Comput. Phys., 231 (2012), pp. 4662–4675.
[36] R. Schaback, *Multivariate interpolation by polynomials and radial basis functions*, Constr. Approx., 21 (2005), pp. 293–317.
[37] V. Shankar, G. B. Wright, R. M. Kirby, and A. L. Fogelson, *A radial basis function*

(RBF)-finite difference (FD) method for diffusion and reaction–diffusion equations on surfaces, J. Sci. Comput., 63 (2014), pp. 745–768.

[38] C. Shu, H. Ding, and K. S. Yeo, *Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier–Stokes equations*, Comput. Methods Appl. Mech. Eng., 192 (2003), pp. 941–954.

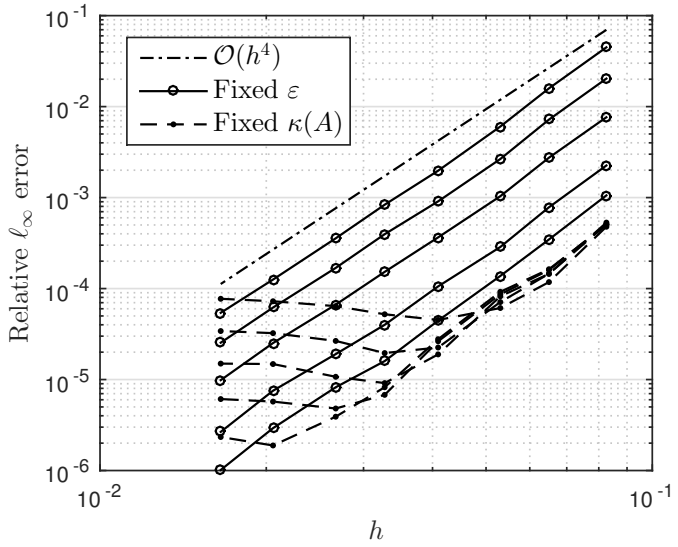
[39] D. Stevens, H. Power, M. Lees, and H. Morvan, *The use of PDE centers in the local RBF Hermitean method for 3D convective-diffusion problems*, J. Comput. Phys., 228 (2009), pp. 4606–4624.

[40] G. W. Stewart, *Gershgorin theory for the generalized eigenvalue problem* $Ax = \lambda Bx$, Math. Comp., 29 (1975), pp. 600–606.

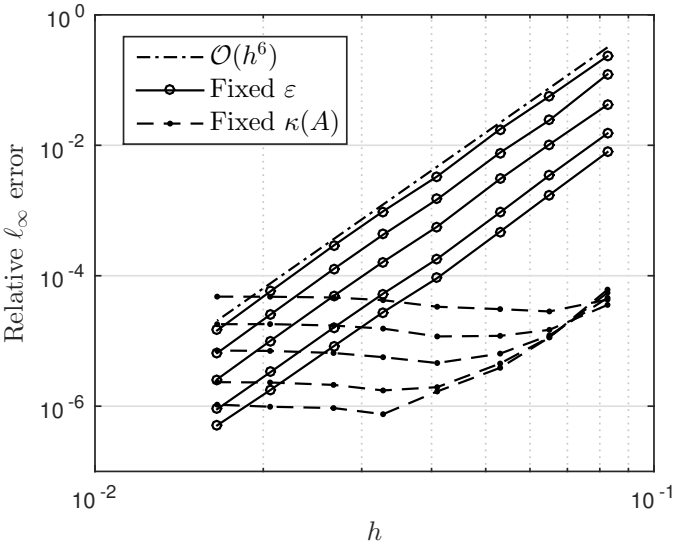
[41] G. B. Wright, N. Flyer, and D. A. Yuen, *A hybrid radial basis function–pseudospectral method for thermal convection in a 3-D spherical shell*, Geochem. Geophys. Geosyst., 11 (2010).

[42] G. B. Wright and B. Fornberg, *Scattered node compact finite difference-type formulas generated from radial basis functions*, J. Comput. Phys., 212 (2006), pp. 99 – 123.

[43] W. Zongmin, *Hermite–Birkhoff interpolation of scattered data by radial basis functions*, Approx. Theory Appl., 8 (1992), pp. 1–10.

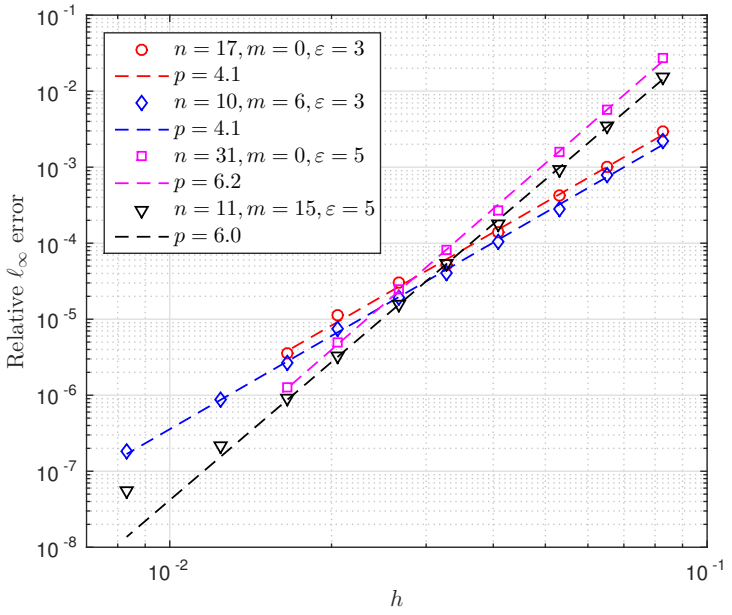


(a) $n = 10$, $m = 6$, $p = 19$.

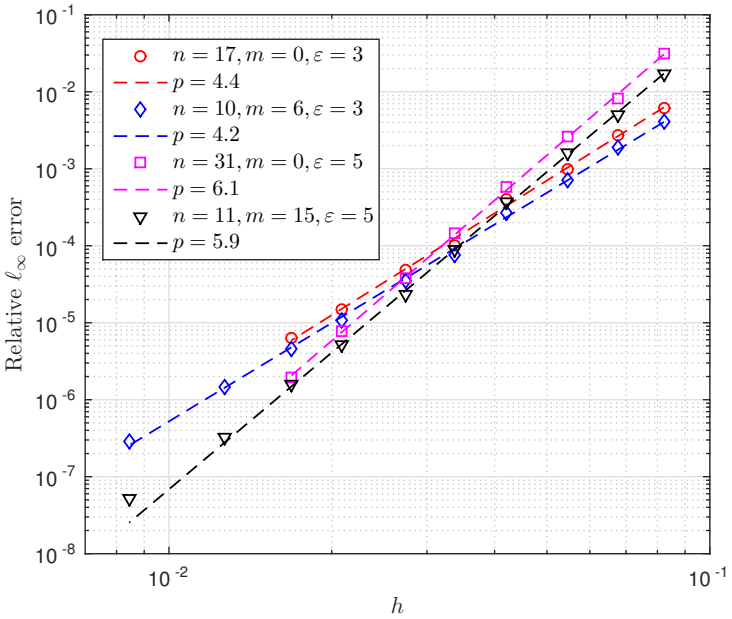


(b) $n = 11$, $m = 15$, $p = 32$.

Fig. 5: The error as a function of h for the forced heat equation on the sphere for different strategies of shape parameter selection. Solid lines correspond to fixed shape parameter, and dashed lines correspond to fixed condition number. The dash-dot line show slopes for convergence rate $\mathcal{O}(h^p)$ with $p = 4, 6$. In (a), the values of ε are $\{6, 5, 4, 3, 2.5\}$ from top to bottom and in (b), the values are $\{8, 7, 6, 5, 4.5\}$, again from top to bottom. The values of $\kappa(A_R)$ are $\{10^{10}, 10^{11}, 10^{12}, 10^{13}, 10^{14}, 10^{15}\}$ from top to bottom (at small h) in both (a) and (b).



(a) Sphere.



(b) Torus.

Fig. 6: The error as a function of h for the forced heat equation with different stencil sizes. The dashed lines show slopes for convergence rate $\mathcal{O}(h^p)$, fitted from the data points. For $n = 11$ and $m = 15$, the last point was excluded from the fit as floating point round off errors limit the accuracy.

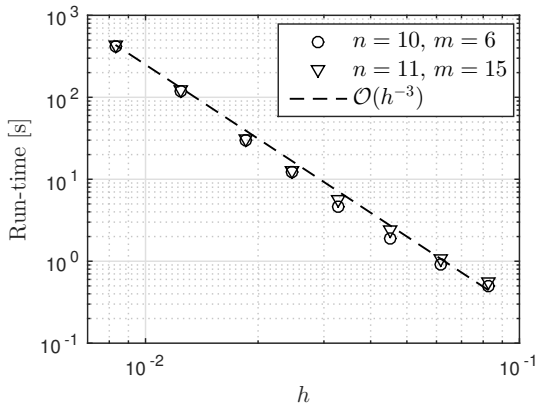


Fig. 7: The run-time for a simulation of the forced heat equation on the sphere as a function of h, using $\Delta t = 10^{-2} \cdot h$.

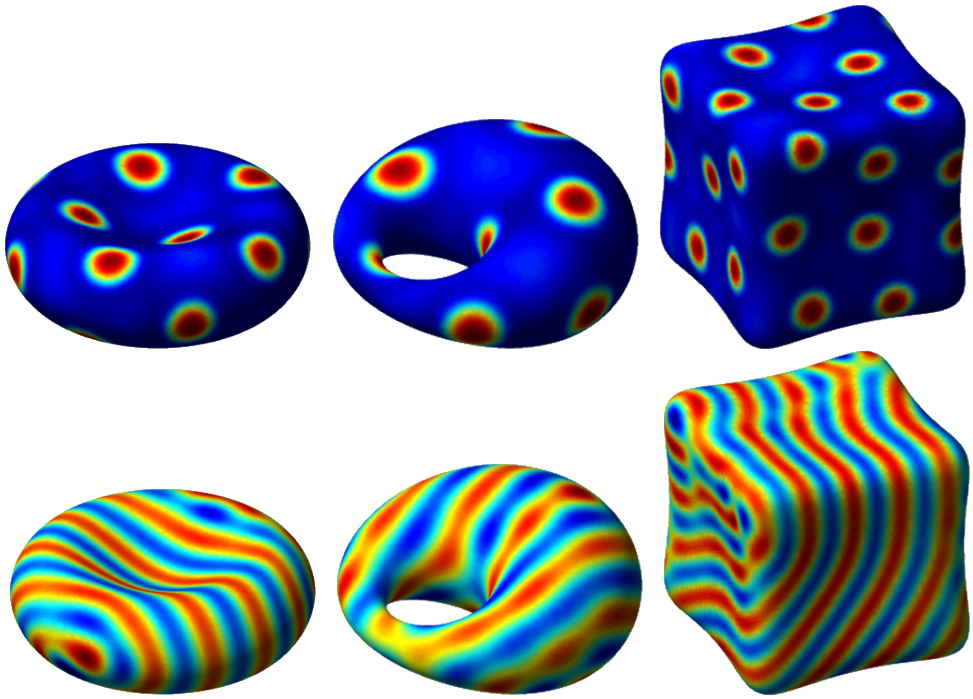


Fig. 8: Quasi steady turing spots and stripe patterns resulting from solving Equations (35) and (36) on the Red Blood Cell model, Dupin's Cyclide, and the "Tooth" model. In all plots, red corresponds to a high concentration of u and blue to a low concentration.

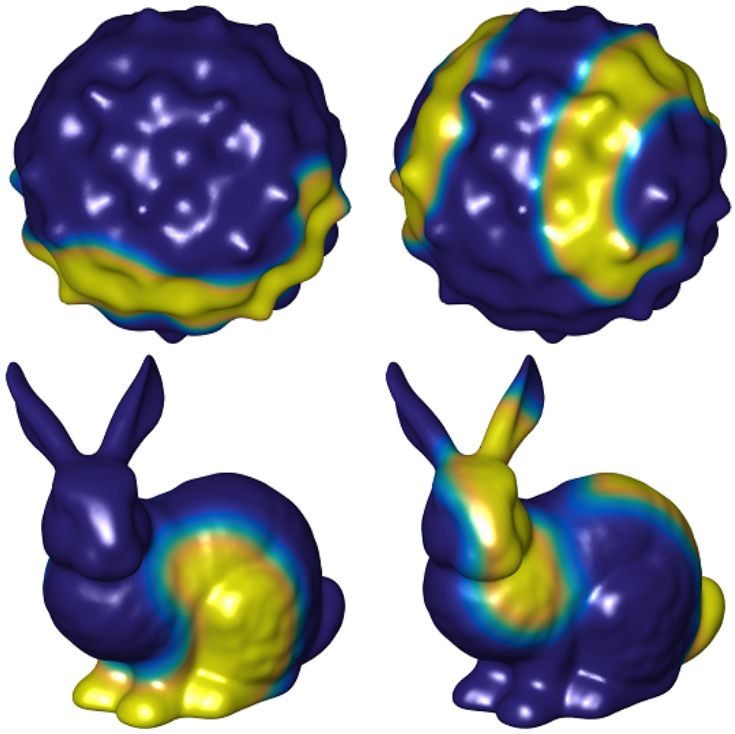


Fig. 9: Fitzhugh–Nagumo spiral wave patterns resulting from solving Equations (37) and (38) on the Bumpy Sphere and Bunny models. In all plots, yellow corresponds to a high concentration of u and blue to a low concentration.

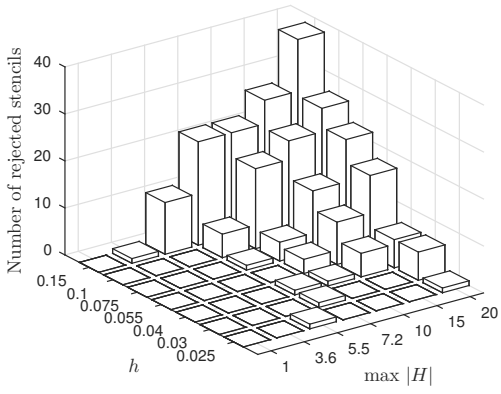


Fig. 10: The number of rejected stencils as a function of h and the mean curvature H.