MESHFREE METHODS FOR PDES ON SURFACES

by

Andrew Michael Jones

A dissertation

submitted i n partial f ulfillment

of the requirements for the degree of

Doctor of Philosophy in Computing

Boise State University

December 2022

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the dissertation submitted by

Andrew Michael Jones

Dissertation Title: Meshfree Methods for PDEs on Surfaces

Date of Final Oral Examination: 20 October 2022

The following individuals read and discussed the dissertation submitted by student Andrew Michael Jones, and they evaluated the student's presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Grady B. Wright Ph.D.	Chair, Supervisory Committee
Michal Kopera Ph.D.	Member, Supervisory Committee
Min Long Ph.D.	Member, Supervisory Committee
Peter A. Bosler Ph.D.	Member, Supervisory Committee

The final reading approval of the dissertation was granted by Grady B. Wright Ph.D., Chair of the Supervisory Committee. The dissertation was approved by the Graduate College.

DEDICATION

To my brothers: Daniel, David, and, Aaron. To my parents: Christina and Sinclair. To my partner and son: Monica and Rhydion.

ACKNOWLEDGMENT

Advisors and Collaborators

- Grady B. Wright
- Peter A. Bosler
- Varun Shankar
- Michal Kopera
- Paul A. Kuberry

Organizations: Boise State University, Sandia National Lab, University of Utah

This work was partially supported the National Science Foundation Grant CCF-1717556. This work was also supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR) Program and Biological and Environmental Research (BER) Program under a Scientific Discovery through Advanced Computing (SciDAC 4) BER partnership pilot project.

ABSTRACT

This dissertation focuses on meshfree methods for solving surface partial differential equations (PDEs). These PDEs arise in many areas of science and engineering where they are used to model phenomena ranging from atmospheric dynamics on earth to chemical signaling on cell membranes. Meshfree methods have been shown to be effective for solving surface PDEs and are attractive alternatives to mesh-based methods such as finite differences/elements since they do not require a mesh and can be used for surfaces represented only by a point cloud. The dissertation is subdivided into two papers and software.

In the first paper, we examine the performance and accuracy of two popular meshfree methods for surface PDEs:generalized moving least squares (GMLS) and radial basis function-finite differences (RBF-FD). While these methods are computationally efficient and can give high orders of accuracy for smooth problems, there are no published works that have systematically compared their benefits and shortcomings. We perform such a comparison by examining their convergence rates for approximating the surface gradient, divergence, and Laplacian on the sphere and a torus as the resolution of the discretization increases. We investigate these convergence rates also as the various parameters of the methods are changed. We also compare the overall efficiencies of the methods in terms of accuracy per computation cost.

The second paper is focused on developing a novel meshfree geometric multilevel

(MGM) method for solving linear systems associated with meshfree discretizations of elliptic PDEs on surfaces represented by point clouds. Multilevel (or multigrid) methods are efficient iterative methods for solving linear systems that arise in numerical PDEs. The key components for multilevel methods: "grid" coarsening, restriction/interpolation operators coarsening, and smoothing. The first three components present challenges for meshfree methods since there are no grids or mesh structures, only point clouds. To overcome these challenges, we develop a geometric point cloud coarsening method based on Poisson disk sampling, interpolation/ restriction operators based on RBF-FD, and apply Galerkin projections to coarsen the operator. We test MGM as a standalone solver and preconditioner for Krylov subspace methods on various test problems using RBF-FD and GMLS discretizations, and numerically analyze convergence rates, scaling, and efficiency with increasing point cloud resolution. We finish with several application problems.

We conclude the dissertation with a description of two new software packages. The first one is our MGM framework for solving elliptic surface PDEs. This package is built in Python and utilizes NumPy and SciPy for the data structures (arrays and sparse matrices), solvers (Krylov subspace methods, Sparse LU), and C++ for the smoothers and point cloud coarsening. The other package is the RBFToolkit which has a Python version and a C++ version. The latter uses the performance library Kokkos, which allows for the abstraction of parallelism and data management for shared memory computing architectures. The code utilizes OpenMP for CPU parallelism and can be extended to GPU architectures.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1:

INTRODUCTION

1.1 Overview

This dissertation is comprised of two papers:

PI. A. M. Jones, G. B. Wright, P. A. Bosler, P. A. Kuberry. A comparison of generalized moving least squares and radial basis function finite difference methods for approximating surface derivatives. Submitted (2022)

PII. G. B. Wright, A. M Jones, and V. Shankar. A meshfree geometric multilevel method for systems arising from elliptic equations on point cloud surfaces. SIAM Journal on Scientific Computing. Accepted (2022)

and two software packages:

SI. A. M. Jones, MGM: Meshfree Geometric Multilevel Solver and Preconditioner. `https://github.com/AndrewJ3/MGM`

SII. A. M. Jones, RBFToolkit: Radial basis function Toolkit.

`https://github.com/AndrewJ3/rbftoolkit`

$\mathbb{PI}$ is reproduced in Appendix A and the author contributions to this paper are described in Chapter 2. Similarly, $\mathbb{PII}$ is reproduced in Appendix B, with author contributions given in Chapter 3. Chapter 4 gives an overview of $\mathbb{SI}$ and $\mathbb{SII}$. Finally,

Chapter 5 gives some concluding remarks on the work. The remainder of this chapter includes relevant background material on the topics of the thesis, overview of the contributions made, and some future work. References for Chapter 1 are given at the end of the chapter, while references for the $\mathbb{PI}$ and $\mathbb{PIII}$ are included with the papers.

1.2 Mathematical modeling

Mathematical models provide a framework for predicting and understanding processes and phenomena in various fields ranging from the natural and social sciences to engineering. The prototypical example comes from classical mechanics formulated by Newton and others in the 17th century. If we have a projectile of mass m subjected to gravitational and linear drag forces $\mathbf{F}_g = m\mathbf{g}$ and $\mathbf{F}_D = -k\mathbf{u}(t)$, respectively and the velocity $\mathbf{u}(t)$ is changing over time t, then according to Newton's second law, we can sum the forces acting on the projectile to obtain a relation for the acceleration of the entire projectile. This gives the following system of ordinary differential equations (ODE) to solve for the velocity:

$$m\frac{d\mathbf{u}(t)}{dt} = -k\mathbf{u}(t) + m\mathbf{g}. \tag{1.1}$$

For a given initial velocity $u(0) = u_0$, this system of ODEs is linear and straightforward to solve analytically. Unfortunately, this is an exceptional situation. It is much more common that mathematical models for real-world applications can never be solved analytically. This can occur because of nonlinear relationships involving the unknowns, the unknowns also depending on space as well as time, and/or the spatial domains being geometrically complex.

Mathematical models for phenomena that depend on multiple variables, such as

two or more spatial variables and/or time, typically lead to partial differential equations (PDEs). A simple example is the diffusion equation, which describes how some quantity diffuses over time through a domain Ω. In three dimensions, the PDE takes the form

$$\frac{\partial u}{\partial t} = \mu \Delta u \tag{1.2}$$

where $\Delta = \partial_{xx} + \partial_{yy} + \partial_{zz}$ is the Laplacian operator and $\mu > 0$ is the diffusion coefficient, which depends on the medium of Ω. To complete the model, we must also specify the initial conditions, and if Ω has boundaries, then we must specify how the quantity behaves at the boundaries (commonly called boundary conditions). The diffusion equation (1.2), and other related linear PDEs like the Poisson and wave equations, can be "solved" with analytical methods like separation of variables or Fourier transforms, only for a very limited number of domains; we illustrate some of these simpler domains in the top row of Figure 1.1. However, even in these cases, the solutions depend on infinite sources and/or integrals that can not be computed analytically. If the domains have any geometric complexity, as illustrated in the bottom row of Figure 1.1. Then these analytical methods fail. In these cases, we are instead forced to solve these models approximately using numerical methods. This thesis focuses on numerical methods for mathematical models on geometrically complex domains, in particular, on PDEs posed on two-dimensional surfaces embedded in three-dimensional space, such as the last three surfaces in Figure 1.1.

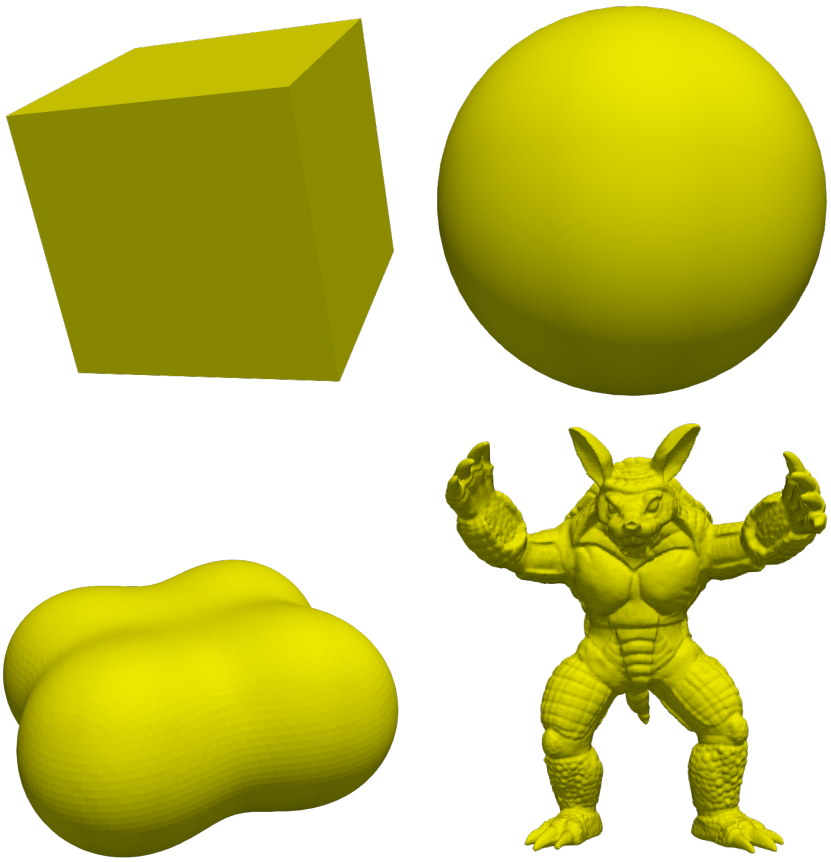


Figure 1.1: Top row: Examples of domains where the diffusion equation (1.2) can be "solved" analytically using separation of variables. Bottom row: examples of domains where separation of variables fails.

1.3 Surface partial differential equations

Surface PDEs arise across many branches of science and engineering, for example, in atmospheric flows [63], bulk surface biomechanics [32], and computer graphics for texture generation [58]. An additional application is to activator and inhibitor systems modeled by surface reaction-diffusion systems; some examples are shown in Figure 1.2. For our purposes, we focus primarily on PDEs like the diffusion and Poisson

equations that are posed on smooth two-dimensional closed surfaces $\mathcal{M} \subset \mathbb{R}^3$.

$$\text{Surface Diffusion Equation: } u_t - \Delta_{\mathcal{M}} u = f, \tag{1.3}$$

$$\text{Screened Surface Poisson Equation: } (\alpha - \Delta_{\mathcal{M}})u = f. \tag{1.4}$$

Here $\Delta_{\mathcal{M}}$ is the surface Laplacian (or Laplace-Beltrami operator) on $\mathcal{M}$, and α is a parameter. The screened Poisson equation is equivalent to the non-homogenous Helmholtz equation with imaginary wave number. When $\alpha = 0$, the surface Poisson equation is recovered. For the remainder of the section, we discuss the historical development of surface PDEs and the mesh-based methods commonly used for solving and discretization (SE, FE, FV, FD). Lastly, we discuss the benefits of meshfree methods for surface PDEs.

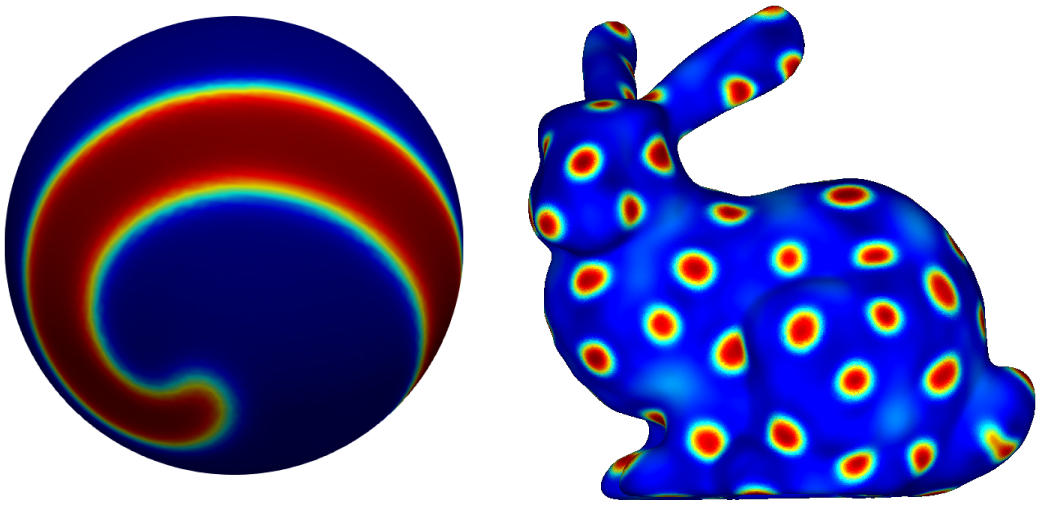


Figure 1.2: Illustrations of solutions to coupled reaction diffusion PDEs Fitzhugh Nagumo spiral wave (left) on a unit sphere and Turing spots (right) on the Standford bunny.

Early numerical studies of surface PDEs were mainly restricted to the sphere

and applied to problems in numerical weather prediction (NWP) [12, 51, 62, 43]. These early studies mainly utilized finite difference and spectral methods, but in the 1980s and onward, additional methods were introduced based on finite volume methods (FV) [44, 64] and spectral/finite element methods (FE/SE) [33, 54, 21]. Concurrently, solving PDEs on general surfaces was examined in the 1980s by [13] using surface FE (SFE) methods, and as interest in these PDEs grew, techniques were developed such as embedded finite element (EFE) [8, 35], and closest point (CP) [31] methods. Development of meshfree methods for surface PDEs began in the early 2010s using global RBF methods [38, 19]; and development of localized versions of these methods soon followed, including radial basis function finite differences (RBF-FD) method [1, 28, 45, 39, 46, 61], generalized finite difference (GFD) methods [52], and generalized moving least squares (GMLS) [29, 56, 23].

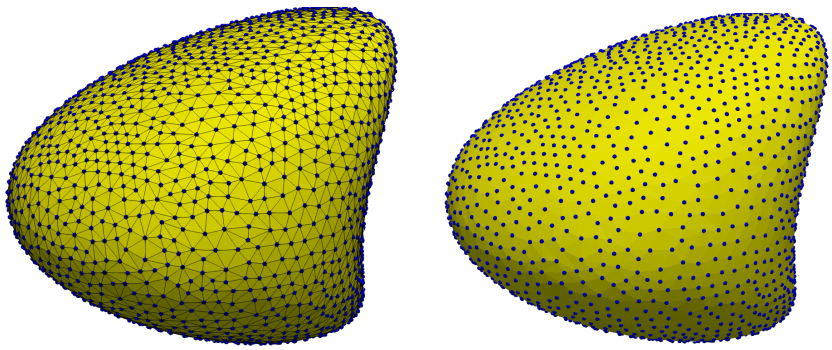


Figure 1.3: Illustrations of a surface discretized with a triangular tessellation (left) and with a point cloud (right).

In contrast with FE-based methods, which use tessellation of triangular or quadrilateral elements like SFE, and also they do not extend into the embedding space like EFE, meshfree techniques only require nodal points of mesh or simply a point cloud as illustrated by Figure 1.3. This allows for more algorithmic flexibility. Meshfree

methods can also proce high order accurate approximationsfor smooth problems. This dissertation focuses on RBF-FD and GFD/GMLS methods for solving surface PDEs. Before outlining the motivations and contributions for this dissertation, we give a brief overview of the meshfree methods used in this work.

1.4 Global meshfree interpolation and approximation

1.4.1 Radial basis functions

RBFs were originally developed in the 1970s for scattered data interpolation problems arising in cartography [24] and have subsequently been used in many applications, including for numerically solving PDEs starting in 1990 [26]. The RBF interpolation process is illustrated in Figure 1.4. This figure shows a reconstruction of data collected at scattered nodes using Gaussian RBFs. Starting with scattered data, the RBF method produces an interpolant of this data using a linear combination of rotations of a single radial kernel (e.g. the Gaussian) centered at each of the data sites.

For $\mathbf{x}, \mathbf{x}_j \in \mathbb{R}^d$, a general radial kernel centered at $\mathbf{x}_j$ is defined as $\Phi(\mathbf{x}, \mathbf{x}_j) := \phi(||\mathbf{x} - \mathbf{x}_j||)$ where $||\cdot||$ is the standard Euclidean norm and ϕ is a scalar function. Given a set of N points $X = \{\mathbf{x}_k\}_{k=1}^N \subset \mathbb{R}^d$, the basic RBF interpolant to a function f sampled at X takes the form,

$$s(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi(||\mathbf{x} - \mathbf{x}_j||). \tag{1.5}$$

where the coefficients c_j are determined by the interpolation conditions,

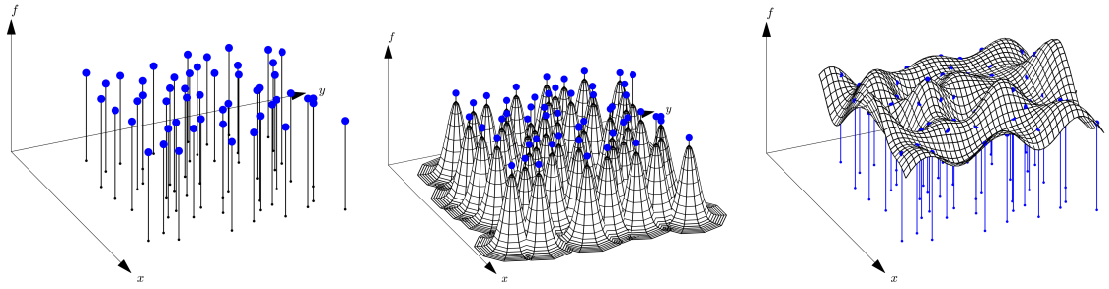


Figure 1.4: Illustration of RBF interpolation of 2D scattered data: (a) scattered data (b) radial basis functions (Gaussians), centered at data locations, and (c) interpolant of (a) .

$$\sum_{j=1}^{N} c_j \phi(||\mathbf{x}_i - \mathbf{x}_j||) = f(\mathbf{x}_i) \ , \ i = 1, \ldots, N, \tag{1.6}$$

These conditions can be written as the following

$$\underbrace{\begin{bmatrix} \phi(||\mathbf{x}_1 - \mathbf{x}_1||) & \ldots & \phi(||\mathbf{x}_1 - \mathbf{x}_N||) \\ \vdots & \ddots & \vdots \\ \phi(||\mathbf{x}_N - \mathbf{x}_1||) & \ldots & \phi(||\mathbf{x}_N - \mathbf{x}_N||) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix}}_{\boldsymbol{c}} = \underbrace{\begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}}_{\boldsymbol{f}} . \tag{1.7}$$

Some commonly used RBFs are the Gaussian, $\phi(r) = \exp(-(\epsilon r)^2)$, multiquadric, $\phi(r) = \sqrt{1 + (\epsilon r)^2}$, and polyharmonic splines (PHS), $\phi(r) = r^{2\ell+1}$ and $r^{2\ell} \log r$, where $\epsilon \in \mathbb{R}^+$ is the shape parameter and $\ell \in \mathbb{Z}^+$ is the smoothness parameter. In this dissertation, we use PHS, which are advantageous because they do not require shape parameters. Choosing suitable shape parameters often requires expensive optimization algorithms [15], and the interpolation matrix can become extremely ill-conditioned for small shape parameters, prompting the use of so-called stable algorithms [18].

When using PHS, one often appends on low degree polynomials to (1.4.1) to guar-

antee the well-posedness of the interpolation problem. However, more importantly, this has the added benefit of improving approximation convergence rates and allows for polynomial reproduction [18]. The new form of the interpolant is as follows:

$$s(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi(||\mathbf{x} - \mathbf{x}_j||) + \sum_{k=1}^{L} b_k p_k(\mathbf{x}), \tag{1.8}$$

where $\{p_k\}_{k=1}^{L}$ are a basis for d-variate polynomials of degree ℓ and L is the dimension of this space. To account for the new L coefficients, b_k, the interpolant is subject to the following moment conditions [60, 45]:

$$\sum_{k=1}^{N} c_k p_j(\mathbf{x}_k) = 0, \;\; j = 1, 2, \ldots, L.$$

One issue with global RBF interpolation methods is that the linear system (1.7) is dense and not well suited for iterative methods. Direct methods require $O(N^3)$ computational cost, which becomes prohibitive for large N. To fix this issue, we can use localized, stencil-based approximations as discussed in Section 1.5.

1.4.2 Polynomial moving least squares (MLS)

The development of MLS methods are mainly based on work done in the 1960s using moving averages for approximating irregular multivariate data in geophysics [48, 3]. Work involving such data had similar applications as RBF methods, such as meteorology and geography. MLS was refined from early 1980s [27], with extensions to approximating PDEs in the 1990s [7] and the 2000s [30]. A MLS approximant to data

$\{f_1, f_2, \ldots, f_N\}$ sampled at X takes the form

$$q(\mathbf{x}) = \sum_{k=1}^{L} b_k(\mathbf{x}) p_k(\mathbf{x}), \tag{1.9}$$

where $\{p_k\}_k^L$ are again a basis for d-variate polynomials of degree ℓ. The coefficients of the approximant are determined from the following weighted least squares problem:

$$b^*(\mathbf{x}) = \underset{b \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{N} w_\rho(\mathbf{x}_i, \mathbf{x})(q(\mathbf{x}_i) - f_i)^2 = \underset{b \in \mathbb{R}^n}{\operatorname{argmin}} \|W(\mathbf{x})^{1/2}(Pb - f)\|_2^2, \tag{1.10}$$

where $W = diag(w_\rho(\mathbf{x}_i, \mathbf{x}))$ and w_ρ is a non-negative kernel. Note that the coefficients depend on $\mathbf{x}$ because of the weight kernel. This is the origin of the name "moving" for MLS methods. Some examples of weighting kernels are given as follows,

$$w_\rho(\mathbf{x}_i, \mathbf{x}) = \left(1 - \frac{\|\mathbf{x}_1 - \mathbf{x}_i\|}{\rho}\right)_+^4 \text{ and } w_\rho(\mathbf{x}_i, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_i\|^2}{\rho^2}\right), \tag{1.11}$$

where ρ is the support radius and $()_+$ is the positive floor operator. These weighting kernels will be reintroduced in later chapters when we discuss the GMLS methods.

We can pose (B.12) as following (weighted) linear system

$$P^T W_\rho(\mathbf{x}) P\mathbf{b} = P^T W_\rho(\mathbf{x})\mathbf{f}, \tag{1.12}$$

which are the normal equations to the weighted least squares problem. Provided that P has full rank and $W_\rho(\mathbf{x})$ has strictly positive entries on its diagonal (1.12) has a unique solution. Note that this system must be solved for each evaluation point $\mathbf{x}$, making MLS computationally expensive.

1.5 Localized meshfree methods

Localized meshfree methods were developed to reduce the high computational cost associated with global methods like RBFs. These methods use local interpolants/approximants over small stencils of points chosen from the global point set to approximate derivatives, similar to finite difference methods.

1.5.1 Stencils and nearest neighbor searches

A stencil is a collection of $n << N$ points from a global point set $X = \{\mathbf{x}_j\}_{j=1}^N$ used for approximating a function f or some derivative of f at a point $\mathbf{x}_c$, called the stencil center. Some examples of stencils are provided in Figure 1.5 for structured and unstructured points X and $n = 5$. We denote a stencil with center $\mathbf{x}_c = \mathbf{x}_k \in X$ as $X_k = \{\mathbf{x}_j\}_{j\in\sigma_k}$, where σ_k is the the index set containing the indices of the points in X contained in a stencil. The points are typically chosen as some collection of the nearest neighbors to $\mathbf{x}_c$. Figure A.1 illustrates two different techniques for determining these nearest neighbors: ball and K nearest neighbors (KNN), which can be implemented efficiently using a k-dimensional tree. Using the above notation, we can write a stencil-based approximation to a linear differential operator $\mathcal{L}$ applied to a function u sampled at X as

$$\mathcal{L}u|_{\mathbf{x}=\mathbf{x}_i} \approx \sum_{j\in\sigma_i}^{n} c_{ij}u(\mathbf{x}_j),\ i = 1,\ldots,n. \tag{1.13}$$

The weights c_{ij} depend on the locations, point spacings, stencil size, and approximation methods [18], which we discuss in the next section. Note that these weights can be assembled into a sparse $N \times N$ "stiffness matrix".

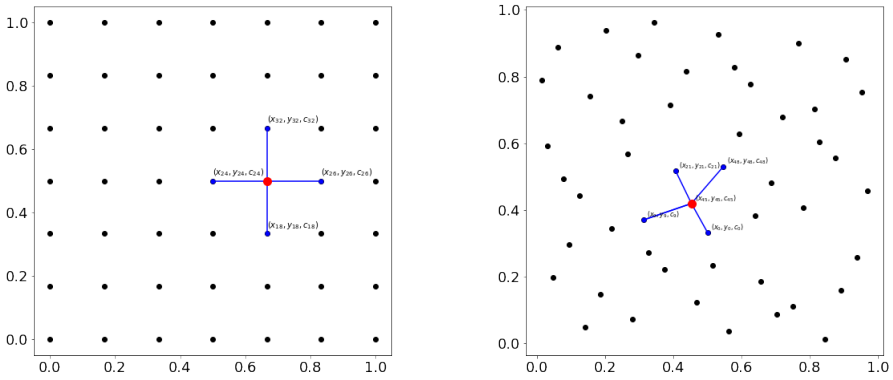


Figure 1.5: Structure (left) and unstructured (right) points on a square domain with a five-point stencil. The red point is the stencil center, and the blue points represent the stencil neighbors.

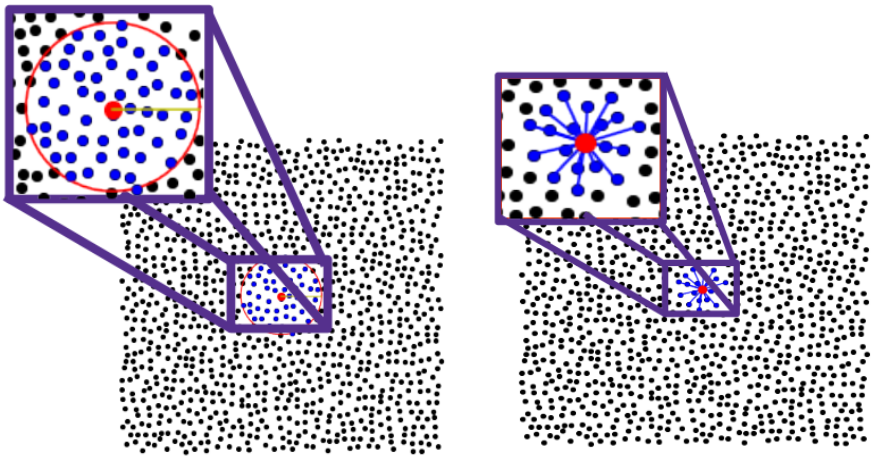


Figure 1.6: Illustration of two algorithms for determining nearest neighbors: ϵ-ball (left) and KNN search (right). The red point is the stencil center, and the blue points represent the stencil neighbors.

1.5.2 RBF-FD

Suppose the first stencil contains the points $X_1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Then RBF-FD determines the weights c_{1j} in (A.1) as the solution to the following system,

$$\sum_{j=1}^{n} c_{1j}\phi(||vx_1 - \mathbf{x}_j||) = \mathcal{L}\phi(||\mathbf{x} - \mathbf{x}_i||)|_{\mathbf{x}=\mathbf{x}_1} \ , \ i = 1, \ldots, n, \tag{1.14}$$

under the constraint that the weights $\{c_{1j}\}$ are exact for all polynomials of degree ℓ:

$$\sum_{j=1}^{n} c_{1j}p_k(\mathbf{x}_j) = \mathcal{L}p_k(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} \quad k = 1, \ldots, L,$$

where $\{p_k\}_{k=1}^{L}$, are again a basis for the set of polynomials of degree ℓ. As described in detail by [18], the weights satisfying (1.14) under the above constraints can be computed by solving the following linear system:

$$\begin{bmatrix} A & P \\ P^T & O \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi(||\mathbf{x} - \mathbf{x}_j||)|_{\mathbf{x}=\mathbf{x}_1} \\ \mathcal{L}P(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} \end{bmatrix}, \tag{1.15}$$

where A is given in (1.7) and $P_{jk} = p_k(\mathbf{x}_j)$, and $\boldsymbol{\gamma}$ is a Lagrange multiplier. The RBF-FD weights for all the remaining stencils $X_i, i = 2, \ldots, N$ can be computed similarly using (1.15).

1.5.3 GMLS/GFD

The GMLS and GFD methods are very similar (and, in most cases, identical). The procedure for generating FD-type weights is similar to RBF-FD, except a local weighted polynomial least squares problem (B.12) is used. Using the notation from the previous section, the weights for the first stencil X_1 can be computed from

the normal equations as follows:

$$\mathbf{c}_1 = W_\rho(\mathbf{x}_1)P(P^T W_\rho(\mathbf{x}_1)P)^{-1}\mathcal{L}\mathbf{p}|_{\mathbf{x}_c}. \tag{1.16}$$

In practice, one would use a QR factorization of $W_\rho(\mathbf{x}_1)P$ for one to solve (1.16) in a numerically stable manner.

1.6 Node generation

Node or point cloud generation are initial or preprocessing stages of solving meshfree discretizations of PDEs on surfaces. Overall there are mesh-based and meshfree techniques for obtaining a point cloud. The mesh-based node generation can be straight-forward, since one simply requires extracting nodes from mesh elements, which must have good quality. A wide range of open source mesh generation packages are available such as Meshlab [37] or Gmsh [20]. Using mesh-based node generation can be computationally expensive and wasteful since we do not require all element information (i.e. faces, edges). Meshfree node generation has been explored on general planar domains and surfaces [66, 17, 47, 49] using node repulsion algorithms and Poisson disk sampling.

1.7 Contributions of this dissertation

1.7.1 Contributions of PI

Motivation

Many multiphysics problems require the computation of derivatives on two-dimensional surfaces embedded in $\mathbb{R}^3$. For example, simulating atmospheric flows with Eulerian or Lagrangian numerical methods requires approximating surface gradients, diver-

gence, and Laplacians of various quantities like wind, pressure, and bathymetry on the sphere [59, 43, 63, 18, 9].Similar differential operators must be approximated for much more complicated surfaces than the sphere in application such as glaciology [22] in surface chemistry [65], computer graphics [34], multiphase flows with surfactants [14], sea-air hydrodynamics [4, 2] and bulk-surface biomechanics [13, 42]. In short, surface derivatives are of great importance across many areas of science and engineering, and efficient and accurate methods are required for approximating these quantities.

RBF-FD and GMLS have separately been developed for this task and have shown to be quite effective since they can produce high orders of accuracy at low computational cost, and they do not require any gridding or meshing. While some studies have been published comparing these methods for approximating functions and derivatives in $\mathbb{R}^2$ and $\mathbb{R}^3$ [6]. No published studies compare these methods for approximating derivatives on surfaces. This study aims to fill this gap and builds on the technical report of Jones and Bosler [25].

Overview

We examine the performance and accuracy of the two methods for approximating the surface gradient, divergence, and Laplacian. The focus is on how the methods compare with each other when stencil sizes, polynomial degrees, differential operators, and point clouds vary. We additionally show for the first time the equivalence of the two local formulations of the surface derivatives, which are designated as the "local coordinate method" and "tangent plane method".

For the problems we tested, we found that RBF-FD and GMLS methods converge

at nearly the same rates when using the same polynomial degree, and RBF-FD gives lower errors for the same degrees of freedom. This is illustrated in Figure 1.7 for the surface Laplacian on the torus and sphere. Additionally, when comparing the accuracy of the methods versus the computational cost, we found that RBF-FD performs better when set-up costs are neglected. When these are accounted for, GMLS is more competitive. This is illustrated for the Laplacian on the sphere in Figure 1.8.

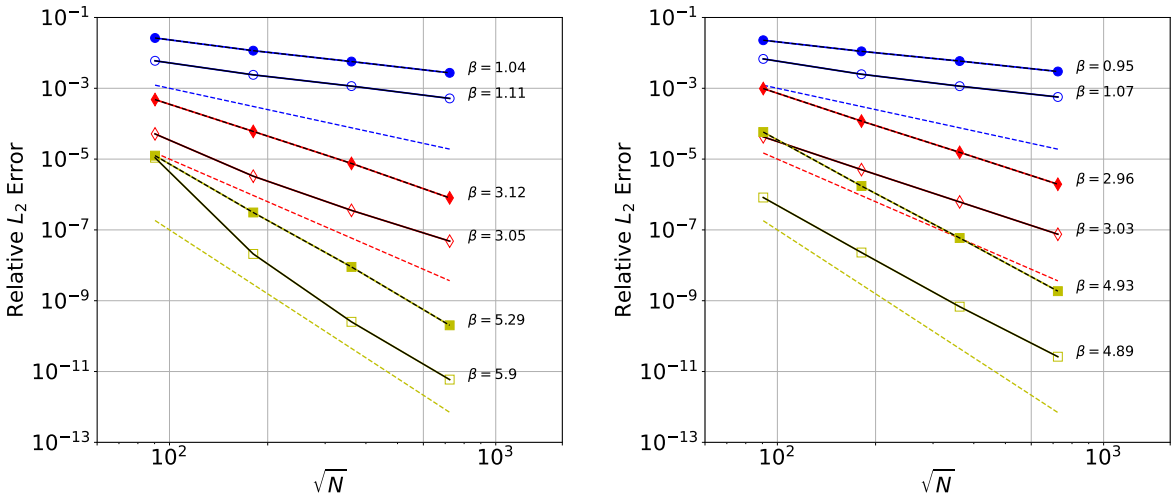


Figure 1.7: Convergence plots for approximating the surface Laplacian on the torus (left) and sphere (right).

1.7.2 Contributions of PII

Motivation

Meshfree methods for PDEs such as RBF-FD and GFD lead to large sparse linear systems of equations that need to be solved. To make these methods practical for large-scale problems, effective iterative methods need to be developed for solving these systems. One class of iterative methods that are particularly effective at solving systems associated with standard FD discretizations are multilevel methods, such

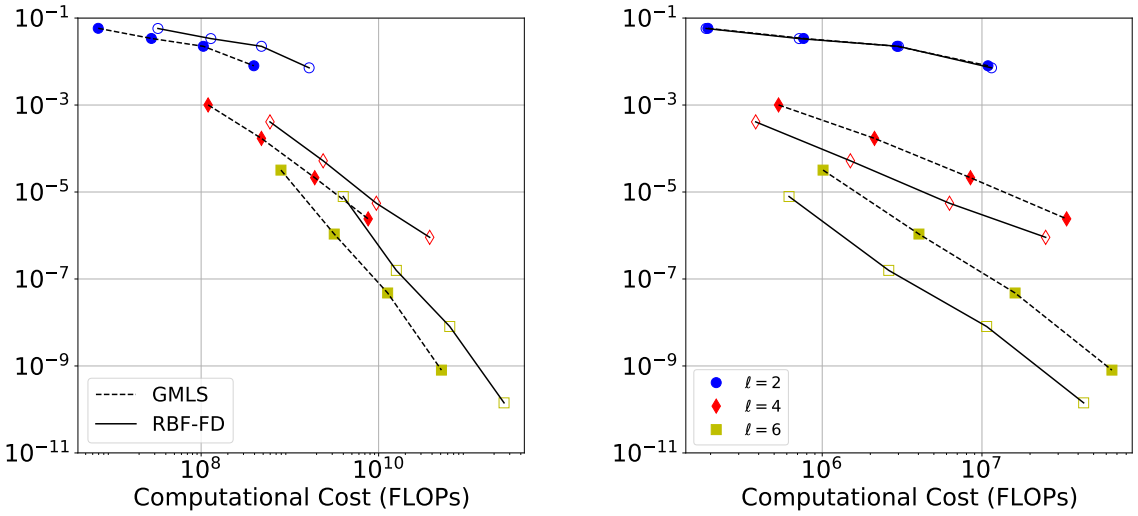


Figure 1.8: Computational efficiency (Error vs. FLOPs) for approximating the surface Laplacian on the sphere: set-up (left) and run-time (right) costs.

as geometric multigrid [10, 57]. However, since meshfree methods do not have an underlying grid structure, the extension of these methods to this setting is not apparent. There is also the need for solvers that can handle degenerate PDEs (surface Poisson equation) and high-order PDE discretizations. In this project, we focus on solving these challenges and develop a meshfree geometric multilevel (MGM) method for RBF-FD and GFD discretizations.

Overview

The components for any multilevel method are: a method for coarsening the points, a solver for the coarse level correction equations, restriction methods for the residuals, and an interpolation method for the corrections [10, 57]. The term "smoother" arises from the nature of a relaxation method (e.g. SOR, damped Jacobi, or Gauss-Seidel)

to "filter" the high-frequency oscillations present in the error. This smoothing step is conducted on all grid levels until the coarsest grid is reached, where the coarse grid solution is computed. The next step is to prolongate (interpolate) the coarse solution to the original fine grid. This completes what is called the V-cycle iteration [10, 57], illustrated is shown in Figure 1.9.

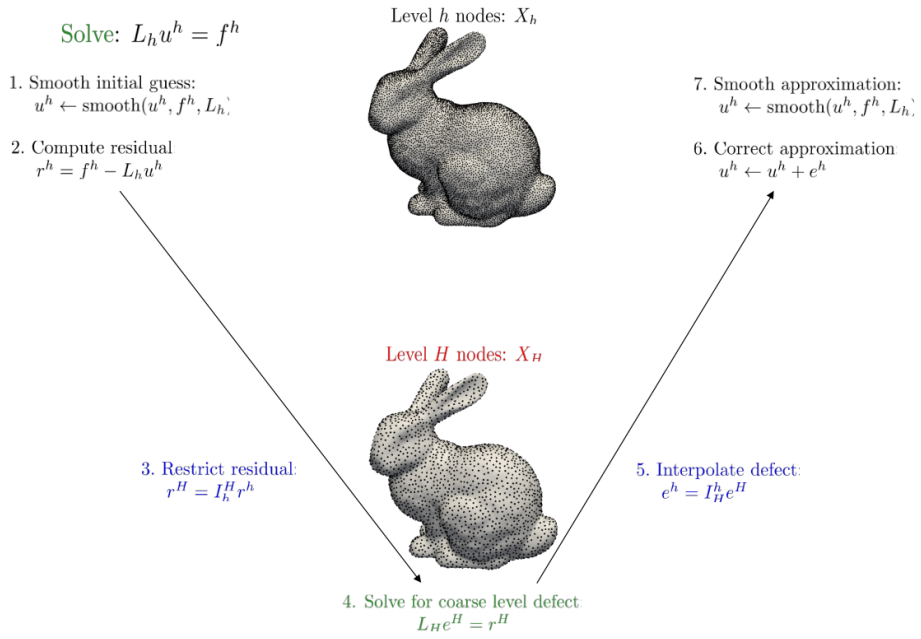


Figure 1.9: A schematic for a two-level V-cycle method for solving an elliptic PDE.

For a meshfree treatment of multilevel methods, we must have a hierarchy of increasingly coarser point clouds rather than grids. We address this challenge using a point cloud coarsening algorithm called weighted sample elimination [66]. The interpolation and restriction is done using RBF-FD with polyharmonic splines (PHS) plus constants. For the smoothing, we apply forward Gauss-Seidel, and a SparseLU for the coarse level solve. In the numerical tests, we examine the convergence with increasing polynomial and PHS degree ℓ using two meshfree discretizations: RBF-FD

and GFD. MGM is used as preconditioner and a independent solver and compared with an algebraic multigrid software package PyAMG [36], some results for solving a screened Poisson problem 1.4 are displayed in Figure 1.10. We finish this paper with several application problems, computing geodesics distance and simulating pattern formation driven by nonlinear reaction-diffusion; see Figure 1.11 for an illustration.

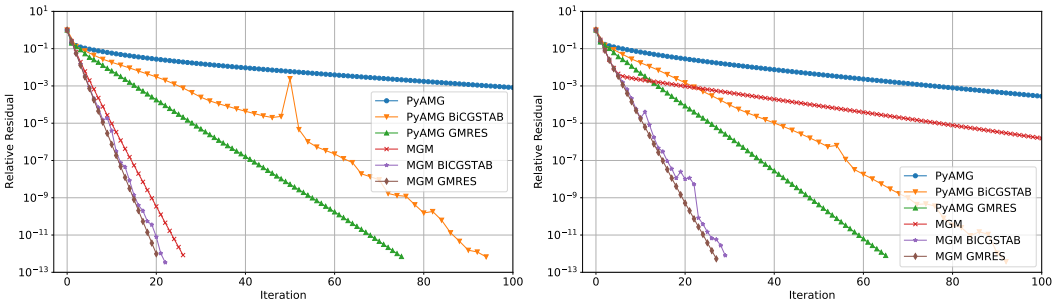


Figure 1.10: Illustration of the residual convergence results for $\ell = 7$ RBF-FD discretizations of the sphere (left) and a cyclide (right) using MGM, PyAMG, preconditioners for GMRES and BiCGStab.

Figure 1.11: Left: Visualizations of the geodesic distance from the black dot marked on the Armadillo. The colormap transitions from white to yellow to indicate the increases in the distance. Right: Pattern formation on the Stanford bunny.

1.7.3 Contributions of PI and PII

We present two new software packages that were developed in conjunction with this thesis, MGM and RBFToolkit. MGM is a Python based meshfree multilevel solver for elliptic PDEs. The libraries NumPy, SciPy, PyAMG handle the data structures (n-dimensional arrays) and the numerical linear algebra (Krylov subspace methods, relaxation/smoothers methods), and Matplotlib provides the data visualization. The link to the code repository is `https://github.com/AndrewJ3/MGM`, the repository homepage is shown in Figure 1.12. RBFToolkit will be a refactor of the RBFKokkos C++ code presented in [25]. This updated version will be written in Python and will include an improved C++ and Kokkos version. The repository can be found with

the following link `https://github.com/AndrewJ3/rbftoolkit`, and the homepage is shown in Figure 1.12.

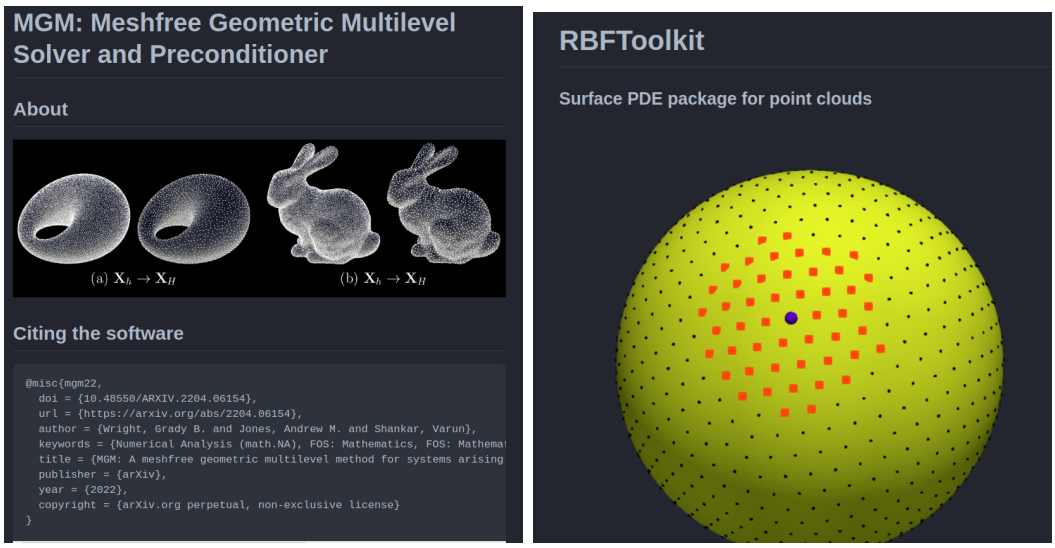


Figure 1.12: Github repositories for MGM and RBFToolkit.

1.8 Future work

Most studies of PDEs on the sphere focus on numerical weather prediction [51, 33, 43] and in the last century, numerous discretizations of the sphere have arisen. Early works in this area focus on latitude-longitude grids, which have failures due to singularities at the poles. This has prompted the examination of other spherical discretizations. Some of these other discretizations are the cubed sphere [44, 40] and icosahedral mesh, [5], which each have a variety of different arrangements. These grids are commonly used in FV/FE methods [63, 54]. Another family of "grids" is based on ideal spiral arrangements that can be derived from the Fibonacci sequence, referenced as "Fibonacci" grids [53]. Pseudo-random point distributions for the sphere have also been studied for point picking [31] and Monte Carlo methods [55]. Also, there is

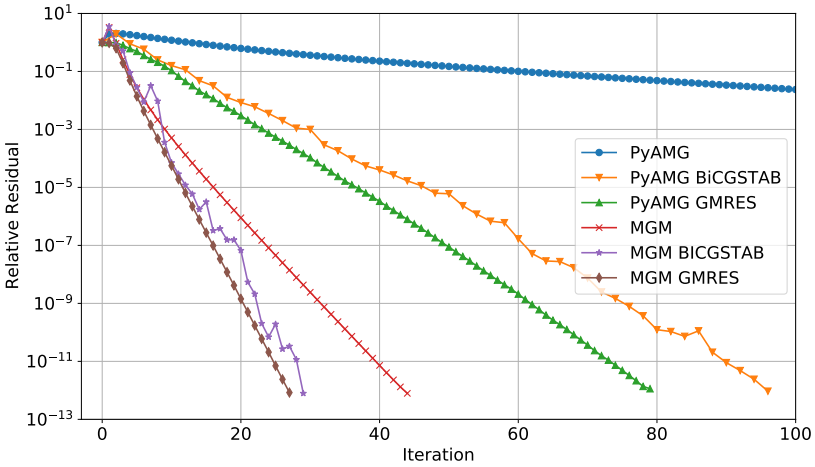


Figure 1.13: Comparison of residual convergence of various itertive methods for solving the screened Poisson equation on the sphere using a SE based discretization.

Poisson disk sampling which can produce quasi-uniform points efficiently [11]. Another study relevant to meshfree methods involves minimal energy [41] and maximum determinant [50] point clouds for the sphere, which have been used with RBF-FD for atmospheric flows [16]. Overall, no comprehensive study of PDEs on the sphere has been conducted for this wide range of point clouds and PDE discretizations. We address these areas of interest with some preliminary results using MGM to solve a SE discretization of the screened Poisson equation (1.4). We compare MGM with PyAMG and find improved convergence rates when using MGM; these results are illustrated in Figure 1.13. Future work related to this dissertation, we will study the various sphere discretizations with MGM.

REFERENCES

[1] Álvarez, Diego, González-Rodríguez, Pedro, and Kindelan, Manuel. 2021. A Local

Radial Basis Function Method for the Laplace–Beltrami Operator. *J. Sci. Comput.*, **86**(3), 28.

[2] Asher, William E., Liang, Hanzhuang, Zappa, Christopher J., Loewen, Mark R., Mukto, Moniz A., Litchendorf, Trina M., and Jessup, Andrew T. 2012. Statistics of surface divergence and their relation to air-water gas transfer velocity. *Journal of Geophysical Research: Oceans*, **117**(C5).

[3] Backus, George, and Gilbert, Freeman. 1968. The Resolving Power of Gross Earth Data. *Geophysical Journal International*, **16**(2), 169–205.

[4] Banerjee, Sanjoy, Lakehal, Djamel, and Fulgosi, Marco. 2004. Surface divergence models for scalar exchange between turbulent streams. *International Journal of Multiphase Flow*, **30**(7), 963–977. A Collection of Papers in Honor of Professor G. Yadigaroglu on the Occasion of his 65th Birthday.

[5] Baumgardner, John, and Frederickson, Paul O. 1985. Icosahedral Discretization of the Two-Sphere. *SIAM Journal on Numerical Analysis*, **22**, 1107–1115.

[6] Bayona, Victor. 2019. Comparison of Moving Least Squares and RBF+Poly for Interpolation and Derivative Approximation. *Journal of Scientific Computing*, **81**(1), 486–512.

[7] Belytschko, T., Lu, Y. Y., and Gu, L. 1994. Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering*, **37**(2), 229–256.

[8] Bertalmío, M., Cheng, L., Osher, S., and Sapiro, G. 2001. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, **174**, 759–780.

[9] Bosler, P.A., Wang, L., Jablonowski, C., and Krasny, R. 2014. A Lagrangian particle/panel method for the barotropic vorticity equations on a rotating sphere. *Fluid Dyn. Res.*, **46**.

[10] Brandt, A., and Livne, O.E. 2011. *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics.

[11] Bridson, Robert. 2007. Fast Poisson Disk Sampling in Arbitrary Dimensions. 22–es.

[12] Charney, J. G., FjÖrtoft, R., and Neumann, J. Von. 1950. Numerical Integration of the Barotropic Vorticity Equation. *Tellus*, **2**(4), 237–254.

[13] Dziuk, G. 1988. Finite elements for the Beltrami operator on arbitrary surfaces. *In:* Hildebrandt, S., and Leis, R. (eds), *Partial Differential Equations and Calculus of Variations*. Lecture Notes in Mathematics, vol. 1357. Berlin: Springer.

[14] Erik Teigen, Knut, Song, Peng, Lowengrub, John, and Voigt, Axel. 2011. A diffuse-interface method for two-phase flows with soluble surfactants. *Journal of Computational Physics*, **230**(2), 375–393.

[15] Fasshauer, G., and Zhang, J. 2007. On choosing "optimal" shape parameters for RBF approximation. *Numerical Algorithms*, **45**, 345–368.

[16] Flyer, Natasha, and Wright, Grady B. 2009. A radial basis function method for the shallow water equations on a sphere. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **465**(2106), 1949–1976.

[17] Fornberg, Bengt, and Flyer, Natasha. 2015a. Fast generation of 2-D node distributions for mesh-free PDE discretizations. *Computers Mathematics with Applications*, **69**(7), 531–544.

[18] Fornberg, Bengt, and Flyer, Natasha. 2015b. *A Primer on Radial Basis Functions with Applications to the Geosciences*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

[19] Fuselier, Edward J., and Wright, Grady B. 2013. A High-Order Kernel Method for Diffusion and Reaction-Diffusion Equations on Surfaces. *Journal of Scientific Computing*, **56**(3), 535–565.

[20] Geuzaine, Christophe, and Remacle, Jean-François. 2009. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, **79**(11), 1309–1331.

[21] Giraldo, Francis X. 1997. Lagrange–Galerkin Methods on Spherical Geodesic Grids. *Journal of Computational Physics*, **136**(1), 197–213.

[22] Gowan, Evan J., Zhang, Xu, Khosravi, Sara, Rovere, Alessio, Stocchi, Paolo, Hughes, Anna L. C., Gyllencreutz, Richard, Mangerud, Jan, Svendsen, John-Inge, and Lohmann, Gerrit. 2021. A new global ice sheet reconstruction for the past 80 000 years. *Nature Communications*, **12**(1), 1199.

[23] Gross, B.J., Trask, N., Kuberry, P., and Atzberger, P.J. 2020. Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: A Generalized Moving Least-Squares (GMLS) approach. *Journal of Computational Physics*, **409**, 109340.

[24] Hardy, Rolland L. 1971. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research (1896-1977)*, **76**(8), 1905–1915.

[25] Jones, Andrew M., and Bosler, Peter A. Radial Basis Functions in the Tangent Plane: Meshfree Approximation Methods for the Sphere. *Computer Science Research Institute Summer Proceedings 2020*, 57–67.

[26] Kansa, E.J. 1990. Multiquadrics-A scattered data approximation scheme with applications to computational fluid-dynamics-II solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers and Mathematics with Applications.*

[27] Lancaster, P., and Salkanskas, K. 1980. Surface Generated by Moving Least Squares Methods. *American Mathematical Society.*

[28] Lehto, E, Shankar, V, and Wright, G. B. 2017. A radial basis function (RBF) compact finite difference (FD) scheme for reaction-diffusion equations on surfaces. *SIAM J. Sci. Comput.*, **39**, A219–A2151.

[29] Liang, Jian, and Zhao, Hongkai. 2013. Solving Partial Differential Equations on Point Clouds. *SIAM J. Sci. Comput.*, **35**(3), A1461–A1486.

[30] Liu, Gui-Rong. 2009. *Meshfree methods: moving beyond the finite element method.* Taylor & Francis.

[31] MacDonald, C. B., and Ruuth, S. J. 2009. The implicit closest point method for the numerical solution of partial differential equations on surfaces. *SIAM J. Sci. Comput.*, **31**, 4330–4350.

[32] Madzvamuse, Anotida, Chung, Andy H. W., and Venkataraman, Chandrasekhar. 2015. Stability analysis and simulations of coupled bulk-surface reaction diffusion systems. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **471**(2175), 20140546.

[33] Mailhot, J., and Benoit, R. 1982. A Finite-Element Model of the Atmospheric Boundary Layer Suitable for Use with Numerical Weather Prediction Models. *Journal of Atmospheric Sciences*, **39**(10), 2249 – 2266.

[34] Mikkelsen, Morten S. 2020. Surface Gradient–Based Bump Mapping Framework. *Journal of Computer Graphics Techniques (JCGT)*, **9**(4), 60–91.

[35] Olshanskii, Maxim A., Reusken, Arnold, and Grande, Jörg. 2009. A Finite Element Method for Elliptic Equations on Surfaces. *SIAM Journal on Numerical Analysis*, **47**(5), 3339–3358.

[36] Olson, L. N., and Schroder, J. B. 2018. *PyAMG: Algebraic Multigrid Solvers in Python v4.0*. Release 4.0.

[37] Paolo Cignoni, Alessandro Muntoni, Guido Ranzuglia Marco Callieri. *MeshLab*.

[38] Piret, Cécile. 2012. The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces. *J. Comput. Phys.*, **231**, 4662–4675.

[39] Piret, Cécile, and Dunn, Jarrett. 2016. Fast RBF OGr for solving PDEs on arbitrary surfaces. *AIP Conference Proceedings*, **1776**(1), 070005.

[40] Putman, William M., and Lin, Shian-Jiann. 2007. Finite-volume transport on various cubed-sphere grids. *Journal of Computational Physics*, **227**(1), 55–78.

[41] Rakhmanov, Evguenii A, Saff, Edward B, and Zhou, YM1306011. 1994. Minimal discrete energy on the sphere. *Mathematical Research Letters*, **1**(6), 647–662.

[42] Rätz, Andreas, and Röger, Matthias. 2014. Symmetry breaking in a bulk-surface reaction-diffusion model for signaling networks. *Nonlinearity*, **27**(8), 1805.

[43] Richardson, Lewis Fry, and Lynch, Peter. 2007. *Weather Prediction by Numerical Process*. 2 edn. Cambridge Mathematical Library. Cambridge University Press.

[44] Ronchi, C., Iacono, R., Struglia, M.V., Rossi, A., Truini, C., Paolucci, P.S., and Pratesi, S. 1997. - The cubed sphere: A new method for solving PDEs on the sphere. applications to climate modeling and planetary circulation problems. *Pages 31–38 of:* Schiano, P., Ecer, A., Periaux, J., and Satofuka, N. (eds), *Parallel Computational Fluid Dynamics 1996*. Amsterdam: North-Holland.

[45] Shankar, Varun, Wright, Grady B., Kirby, Robert M., and Fogelson, Aaron L. 2014. A Radial Basis Function (RBF)-Finite Difference (FD) Method for Diffusion and Reaction-Diffusion Equations on Surfaces. *J. Sci. Comput.*, **63**(3), 745–768.

[46] Shankar, Varun, Narayan, Akil, and Kirby, Robert M. 2018a. RBF-LOI: Augmenting Radial Basis Functions (RBFs) with Least Orthogonal Interpolation (LOI) for solving PDEs on surfaces. *J. Comput. Phys.*, **373**, 722–735.

[47] Shankar, Varun, Kirby, Robert M., and Fogelson, Aaron L. 2018b. Robust Node Generation for Mesh-free Discretizations on Irregular Domains and Surfaces. *SIAM Journal on Scientific Computing*, **40**(4), A2584–A2608.

[48] Shepard, Donald. 1968. A Two-Dimensional Interpolation Function for Irregularly-Spaced Data. *Page 517–524 of: Proceedings of the 1968 23rd ACM*

National Conference. ACM '68. New York, NY, USA: Association for Computing Machinery.

[49] Slak, Jure, and Kosec, Gregor. 2019. On generation of node distributions for meshless PDE discretizations. *SIAM journal on scientific computing*, **41**(5), A3202–A3229.

[50] Sloan, Ian H, and Womersley, Robert S. 2004. Extremal systems of points and numerical integration on the sphere. *Advances in Computational Mathematics*, **21**(1), 107–125.

[51] Spinelli, R. A. 1965. Poisson Equation on a Sphere. *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, **2**(3), 489–499.

[52] Suchde, Pratik, and Kuhnert, Jörg. 2019. A meshfree generalized finite difference method for surface PDEs. *Comp. Math. Appl.*, **78**(8), 2789–2805.

[53] Swinbank, Richard, and James Purser, R. 2006. Fibonacci grids: A novel approach to global modelling. *Quarterly Journal of the Royal Meteorological Society*, **132**(619), 1769–1793.

[54] Taylor, Mark, Tribbia, Joseph, and Iskandarani, Mohamed. 1997. The Spectral Element Method for the Shallow Water Equations on the Sphere. *Journal of Computational Physics*, **130**(1), 92–108.

[55] Tichy, Robert F. 1990. Random points in the cube and on the sphere with applications to numerical analysis. *Journal of Computational and Applied Mathematics*, **31**(1), 191–197.

[56] Trask, Nathaniel, Patel, Ravi G, Atzberger, Paul J, and Gross, Ben J. 2020. GMLS-Nets: A Machine Learning Framework for Unstructured Data. *In: AAAI Spring Symposium: MLPS*.

[57] Trottenberg, Ulrich, Oosterlee, Cornelis W., and Schüller, Anton. 2001. *Multigrid*. Texts in Applied Mathematics. Bd., vol. 33. San Diego [u.a.]: Academic Press. With contributions by A. Brandt, P. Oswald and K. Stüben.

[58] Turk, Greg. 1991. Generating textures on arbitrary surfaces using reaction-diffusion. *Comput. Graph.*, **25**(4), 289–298.

[59] Vallis, Geoffrey K. 2006. *Atmospheric and Oceanic Fluid Dynamics: Fundamentals and Large-scale Circulation*. Cambridge University Press.

[60] Wendland, Holger. 2004. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.

[61] Wendland, Holger, and Künemund, Jens. 2020. Solving partial differential equations on (evolving) surfaces with radial basis functions. *Advances in Computational Mathematics*, **46**, 64.

[62] White, P. W. 1971. Finite-Difference Methods in Numerical Weather Prediction. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, **323**(1553), 285–292.

[63] Williamson, David L. 2007a. The Evolution of Dynamical Cores for Global Atmospheric Models. *Journal of the Meteorological Society of Japan 85B*, 241–269.

[64] Williamson, David L. 2007b. The Evolution of Dynamical Cores for Global Atmospheric Models. *J. Meteorol. Soc. Jpn.*, **85B**, 241–269.

[65] Yu, Xi, Wang, Zhiqiang, Jiang, Yugui, and Zhang, Xi. 2006. Surface Gradient Material: From Superhydrophobicity to Superhydrophilicity. *Langmuir*, **22**(10), 4483–4486.

[66] Yuksel, Cem. 2015. Sample Elimination for Generating Poisson Disk Sample Sets. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2015)*, **34**(2), 25–32.

CHAPTER 2:

A COMPARISON OF GENERALIZED MOVING LEAST SQUARES AND RADIAL BASIS FUNCTION FINITE DIFFERENCE METHODS FOR APPROXIMATING SURFACE DERIVATIVES

This manuscript has been submitted to the Springer Journal of Scientific Computing. A complete draft of the manuscript is given in Appendix A.

2.1 Author Contributions

- Andrew M. Jones: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing- Orginal draft preparation, Writing - Review & Editing.
- Grady B. Wright: Methodology, Writing -Orignal draft preparation, Writing- Review & Editing Funding acquistion, Supervision
- Peter A. Bosler: Methodology, Resources, Writing- Review & Editing, Funding acquistion, Supervision
- Paul A. Kuberry: Resources, Software

CHAPTER 3:

MGM: A MESHFREE GEOMETRIC MULTILEVEL METHOD FOR LINEAR SYSTEMS ARISING FROM ELLIPTIC EQUATIONS ON POINT CLOUD SURFACES

This paper has been accepted in the SIAM Journal of Scientific Computing. The accepted version of the paper is given in Appendix B.

3.1 Author Contributions

- Grady B. Wright: Conceptualization, Methodology, Software, Formal Analysis, Investigation, Writing - Original draft preparation, Writing - Review & Editing, Funding Acquision, Supervision
- Andrew M. Jones: Conceptualization, Methodology, Software, Formal Analysis, Investigation, Writing - Original draft preparation, Writing - Review & Editing
- Varun Shankar: Conceptualization, Writing - Review & Editing, Supervision

CHAPTER 4:

SOFTWARE CONTRIBUTION

The code repositories are available on Github.

`https://github.com/AndrewJ3/MGM`

`https://github.com/AndrewJ3/rbftoolkit`

4.1 Summary

One novelty of the software packages is they are written in open source languages and they are some of the first of their kind. MGM will be the first published meshfree multilevel solver and preconditioner for surface PDEs, degenerate problems, and extendable to mesh-based PDE discretizations. RBFToolkit will be the first RBF-FD software package for computing derivatives, solving PDEs, and interpolating on surfaces.

Open-source software is essential for transparency and wider collaboration in scientific research. We present two open source codes for surface interpolation and for discretization and solving of partial differential equations using meshfree methods. The first code is our meshfree geometric multilevel (MGM) solver and preconditioner for surface partial differential equations. This framework is written in Python and utilizes NumPy [**?**], SciPy [**?**], and C++, for the data structures (lists, arrays, k-dimensional tree, sparse matrices), linear algebra (QR, SparseLU), iterative meth-

ods (BiCGStab, GMRES). The second code is RBFToolkit, this code is a refactored version of the test code RBFKokkos [**?**]. This code includes methods for interpolation and evaluation of surface derivatives on point clouds, the first version is available in Python with performance libraries NumPy, SciPy, and, scikit-learn. The C++ version will build on the RBFKokkos framework, by utilizing Kokkos for shared memory parallelism. Additionally, the Trilinos [**?**] packages Teuchos and Tpetra will be included for distributed memory parallelism and sparse linear algebra. For construction of k-dimensional trees and nearest neighbor searches (ball or kNN) we use the NanoFlann [**?**]. MGM will also be integrated into the C++ version of the RBFToolkit as development progresses.

CHAPTER 5:

CONCLUSIONS

We conclude this dissertation with an overview of the findings and accomplishments; we presented a comparison of two meshfree methods for computing surface derivatives (gradient, divergence, Laplacian). We found the formulations of surface differential operators are equivalent when the tangent space is known exactly. Both methods converge at the same asymptopic rates when using the same polynomial degree. RBF-FD generally produces lower errors than GMLS for the same polynomial degree, stnecil size and point cloud resolution. GMLS is more computationally efficient when compariong the cost of building operators, however RBF-FD was more efficient for evaluation cost.

MGM is the first geometric multilevel method for solving ellitpic equations on surfaces. We find it scales nearly independent of point cloud spacing when used as preconditioner. The overall computational complexity is $O(N \log N)$. The method converges more rapidly than PyAMG especially as order of accuracy increases. Iteration count for MGM is mostly constant comparedto solver and preconditioner for elliptic surface PDEs, displayed favorable results compared to the AMG software, PyAMG. Lastly, MGM can solve complicated problems (i.e. nonlinear reaction diffusion).

APPENDIX A:

PAPER 1: A COMPARISON OF GENERALIZED MOVING LEAST SQUARES AND RADIAL BASIS FUNCTION FINITE DIFFERENCE METHODS FOR APPROXIMATING SURFACE DERIVATIVES

Authors: Andrew M. Jones , Peter A. Bosler , Paul A. Kuberry , Grady B. Wright

A.1 Abstract

Approximating differential operators defined on two-dimensional surfaces is an important problem that arises in many areas of science and engineering. Over the past ten years, localized meshfree methods based on generalized moving least squares (GMLS) and radial basis function finite differences (RBF-FD) have been shown to be effective for this task as they can give high orders of accuracy at low computational cost, and they can be applied to surfaces defined only by point clouds. However, there have yet to be any studies that perform a direct comparison of these methods for approximating surface differential operators (SDOs). The purpose of this work is to fill that gap. We focus on RBF-FD methods based on polyharmonic spline (PHS) kernels and polynomials since they are most closely related to the GMLS method. We give a detailed description of both methods, including the different ways that they formulate SDOs. One key finding we make here is that these formulations are equivalent when the tangent space to the surface is known exactly. We numerically examine the convergence rates of the methods for various parameter choices as the discretizations of the surfaces are refined. We also compare their efficiency in terms of accuracy per computation cost.

A.2 Introduction

The problem of approximating differential operators defined on two dimensional surfaces embedded in $\mathbb{R}^3$ arises in many multiphysics models. For example, simulating atmospheric flows with Eulerian or Lagrangian numerical methods requires approximating the surface gradient, divergence, and Laplacian on the two-sphere [38, 42, 15, 8]. Similar surface differential operators (SDOs) must be approximated on more geomet-

rically complex surfaces in models of ice sheet dynamics [17], biochemical signaling on cell membranes [23], morphogenesis [34], texture synthesis [25], and sea-air hydrodynamics [2].

Localized meshfree methods based on generalized moving least squares (GMLS) and radial basis function finite differences (RBF-FD) have become increasingly popular over the last ten years for approximating SDOs and solving surface partial differential equations (PDEs) (see, for example, [24, 22, 36, 35, 18] for GMLS and [1, 21, 14, 31, 30, 32, 33, 29, 43, 19, 41] for RBF-FD methods). This is because they can provide high accuracy at low computational cost and can even be applied to surfaces defined by point clouds, without having to form a triangulation like surface finite element methods [12] or a level-set representation like embedded finite element methods [7]. While there is one study dedicated to comparing these methods for approximating functions and derivatives in $\mathbb{R}^2$ and $\mathbb{R}^3$ [3], there are no studies that compare the methods for approximating SDOs. The present work aims to fill this gap.

The RBF-FD methods referenced above use different approaches for approximating SDOs, while the GMLS methods essentially use the same approach. To keep the comparison to GMLS manageable, we will thus limit our focus to the RBF-FD method based on polyharmonic spline (PHS) kernels augmented with polynomials since they are most closely related to GMLS [3]. Additionally, these RBF-FD methods are becoming more and more prevalent as they can give high orders of accuracy that are controlled by the augmented polynomial degree [5] and they do not require choosing a shape parameter, which can be computationally intensive to do in an automated way. The RBF-FD methods referenced above also use distinct techniques for formulating the SDOs. To again keep this comparison manageable, we limit our

focus to the so-called tangent plane formulation, as it provides a more straightforward technique for incorporating polynomials in the PHS-based RBF-FD methods than [1, 21, 14, 31, 30, 32, 29, 41]. Additionally, the comparison in [33] of several RBF-FD methods for approximating the surface Laplacian (Laplace-Beltrami operator) revealed the tangent plane approach to be the most computationally efficient in terms of accuracy per computational cost. The tangent plane method was first introduced by Demanet [11] for approximating the surface Laplacian using polynomial based approximations. Suchde & Kuhnert [35] generalized this method to other SDOs using polynomial weighted least squares to generate the stencil weights. Shaw [33] (see also [43]) was the first to use this method for approximating the surface Laplacian with RBF-FD and Gunderman et. al. [19] independently developed the method for RBF-FD specialized to the surface gradient and divergence on the unit two-sphere.

The GMLS and RBF-FD methods are similar in that they use weighted combinations of function values over a local stencil of points to approximate SDOs. They also feature a parameter ℓ for controlling the degree of polynomial precision of the formulas. However, they also have several major differences. First, GMLS is based on weighted least squares polynomial approximants, whereas the type of RBF-FD method considered here is based on PHS interpolants augmented with polynomials. Second, for GMLS one has to choose a weight kernel for the least squares problem, while one has to choose the order of the PHS kernel in the RBF-FD method. Third, GMLS uses local coordinates and approximations to metric terms to formulate the SDOs. The RBF-FD method examined in this study, on the other hand, is based on the tangent plane method, which does not explicitly include any metric terms.

In this study, we compare GMLS and RBF-FD for approximating the surface gra-

dient, divergence, and Laplacian operators on two topologically distinct surfaces, the unit two-sphere and the torus, which are representative of a broad range of application domains. We investigate their convergence rates as the sampling density of points on these surfaces increases for various parameter choices, including polynomial degree and stencil sizes. In the case of the sphere, we also study the convergence rates of the methods for different point sets, including two popular ones used in applications: icosahedral and cubed sphere points. Finally, we investigate the efficiency of the methods in terms of their accuracy versus computational cost, both when including and excluding setup costs.

One key result we show analytically is that the local coordinate formulation of SDOs used in GMLS is identical to the formulation of the tangent plane method when the tangent space for the surface is known exactly for the given point cloud. Furthermore, our numerical results demonstrate that RBF-FD and GMLS give similar convergence rates for the same choice of polynomial degree ℓ, but overall RBF-FD results in lower errors. We also show that the often-reported super convergence of GMLS for the surface Laplacian only happens for highly structured, quasi-uniform point sets, and when the point sets are more general (but still possibly quasi-uniform), this convergence rate drops to the theoretical rate. Additionally, we find that the errors for RBF-FD can be further reduced with increasing stencil sizes, but that this does not generally hold for GMLS, and the errors can actually deteriorate. Finally, we find that when setup costs are included, GMLS has an advantage in terms of efficiency, but if these are neglected then RBF-FD is more efficient.

The remainder of the paper is organized as follows. In Section A.3, we provide some background and notation on stencil-based approximations and on surface dif-

ferential operators. We follow this with a detailed overview of the GMLS and RBF methods in Section A.4 and B.3.2, respectively. In Section A.6 we compare the two methods in terms of some of their theoretical properties and in Section B.6 we give an extensive numerical comparison of the methods. We end with some concluding remarks in Section A.8.

A.3 Background and notation

A.3.1 Stencils

The RBF-FD and GMLS methods both discretize SDOs by weighted combinations of function values over a local *stencil* of points. This makes them similar to traditional finite-difference methods, but the lack of a grid, a tuple indexing scheme, and inherent awareness of neighboring points requires that some different notation and concepts be introduced. In this section we review the stencil notation that will be used in the subsequent sections.

Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ be a global set of points (point cloud) contained in some domain Ω. A *stencil of* $\mathbf{X}$ is a subset of $n \leq N$ nodes of $\mathbf{X}$ that are close (see discussion below for what this means) to some point $\mathbf{x}_c \in \Omega$, which is called the *stencil center*. In this work, the stencil center is some point from $\mathbf{X}$, so that $\mathbf{x}_c = \mathbf{x}_i$, for some $1 \leq i \leq N$, and this point is always included in the stencil. We denote the subset of points making up the stencil with stencil center $\mathbf{x}_i$ as $\mathbf{X}^i$ and allow the number of points in the stencil to vary with $\mathbf{x}_i$. To keep track of which points in $\mathbf{X}^i$ belong to $\mathbf{X}$, we use *index set* notation and let σ^i denote the set of indices of the $1 < n_i \leq N$ points from $\mathbf{X}$ that belong to $\mathbf{X}^i$. Using this notation, we write the elements of the stencil as $\mathbf{X}^i = \{\mathbf{x}_j\}_{j \in \sigma^i}$. We also use the convention that the indices are sorted by

the distance the stencil points are from the stencil center $\mathbf{x}_i$, so that the first element of σ^i is i.

With the above notation, we can define a general stencil-based approximation method to a given (scalar) linear differential operator $\mathcal{L}$. Let u be a scalar-valued function defined on Ω that is smooth enough so that $\mathcal{L}u$ is defined for all $\mathbf{x} \in \Omega$. The approximation to $\mathcal{L}u$ at any $\mathbf{x}_i \in \mathbf{X}$ is given as

$$\mathcal{L}u|_{\mathbf{x}=\mathbf{x}_i} \approx \sum_{j \in \sigma^i} c_{ij} u(\mathbf{x}_j). \tag{A.1}$$

The weights c_{ij} are determined by the method of approximation, which in this study will be either GMLS or RBF-FD. These weights can be assembled into a sparse $N \times N$ "stiffness" matrix, similar to mesh-based methods. Vector linear differential operators (e.g., the gradient) can be similarly defined where (A.1) is used for each component and $\mathcal{L}$ is the scalar operator for that component.

There are two main approaches used in the meshfree methods literature for determining the stencil points, one based on k-nearest neighbors (KNN) and one based on ball searches. These are illustrated in Figure A.1 for a scattered point set $\mathbf{X}$ in the plane. The approach that uses KNN is straightforward since it amounts to simply choosing the stencil $\mathbf{X}^i$ as the subset of n_i points from $\mathbf{X}$ that are closest to $\mathbf{x}_i$. The approach that uses ball searches is a bit more involved, so we summarize it in Algorithm 1. Both methods attempt to select points such that the stencil satisfies polynomial unisolvency conditions (see the discussion in Section A.4.1). In this work, we use the method in Algorithm 1 since

- it is better for producing stencils with symmetries when $\mathbf{X}$ is regular, which can

Algorithm 1: Procedure for determining the stencil points based on ball searches.

Input: Point cloud $\mathbf{X}$; stencil center $\mathbf{x}_\mathrm{c}$; number initial stencil points n; radius factor $\tau \geq 1$;
Output: Indices σ^c in $\mathbf{X}$ for the stencil center $\mathbf{x}_\mathrm{c}$;
Find the n nearest neighbors in $\mathbf{X}$ to $\mathbf{x}_c$, using the Euclidean distance;
Compute the max distance $h_{\max}$ between $\mathbf{x}_\mathrm{c}$ and its n nearest neighbors;
Find the indices σ^c of the points in $\mathbf{X}$ contained in the ball of radius $\tau h_{\max}$ centered at $\mathbf{x}_\mathrm{c}$;

be beneficial for improving the accuracy of the approximations;

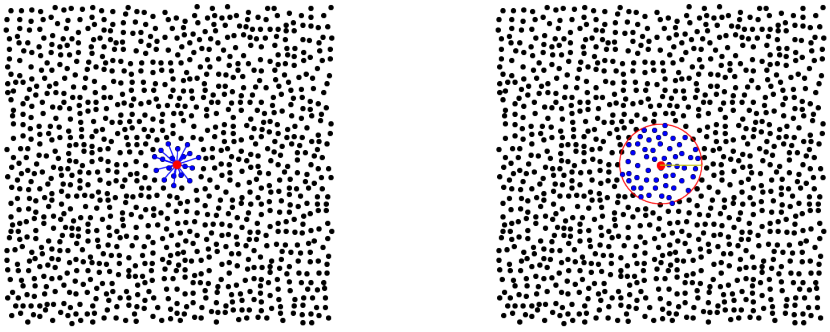
- it is more natural to use with the weighting kernel inherent to GMLS; and
- it produces stencils that are not biased in one direction when the spacing of the points in X are anisotropic.

To measure distance in the ball search, we use the standard Euclidean distance measured in $\mathbb{R}^3$ rather than distance on the surface since this is simple to compute for any surface. We also use a k-d tree to efficiently implement the method. Finally, the choice of parameters we use in Algorithm 1 are discussed in Section A.4.3.

A.3.2 Surface differential operators in local coordinates

Here we review some differential geometry concepts that will be used in the subsequent sections. Much of this material can be found in a general book on this subject, e.g. [39, 28].

We assume that $\mathcal{M} \subset \mathbb{R}^3$ is a regular, orientable surface so that it can be described by an atlas of local smooth charts [39]. Let $T_\mathbf{x}\mathcal{M}$ denote the tangent space to $\mathcal{M}$ at $\mathbf{x} \in \mathcal{M}$. We can express surface differentiable operators in a neighborhood of $\mathbf{x} \in \mathcal{M}$



(a) KNN (b) Ball search

Figure A.1: Comparison of the two search algorithms used in this paper for determining a stencil. The nodes X are marked with solid black disks and all the stencil points are marked with solid blue disks, except for the stencil center, which is marked in red.

using the local chart

$$\mathbf{f} = (\hat{x}, \hat{y}, f(\hat{x}, \hat{y})), \tag{A.2}$$

where $\hat{x}$, $\hat{y}$ are local coordinates for $T_{\mathbf{x}}\mathcal{M}$, and $f(\hat{x}, \hat{y})$ can be interpreted as a function over the $\hat{x}\hat{y}$-plane. This local parametric representation of $\mathcal{M}$ about $\mathbf{x}$ is called the Monge patch or Monge form [28] and is illustrated for a bumpy sphere surface in Figure B.1.

Using this parameterization, the local metric tensor G about $\mathbf{x}$ for the surface is given as

$$G = \begin{bmatrix} \partial_{\hat{x}}\mathbf{f} \cdot \partial_{\hat{x}}\mathbf{f} & \partial_{\hat{x}}\mathbf{f} \cdot \partial_{\hat{y}}\mathbf{f} \\ \partial_{\hat{y}}\mathbf{f} \cdot \partial_{\hat{x}}\mathbf{f} & \partial_{\hat{y}}\mathbf{f} \cdot \partial_{\hat{y}}\mathbf{f} \end{bmatrix} = \begin{bmatrix} 1 + (\partial_{\hat{x}} f)^2 & (\partial_{\hat{x}} f)(\partial_{\hat{y}} f) \\ (\partial_{\hat{x}} f)(\partial_{\hat{y}} f) & 1 + (\partial_{\hat{y}} f)^2 \end{bmatrix}. \tag{A.3}$$

Letting g^{ij} denote the (i,j) entry of G^{-1}, the surface gradient operator locally about $\mathbf{x}$ is given as

$$\widehat{\nabla}_{\mathcal{M}} = (\partial_{\hat{x}}\mathbf{f})\left(g^{11}\partial_{\hat{x}} + g^{12}\partial_{\hat{y}}\right) + (\partial_{\hat{y}}\mathbf{f})\left(g^{21}\partial_{\hat{x}} + g^{22}\partial_{\hat{y}}\right). \tag{A.4}$$

However, this is the surface gradient with respect to the horizontal $\hat{x}\hat{y}$-plane (see Figure B.1 (b)), and subsequently needs to be rotated so that it is with respect to $T_{\mathbf{x}}\mathcal{M}$ in its original configuration. If $\boldsymbol{\xi}^1$ and $\boldsymbol{\xi}^2$ are orthonormal vectors that span $T_{\mathbf{x}}\mathcal{M}$ and $\boldsymbol{\eta}$ is the unit outward normal to $\mathcal{M}$ at $\mathbf{x}$, then the surface gradient in the correct orientation is given as

$$\nabla_{\mathcal{M}} = \underbrace{\begin{bmatrix}\boldsymbol{\xi}^1 & \boldsymbol{\xi}^2 & \boldsymbol{\eta}\end{bmatrix}}_{R} \widehat{\nabla}_{\mathcal{M}}. \tag{A.5}$$

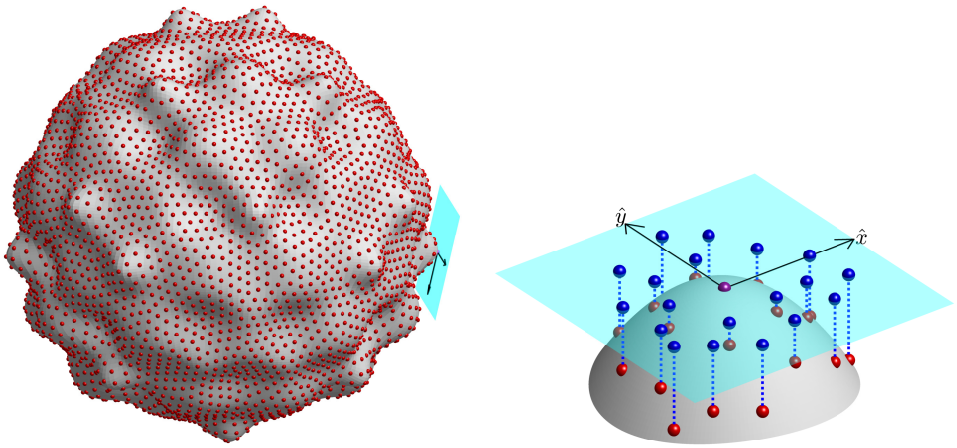
Using this result, the surface divergence of a smooth vector $\mathbf{u} \in T_{\mathbf{x}}\mathcal{M}$ can be written as

$$\nabla_{\mathcal{M}} \cdot \mathbf{u} = \left(g^{11}\partial_{\hat{x}} + g^{12}\partial_{\hat{y}}\right)(\partial_{\hat{x}}\mathbf{f})^T R^T\mathbf{u} + \left(g^{21}\partial_{\hat{x}} + g^{22}\partial_{\hat{y}}\right)(\partial_{\hat{y}}\mathbf{f})^T R^T\mathbf{u} \tag{A.6}$$

The surface Laplacian operator locally about $\mathbf{x}$ is given as

$$\begin{aligned}\Delta_{\mathcal{M}} = \frac{1}{\sqrt{|g|}}\Bigg(&\partial_{\hat{x}}\left(\sqrt{|g|}g^{11}\partial_{\hat{x}}\right) + \partial_{\hat{x}}\left(\sqrt{|g|}g^{12}\partial_{\hat{y}}\right) + \\ &\partial_{\hat{y}}\left(\sqrt{|g|}g^{21}\partial_{\hat{x}}\right) + \partial_{\hat{y}}\left(\sqrt{|g|}g^{22}\partial_{\hat{y}}\right)\Bigg),\end{aligned} \tag{A.7}$$

where $|g| = \det(G)$. This operator is invariant to rotations of the surface in $\mathbb{R}^3$, so



(a) (b)

Figure A.2: Illustration of a Monge patch parameterization for a local neighborhood of a regular surface $\mathcal{M}$ in 3D. (a) Entire surface (in gray) together with the tangent plane (in cyan) for a point $\mathbf{x}_c$ where the Monge patch is constructed (i.e., $T_{\mathbf{x}_c}\mathcal{M}$); red spheres mark a global point cloud $\mathbf{X}$ on the surface. (b) Close-up view of the Monge patch parameterization, together with the points from a stencil $\mathbf{X}_c$ (red spheres) formed from $\mathbf{X}$ and the projection of the stencil to the tangent plane (blue spheres); the stencil center $\mathbf{x}_c$ is at the origin of the axes for the $\hat{x}\hat{y}$-plane and is marked with a violet sphere.

no subsequent modifications of (A.7) are necessary.

A.4 GMLS using local coordinates

The formulation of GMLS on a manifold was introduced by Liang & Zhao [22] and further refined by Trask, Kuberry, and collaborators [36, 18]. It uses local coordinates to approximate SDOs as defined in (A.5)–(A.7) and requires a method to also approximate the metric terms. Both approximations are computed for each $\mathbf{x}_i \in \mathbf{X} \subset \mathcal{M}$ using GMLS over a local stencil of points $\mathbf{X}^i \subset \mathbf{X}$. Below we give a brief overview of the method assuming that the tangent/normal vectors for the surface are known for each $\mathbf{x}_i \in \mathbf{X}$. We then discuss a method for approximating these that is used in the

Compadre Toolkit [20], which we use in the numerical experiments.

We present the GMLS method through the lens of derivatives of MLS approximants as we feel this makes the analog to RBF-FD clearer, it is also closer to the description from [22]. Other derivations of GMLS are based on weighted least squares approximants of general linear functionals given at some set of points, e.g. [40, 27, 37]. However, both techniques produce the same result in the end [27]. For a more thorough discussion of MLS approximants, see for example [13, ch. 22] and the references therein.

A.4.1 Approximating the metric terms

The metric terms are approximated from an MLS reconstruction of the Monge patch of $\mathcal{M}$ centered at each target point $\mathbf{x}_i$ using a local stencil of n_i points $\mathbf{X}^i \subset \mathbf{X}$. This procedure is illustrated in Figure B.1 and can be described as follows. First, the stencil $\mathbf{X}^i$ is expressed in the form of (A.2) (i.e., $(\hat{x}_j, \hat{y}_j, f_j)$, $j \in \sigma^i$), where $(\hat{x}_j, \hat{y}_j)$ are the coordinates for the stencil points in $T_{\mathbf{x}_i}\mathcal{M}$, and $f_j = f(\hat{x}_j, \hat{y}_j)$ are samples of the surface as viewed from the $\hat{x}\hat{y}$-plane. These can be computed explicitly as

$$\begin{bmatrix} \hat{x}_j \\ \hat{y}_j \\ f_j \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{\xi}_i^1 & \boldsymbol{\xi}_i^2 & \boldsymbol{\eta}_i \end{bmatrix}^T}_{R_i^T} (\mathbf{x}_j - \mathbf{x}_i), \tag{A.8}$$

where $\boldsymbol{\xi}_i^1$ and $\boldsymbol{\xi}_i^2$ are orthonormal vectors that span $T_{\mathbf{x}_i}\mathcal{M}$ and $\boldsymbol{\eta}_i$ is the unit normal to $\mathcal{M}$ at $\mathbf{x}_i$. To simplify the notation that follows, we let $\hat{\mathbf{x}}_j = (\hat{x}_j, \hat{y}_j)$ and $\hat{\mathbf{X}}^i = \{\hat{\mathbf{x}}_j\}_{j \in \sigma^i}$ denote the projection of the stencil $\mathbf{X}^i$ to $T_{\mathbf{x}_i}\mathcal{M}$. Note that for convenience in what comes later we have shifted the coordinates so that the center of the projected stencil is $\hat{\mathbf{x}}_i = (0, 0)$.

In the second step, the approximate Monge patch at $\mathbf{x}_i$ is constructed from a MLS approximant of the data $(\hat{\mathbf{x}}_j, f_j)$, $j \in \sigma^i$, which can be written as

$$q(\hat{\mathbf{x}}) = \sum_{k=1}^{L} b_k(\hat{\mathbf{x}}) p_k(\hat{\mathbf{x}}), \tag{A.9}$$

where $\{p_1, \ldots, p_L\}$ is a basis for $\mathbb{P}^2_\ell$ (the space of bivariate polynomials of degree ℓ) and $L = \dim(\mathbb{P}^2_\ell) = (\ell+1)(\ell+2)/2$ is the dimension of this space. The coefficients $b_k(\hat{\mathbf{x}})$ of the approximant are determined from the data according to the weighted least squares problem

$$\underline{b}^*(\hat{\mathbf{x}}) = \underset{\underline{b} \in \mathbb{R}^L}{\operatorname{argmin}} \sum_{j \in \sigma^i} w_\rho(\hat{\mathbf{x}}_j, \hat{\mathbf{x}})(q(\hat{\mathbf{x}}_j) - f_j)^2 = \underset{\underline{b} \in \mathbb{R}^L}{\operatorname{argmin}} \|W_\rho(\hat{\mathbf{x}})^{1/2}(P\underline{b} - \underline{f})\|_2^2, \tag{A.10}$$

where $w_\rho : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^{\geq 0}$ is a weight kernel that depends on a support parameter ρ, $W_\rho(\hat{\mathbf{x}})$ is the $n_i \times n_i$ diagonal matrix $W_\rho(\hat{\mathbf{x}}) = (w_\rho(\hat{\mathbf{x}}_j, \hat{\mathbf{x}}))$, and P is the $n_i \times L$ Vandermonde-type matrix

$$P = \begin{bmatrix} p_1(\hat{\mathbf{x}}_j) & p_2(\hat{\mathbf{x}}_j) & \cdots & p_L(\hat{\mathbf{x}}_j) \end{bmatrix}, \; j \in \sigma^i \tag{A.11}$$

Here we use underlines to denote vectors (i.e., $\underline{b}$ and $\underline{f}$ denote vectors containing coefficients and data from (B.12), respectively). Note that the coefficients b_k depend on $\hat{\mathbf{x}}$ because the kernel w_ρ depends on $\hat{\mathbf{x}}$ (this gives origin to the term "moving" in MLS). We discuss the selection of the stencils and weighting kernel below, but for now it is assumed that $n_i > L$ and $\mathbf{X}^i$ is unisolvent on the space $\mathbb{P}^2_\ell$ (i.e., P is full rank), so that (B.12) has a unique solution.

The MLS approximant q is used in place of f in the Monge patch (A.2) and it is

used to approximate the metric terms in (A.5)–(A.7). To compute these terms, various derivatives need to be approximated at the projected stencil center $\hat{\mathbf{x}}_i$. Considering, for example, $\partial_{\hat{x}} q$, the approximation is computed as follows:

$$\partial_{\hat{x}} q\big|_{\hat{\mathbf{x}}_i} \approx \sum_{k=1}^{L} b_k^*(\hat{\mathbf{x}}_i) \partial_{\hat{x}}(p_k(\hat{\mathbf{x}}))\big|_{\hat{\mathbf{x}}_i}, \tag{A.12}$$

where $b_k^*(\hat{\mathbf{x}}_i)$ come from (B.12) with $\hat{\mathbf{x}} = \hat{\mathbf{x}}_i$. Other derivatives of metric terms in (A.5)–(A.7) are approximated in a similar way to (A.12). We note if the standard monomial basis is used for $\{p_1, \ldots, p_L\}$, then by centering the projected stencil in (A.8) about the origin, only one of the derivatives of p_k in (A.12) is non-zero when evaluated at $\hat{\mathbf{x}}_i$.

Note that (A.12) is only an approximation of $\partial_{\hat{x}} q$ because it does not include the contribution of $\partial_{\hat{x}}(b_k^*(\hat{\mathbf{x}}))\big|_{\hat{\mathbf{x}}_i}$. This approximation is referred to as a "diffuse derivative" in the literature and is equivalent to the GMLS formulation of approximating derivatives [27]. The term "GMLS derivatives" is preferred over "diffuse derivatives" to describe (A.12), since the approximation is not diffuse or uncertain and has the same order of accuracy as the approximations that include the derivatives of the weight kernels [26].

A.4.2 Approximating SDOs

The procedure for approximating any of the SDOs in (A.5)–(A.7) is similar to the one for approximating the metric terms, but for this task we are interested in computing stencil weights as in (A.1) instead of the value of a derivative at a point. Since these SDOs involve computing various partial derivatives with respect to $\hat{x}$ and $\hat{y}$, we can use (A.12) as a starting point for generating these stencil weights. If $\{u_j\}_{j \in \sigma^i}$ are

samples of a function u over the projected stencil $\hat{\mathbf{X}}^i$, then we can again approximate $\partial_{\hat{\mathbf{x}}} u\big|_{\hat{\mathbf{x}}=\hat{\mathbf{x}}_i}$ using (A.12), with $b_k^*(\hat{\mathbf{x}}_i)$ defined in terms of the samples of u. To write this in stencil form we note that (A.12) can be written using vector inner products as

$$\partial_{\hat{x}} u\big|_{\hat{\mathbf{x}}_i} \approx \partial_{\hat{x}} q\big|_{\hat{\mathbf{x}}_i} \approx \underbrace{\left[\partial_{\hat{x}} p_1\big|_{\hat{\mathbf{x}}_i} \quad \cdots \quad \partial_{\hat{x}} p_L\big|_{\hat{\mathbf{x}}_i}\right]}_{(\partial_{\hat{x}} \underline{p}(\hat{\mathbf{x}}_i))^T} \underline{b}^*(\hat{\mathbf{x}}_i) = \underbrace{\left[c_1^i \quad \cdots \quad c_{n^i}\right]}_{(\underline{c}_{\hat{x}}^i)^T} \underline{u}, \tag{A.13}$$

where we have substituted the solution of $\underline{b}^*(\hat{\mathbf{x}}_i)$ in (B.12) to obtain the term in the last equality. This gives the stencil weights $\underline{c}_{\hat{x}}^i$ as the (weighted) least squares solution of the overdetermined system

$$W_\rho(\hat{\mathbf{x}}_i)^{1/2} P \underline{c}_{\hat{x}}^i = W_\rho(\hat{\mathbf{x}}_i)^{1/2} (\partial_{\hat{x}} \underline{p}(\hat{\mathbf{x}}_i)), \tag{A.14}$$

which is typically solved using a QR factorization of $W_\rho(\hat{\mathbf{x}}_i)^{1/2} P$ to promote numerical stability.

Stencil weights $\underline{c}_{\hat{y}}^i$, $\underline{c}_{\hat{x}\hat{x}}^i$, $\underline{c}_{\hat{x}\hat{y}}^i$, and $\underline{c}_{\hat{y}\hat{y}}^i$ for the other derivative operators appearing in (A.5)–(A.7) can be computed in a similar manner for each stencil $\hat{\mathbf{X}}^i$, $i = 1, \ldots, N$. These can then be combined together with the approximate metric terms to define the weights $\{c_{ij}\}$ in (A.1) for any of the SDOs in (A.5)–(A.7).

A.4.3 Choosing the stencils and weight kernel

As discussed in Section A.3.1, we use Algorithm 1 to choose the stencil weights. For the initial stencil size, we use $L = \dim(\mathbb{P}_\ell^2)$. The radius factor τ controls the size of the stencil, with larger τ resulting in larger stencils, and we experiment with this parameter in the numerical results section.

There are many choices for the weight kernel w_ρ in (B.12). Typically, a single

radial kernel is used to define w_ρ as $w_\rho(\mathbf{x}, \mathbf{y}) = w(\|\mathbf{x} - \mathbf{y}\|/\rho)$, where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and $\|\cdot\|$ is the standard Euclidean norm for $\mathbb{R}^d$. In this work, we use the same family of compactly supported radial kernels as [36, 18] and implemented in [20]:

$$w_\rho(\mathbf{x}, \mathbf{y}) = \left(1 - \frac{\|\mathbf{x} - \mathbf{y}\|}{\rho}\right)_+^{2m}, \tag{A.15}$$

where m is a positive integer and $(\cdot)_+$ is the positive floor function. These C^0 kernels have support over the ball of radius ρ centered at $\mathbf{y}$. While smoother kernels can be used such as Gaussians, splines, or Wendland kernels [13], we have not observed any significant improvement in the accuracy of GMLS derivative approximations with smoother kernels. In general, proofs on how the choice of kernels effects the accuracy of GMLS approximations have yet to be found.

Finally, we note that the support parameter ρ is chosen on a per stencil basis and is set equal to τh_{max} from Algorithm 1.

A.4.4 Approximating the tangent space

When the tangent space $T_{\mathbf{x}_i}\mathcal{M}$ is unknown, a coarse approximation to it can be computed for each stencil $\mathbf{X}^i$ using principal component analysis [22]. In this method, one computes the eigenvectors of the covariance matrix $\overline{X}_i\overline{X}_i^T$, where $\overline{X}_i$ is the 3-by-n_i matrix formed from the stencil points $\mathbf{X}^i$ centered about their mean. The two dominant eigenvectors of this matrix are taken as a coarse approximation to $T_{\mathbf{x}_i}\mathcal{M}$ and the third is taken as a coarse approximation to the normal to $\mathcal{M}$ at $\mathbf{x}_i$; we denote these by $\tilde{\boldsymbol{\xi}}_i^1$, $\tilde{\boldsymbol{\xi}}_i^2$, and $\tilde{\boldsymbol{\eta}}_i$, respectively. Next, an approximate Monge patch parameterization is formed with respect to this approximate tangent space using MLS following the same procedure outlined at the beginning of Section A.4.1. This

procedure is illustrated in Figure A.3 (a), where the coarse approximate tangent plane is given in yellow. A refined approximation to the true tangent plane and normal at the stencil center $\mathbf{x}_c$ can be obtained by computing the tangent plane and normal to the MLS approximant of the Monge patch at $\mathbf{x}_c$; this plane is given in cyan in Figure A.3 (a). Once this plane is computed, a new Monge patch parameterization with respect to this refined tangent plane approximation is formed, as illustrated in Figure A.3 (b). This procedure is repeated for each stencil $\mathbf{X}^i$ and the refined tangent space computed for each stencil is used in the procedure described in Section A.4.1 for approximating the metric terms.

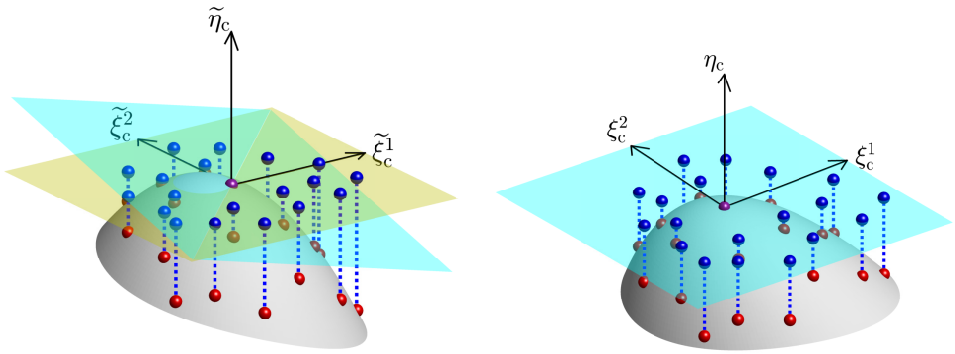


Figure A.3: Illustration of the tangent plane correction method. (a) Monge patch parameterization for a local neighborhood of a regular surface $\mathcal{M}$ (in gray) in 3D using a coarse approximation to the tangent plane (in yellow) at the center of the stencil $\mathbf{x}_c$ and the refined approximation to the tangent plane (in cyan). (b) Same as (a), but for the Monge patch with respect to the refined tangent plane. The red spheres denote the points from the stencil and the blue spheres mark the projection of the stencil to the (a) coarse and (b) refined tangent planes. The coarse and refined approximations to the tangent and normal vectors are given as $\boldsymbol{\xi}_c^1$, $\boldsymbol{\xi}_c^2$, and $\boldsymbol{\eta}_c$, respectively, with tildes on these variables denoting the coarse approximation.

A.5 RBF-FD using the tangent plane

As discussed in the introduction, there are several RBF-FD methods that have been developed over the past ten years for approximating SDOs. We use the one based on the tangent plane method for formulating SDOs and PHS interpolants augmented with polynomials for approximating the derivatives that appear in this formulation. The subsections below provide a detailed overview of these respective techniques.

A.5.1 Tangent plane method

The tangent plane method similarly uses local coordinates for the surface in the tangent plane formed at each $\mathbf{x}_i \in \mathbf{X}$, but unlike the method from Section A.4.1, it does not use approximations to the metric terms. It instead approximates the SDOs at each $\mathbf{x}_i$ using the standard definitions for the derivatives in the tangent plane. So, using local coordinates (A.2) about $\mathbf{x}_i$, the the surface gradient for the tangent plane method is taken as

$$\nabla_{\mathcal{M}} = R_i \begin{bmatrix} \partial_{\hat{x}} \\ \partial_{\hat{y}} \\ 0 \end{bmatrix}, \tag{A.16}$$

and the surface divergence of a smooth vector $\mathbf{u} \in T_{\mathbf{x}_i}\mathcal{M}$ is taken as

$$\nabla_{\mathcal{M}} \cdot \mathbf{u} = \begin{bmatrix} \partial_{\hat{x}} & \partial_{\hat{y}} & 0 \end{bmatrix} R_i^T \mathbf{u}, \tag{A.17}$$

where R_i is the rotation matrix given in (A.8). Similarly, the surface Laplacian in the tangent plane method is

$$\Delta_{\mathcal{M}} = \partial_{\hat{x}\hat{x}} + \partial_{\hat{y}\hat{y}}. \tag{A.18}$$

We next show that if $T_{\mathbf{x}_i}\mathcal{M}$ is known exactly for each $\mathbf{x}_i \in \mathbf{X}$ and the point at which the SDOs are evaluated is $\mathbf{x}_i$, then the SDOs (A.16)–(A.18) are equivalent to the corresponding ones involving metric terms (A.5)–(A.7). This was shown indirectly in [11] for the surface Laplacian using the distributional definition of the surface Laplacian. Here we show the result follows explicitly for each surface differential operator (A.5)–(A.7) from the local coordinate formulation in Section A.3.2.

The first step is to note that the vectors $\partial_{\hat{x}}\mathbf{f}\big|_{\hat{\mathbf{x}}_i}$ and $\partial_{\hat{y}}\mathbf{f}\big|_{\hat{\mathbf{x}}_i}$ from the Monge parameterization (A.2) are tangential to the $\hat{x}\hat{y}$-plane and must therefore be orthogonal to the vector $\begin{bmatrix}0 & 0 & 1\end{bmatrix}$. This implies $\partial_{\hat{x}} f = \partial_{\hat{y}} f = 0$ at $\hat{\mathbf{x}}_i$, which means the metric tensor (A.3) reduces to the identity matrix when evaluated at $\hat{\mathbf{x}}_i$. Using this result in (A.5) for $\widehat{\nabla}_{\mathcal{M}}$ means that the surface gradient formula (A.5) is exactly (A.16) when evaluated at $\hat{\mathbf{x}}_i$. The equivalence of the surface divergence formulas (A.6) and (A.17) also follow immediately from this result.

The steps for showing the equivalence of the surface Laplacian operator are more involved. To simplify the notation in showing this result, we denote partial derivatives of f with subscripts. For the first step of this process, we substitute the explicit metric terms, $|g| = (1 + f_{\hat{x}}^2)(1 + f_{\hat{y}}^2) - (f_{\hat{x}} f_{\hat{y}})^2$, $g^{11} = (1 + f_{\hat{y}})/|g|$, $g^{12} = g^{21} = -(f_{\hat{x}})(f_{\hat{y}})/|g|$, and $g^{22} = (1 + f_{\hat{x}})/|g|$, into (A.7) and expand the derivatives. Next, we simplify to obtain the following formula:

$$\begin{aligned}\Delta_{\mathcal{M}} = \frac{1}{\left(f_{\hat{x}}^2 + f_{\hat{y}}^2 + 1\right)^2}\Big(&\left(f_{\hat{y}} f_{\hat{x}\hat{y}}\left(1 + 2f_{\hat{x}}^2 + f_{\hat{y}}^2\right) - (f_{\hat{x}} f_{\hat{x}\hat{x}} + f_{\hat{y}} f_{\hat{x}\hat{y}})(1 + f_{\hat{y}}^2) - f_{\hat{x}} f_{\hat{y}\hat{y}}(1 + f_{\hat{x}}^2)\right)\partial_{\hat{x}} + \\ &\left(f_{\hat{x}} f_{\hat{x}\hat{y}}\left(1 + 2f_{\hat{y}}^2 + f_{\hat{x}}^2\right) - (f_{\hat{y}} f_{\hat{y}\hat{y}} + f_{\hat{x}} f_{\hat{x}\hat{y}})(1 + f_{\hat{x}}^2) - f_{\hat{y}} f_{\hat{x}\hat{x}}(1 + f_{\hat{y}}^2)\right)\partial_{\hat{y}}\Big) + \\ &g^{11}\partial_{\hat{x}\hat{x}} + 2g^{12}\partial_{\hat{x}\hat{y}} + g^{22}\partial_{\hat{y}\hat{y}}\end{aligned}$$

Using $f_{\hat{x}} = f_{\hat{y}} = g^{12} = 0$ and $g^{11} = g^{22} = 1$ at $\hat{\mathbf{x}}_i$, this formula reduces to (A.18).

A.5.2 Approximating the SDOs

Since the tangent plane method does not require computing approximations to any metric terms, we only need to describe the RBF-FD method for approximating the derivatives that appear in (A.16)–(A.18). We derive this method from derivatives of interpolants over the projected stencils for each point $\mathbf{x}_i \in X$ using the same notation as Section A.4 and we assume that the tangent space is known. A method for approximating the tangent space also using RBF-FD is discussed in Section A.5.4.

Let $\{u_j\}_{j\in\sigma^i}$ be samples of some function u over the projected stencil $\hat{\mathbf{X}}^i = \{\hat{\mathbf{x}}_j\}_{j\in\sigma^i}$. The PHS interpolant to this data can be written

$$s(\hat{\mathbf{x}}) = \sum_{j=1}^{n_i} a_j \phi(\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_{\sigma_j^i}\|) + \sum_{k=1}^{L} b_k p_k(\hat{\mathbf{x}}), \tag{A.19}$$

where $\phi(r) = r^{2\kappa+1}$ is the PHS kernel of order $2\kappa+1$, $\kappa \in \mathbb{Z}^{\geq 0}$, σ_j^i is the jth index in σ^i, $\|\cdot\|$ denotes the Euclidean norm, and $\{p_1, \ldots, p_L\}$ are a basis for $\mathbb{P}_\ell^2$. The expansion coefficients are determined by the n_i interpolation conditions and L additional moment conditions:

$$s(\hat{\mathbf{x}}_j) = u_j,\ j \in \sigma^i \quad \text{and} \quad \sum_{j=1}^{n_i} a_j p_k(\hat{\mathbf{x}}_{\sigma_j^i}) = 0,\ k = 1, \ldots, L. \tag{A.20}$$

These conditions can be written as the following $(n_i + L) \times (n_i + L)$ linear system

$$\begin{bmatrix} A & P \\ P^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} = \begin{bmatrix} \underline{u} \\ \underline{0} \end{bmatrix}, \tag{A.21}$$

where $A_{jk} = \|\hat{\mathbf{x}}_{\sigma_j^i} - \hat{\mathbf{x}}_{\sigma_k^i}\|^{2\kappa+1}$ $(j,k = 1,\ldots,n_i)$ and P is the same Vandermonde-type matrix given in (A.11). The PHS parameter κ controls the smoothness of the kernel and should be chosen such that $0 \leq \kappa \leq \ell$. With this restriction on κ, it can be shown that A is positive definite on the subspace of vectors in $\mathbb{R}^n$ satisfying the L moment conditions in (B.8) [40]. Hence, if the stencil points $\mathbf{X}^i$ are such that $\text{rank}(P) = L$ (i.e., they are unisolvent on the space $\mathbb{P}_\ell^2$), then the system (B.9) is non-singular and the PHS interpolant is well-posed. Note that this is the same restriction on $\mathbf{X}^i$ for the MLS problem (B.12) to have a unique solution.

The stencil weights for approximating any of the derivatives appearing in the SDOs (A.16)-(A.18) can be obtained from differentiating the PHS interpolant (B.7). Without loss of generality, consider approximating the operator $\partial_{\hat{x}}$ over the stencil $\hat{\mathbf{X}}^i$. Using vector inner products as in (A.13), the stencil weights for this operator are determined from the approximation

$$\partial_{\hat{x}} u\big|_{\hat{\mathbf{x}}_i} \approx \partial_{\hat{x}} s\big|_{\hat{\mathbf{x}}_i} = \begin{bmatrix} \partial_{\hat{x}}\underline{\phi}(\hat{\mathbf{x}}_i) & \partial_{\hat{x}}\underline{p}(\hat{\mathbf{x}}_i) \end{bmatrix}^T \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix}.$$

where $\partial_{\hat{x}}\underline{\phi}(\hat{\mathbf{x}}_i)$ and $\partial_{\hat{x}}\underline{p}(\hat{\mathbf{x}}_i)$ are vectors containing the entries $\partial_{\hat{x}}\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_{\sigma_j^i}\|^{2\kappa+1}\big|_{\hat{\mathbf{x}}_i}$, $j = 1,\ldots,n_i$, and $\partial_{\hat{x}} p_k(\hat{\mathbf{x}})\big|_{\hat{\mathbf{x}}_i}$, $k = 1,\ldots,L$, respectively. Using (B.9) in the preceding expression gives the stencil weights as the solution to the following linear system

$$\begin{bmatrix} A & P \\ P^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \underline{c}_{\hat{x}}^i \\ \underline{\lambda} \end{bmatrix} = \begin{bmatrix} \partial_{\hat{x}}\underline{\phi}(\hat{\mathbf{x}}_i) \\ \partial_{\hat{x}}\underline{p}(\hat{\mathbf{x}}_i) \end{bmatrix}, \tag{A.22}$$

where the entries in $\underline{\lambda}$ are not used as part of the weights.

Stencil weights $\underline{c}_{\hat{y}}^i$, $\underline{c}_{\hat{x}\hat{x}}^i$, $\underline{c}_{\hat{x}\hat{y}}^i$, and $\underline{c}_{\hat{y}\hat{y}}^i$ for the other partial derivatives can be com-

puted in an analogous way for each stencil $\hat{\mathbf{X}}^i$, $i = 1, \ldots, N$. These can then be combined together to define the weights $\{c_{ij}\}$ in (A.1) for any of the SDOs in (A.16)–(A.18).

A.5.3 Choosing the stencils and PHS order

Similar to GMLS, we use Algorithm 1 to choose the stencils and also use the same initial stencil size of $n = L$ for this algorithm. The parameter κ used to determine the PHS order should be chosen with an upper bound of $\kappa \leq \ell$ (so that (B.10) is well posed) and a lower bound such that the derivatives of the PHS kernels make sense for whatever operator the RBF-FD stencils are being used to approximate. In this work we use $\kappa = \ell$ as we have found that this choice works well for approximating various SDOs across a wide range of surfaces. Choosing $\kappa < \ell$ can be useful for improving the conditioning of the system (B.10) and for reducing Runge Phenomenon-type edge effects in RBF-FD approximations near boundaries [6].

A.5.4 Approximating the tangent space

If $T_{\mathbf{x}_i}\mathcal{M}$ is unknown for any $\mathbf{x}_i \in \mathbf{X}$, then we use a similar procedure to the one discussed for GMLS in Section A.4.4 (and illustrated in Figure A.3) to approximate it. The difference for RBF-FD is that instead of using an MLS reconstruction of the Monge patch parameterization formed from the coarse tangent plane approximation at each $\mathbf{x}_i$, we use the PHS interpolant (B.7) for the reconstruction. The refined approximation to the tangent plane at each $\mathbf{x}_i$ is then obtained from derivatives of the PHS interpolant of the Monge patch for stencil $\mathbf{X}^i$. We note that this approach is new amoungst the different tangent plane methods, as previous approaches assumed the tangent space was computed by some other, possibly unrelated techniques, and not directly from the stencils [35, 33, 43].

A.6 Theoretical comparison of GMLS and RBF-FD

In this section, we make comparisons of the GMLS and RBF-FD methods in terms of some of their theoretical properties, including the different approaches in formulating SDOs, the parameters of the approximations, and the computational cost.

One of the main differences between the GMLS and RBF-FD approaches is that the former uses the local coordinate method to formulate SDOs, while the latter uses the tangent plane method. As shown in Section B.3.1 these methods are equivalent if the tangent space for $\mathcal{M}$ is known for each $\mathbf{x}_i \in X$ and the SDOs are evaluated at the stencil center $\mathbf{x}_i$. However, the GMLS method does not take advantage of this and instead includes metric terms in the formulation. These metric terms are approximated with the same order of accuracy as the GMLS approximation of the derivatives (see below), so that these errors are asymptotically equivalent as the spacing of the points in the stencil goes to zero. When the tangent space is unknown, both methods again approximate it to the same order of accuracy as their respective approximations of the derivatives.

The GMLS and RBF-FD methods each feature the parameter ℓ, which controls the degree of the polynomials used in the approximation. For a given ℓ, the formulas for either method are exact for all bivariate polynomials of degree ℓ in the tangent plane formed by the stencil center $\mathbf{x}_i$. Unsurprisingly, ℓ also effects the local accuracy of the formulas in the tangent plane with increasing ℓ giving higher orders of accuracy for smooth problems; see [26, 22] for a study of the accuracy of GMLS and [10, 4] for RBF-FD. The order of accuracy of both methods depends on the highest order

derivative appearing in the SDOs, and is generally ℓ if the derivative order is 1 and $\ell - 1$ if the derivative order is two. However, for certain quasi-uniform point clouds with symmetries, the order has been shown to be ℓ for GMLS applied to second order operators like the surface Laplacian [22].

The computational cost of the methods can be split between the setup cost and the evaluation cost. The setup cost depends on ℓ and n_i (which depends on τ). For each stencil $\mathbf{X}^i$, the dominant setup cost of GMLS comes from solving the $n_i \times L$ system (B.13), while the dominant cost for RBF-FD comes from solving the $(n_i + L) \times (n_i + L)$ system (B.10). We use QR factorization to solve the GMLS system and LU factorization to solve the RBF-FD system, which gives the following (to leading order):

$$\text{Setup cost GMLS} \sim 2\sum_{i=1}^{N} n_i L^2 \quad \text{and} \quad \text{Setup cost RBF-FD} \sim \frac{2}{3}\sum_{i=1}^{N}(n_i + L)^3. \tag{A.23}$$

The stencil sizes depend on ℓ and τ, and for quasi-uniform point clouds $\mathbf{X}$, n_i is typically some multiple γ of L. In this case, the setup cost of RBF-FD is higher by approximately $\frac{1}{3}(1 + \gamma)^3$. We note that the setup procedure for both methods is an embarrassingly parallel process, as each set of stencil weights can be computed independently of every other set. The evaluation costs of both methods are the same and can be reduced to doing sparse matrix-vector products. So, for a scalar SDO like the surface Laplacian

$$\text{evaluation cost GMLS \& RBF-FD: } \sim 2\sum_{i=1}^{N} n_i. \tag{A.24}$$

If the ℓ and τ parameters remain fixed so that size of the stencils remain fixed as N increases, then both the setup and evaluation cost are linear in N.

A.7 Numerical comparison of GMLS and RBF-FD

We perform a number of numerical experiments comparing GMLS and RBF-FD for approximating the gradient, divergence, and Laplacian on two topologically distinct surfaces: the unit two sphere $\mathbb{S}^2$ and the torus defined implicitly as

$$\mathbb{T}^2 = \left\{ (x, y, z) \in \mathbb{R}^3 \,\middle|\, (1 - \sqrt{x^2 + y^2})^2 + z^2 - 1/9 = 0 \right\}. \tag{A.25}$$

For the experiments with the sphere, we consider three different point sets $\mathbf{X}$: icosahedral, cubed-sphere and Poisson disk points. The first two are commonly used in numerical weather prediction [8, 42] and have been used in other studies on GMLS [36] and RBF-FD [14] methods on the sphere. Unlike these first two point sets, Poisson disk points can be generalized to other surfaces than the sphere and have also previously been used in studies on GMLS and RBF-FD methods [43]. Hence, we use Poisson disk points also for the torus and use the weighted sample elimination (WSE) algorithm [44] to generate them.

All of point sets we consider are quasi-uniform in the sense that the average spacing between the points h (or more generally the mesh-norm [13]) decreases like $h \sim N^{-1/2}$. Therefore, these points sets are well-suited for testing convergence rates of GMLS and RBF-FD methods with N (i.e., convergence as the density of the sampling of the surfaces increases). Table A.1 lists the different values of N used for each point set. We experimentally examine the algebraic convergence rates β versus the $\sqrt{N}$,

assuming the error behaves like $\mathcal{O}(N^{-\beta/2})$. We include convergence rates results for polynomial degrees $\ell = 2$, 4, and 6.

Node set	Values of N
Icosahedral	10242, 40962, 163842, 655362
Cubed sphere	6146, 24578, 98306, 393218
Poisson disk, sphere	8192, 32768, 131072, 524288
Poisson disk, torus	8153, 32615, 130463, 521855

Table A.1: Sizes of the node sets used in the numerical experiments.

All RBF-FD results that follow were obtained from a Python implementation of the method that only utilizes the scientific computing libraries SciPy and NumPy. For the GMLS results, we use the software package Compadre [20], which is implemented in C++ and uses the portable performance library Kokkos.

A.7.1 Convergence comparison: Sphere

We base all the convergence comparisons for the sphere on the following function consisting of a random linear combination of translates of 50 Gaussians of different widths on the sphere:

$$u(\mathbf{x}) = \sum_{j=1}^{50} d_j \exp(-\gamma_j \|\mathbf{x} - \mathbf{y}_j\|^2), \ \mathbf{x}, \mathbf{y}_j \in \mathbb{S}^2, \tag{A.26}$$

where $\mathbf{y}_j$ are the centers and are selected as low discrepancy points on the sphere [9], and d_j & γ_j are sampled from the normal distributions $\mathcal{N}(0,1)$ & $\mathcal{N}(15,4)$, respectively. This function has also been used in other studies on RBF-FD methods [21]. We use samples of u in the surface gradient tests and measure the error against the exact surface gradient, which can be computed using the Cartesian gradient ∇ in $\mathbb{R}^3$ as $\nabla_{\mathcal{M}} u = \nabla u - \boldsymbol{\eta}(\boldsymbol{\eta} \cdot \nabla u)$, where $\boldsymbol{\eta}$ is the unit outward normal to $\mathbb{S}^2$ [16] (which is

just $\mathbf{x}$). Applying this to (A.26) gives

$$\nabla_{\mathcal{M}} u = 2 \sum_{j=1}^{50} d_j \gamma_j (\mathbf{y}_j - \mathbf{x}(\mathbf{x} \cdot \mathbf{y}_j)) \exp(-\gamma_j \|\mathbf{x} - \mathbf{y}_j\|^2). \tag{A.27}$$

We use samples of this field in the surface divergence tests. Since $\nabla_{\mathcal{M}} \cdot \nabla_{\mathcal{M}} u = \Delta_{\mathcal{M}} u$, we compare the errors in this test against the exact surface Laplacian of u, which can be computed using the results of [15] as

$$\Delta_{\mathcal{M}} u = -\sum_{j=1}^{50} d_j \gamma_j (4 - \|\mathbf{x} - \mathbf{y}_j\|^2 (2 + \gamma_j (4 - \|\mathbf{x} - \mathbf{y}_j\|^2))) \exp(-\gamma_j \|\mathbf{x} - \mathbf{y}_j\|^2).$$

We also use this in the tests of the surface Laplacian using samples of u.

For all these tests, we set radius factor τ in the stencil selection Algorithm 1 to 1.5. This gave the best results for GMLS (see the next section for some results on the effects of increasing τ). While the exact tangent space for the sphere is trivially determined, we approximate it in all the results using the methods discussed in the Section A.4.4 for GMLS and Section A.5.4 for RBF-FD. These approximations are done with the same parameters for approximating the different SDOs to keep the asymptotic orders of accuracy comparable. Although not included here, we did experiments with the exact tangent space and obtained similar results to those presented here.

Figures A.4—A.6 display the convergence results for GMLS and RBF-FD as a function of N. Each figure is for a different point set type and contains the results for approximating the surface gradient, divergence, and Laplacian in both the relative two- and max-norms and for different polynomial degrees ℓ. We see from all the figures that the measured convergence rates for GMLS and RBF-FD are similar, but that RBF-FD gives lower errors for the same N and ℓ in almost all cases. One

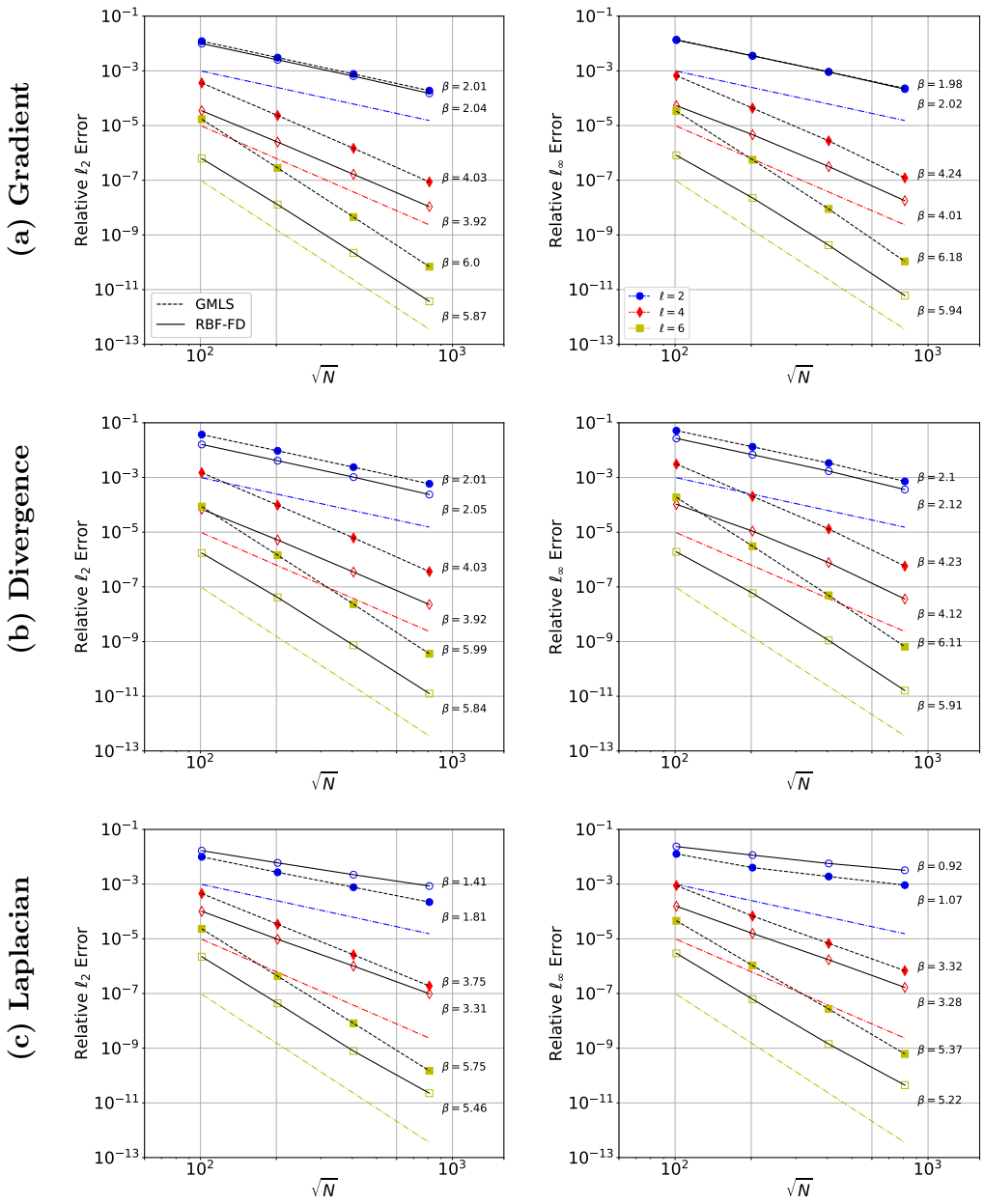


Figure A.4: Convergence results for (a) surface gradient, (b) divergence, and (b) Laplacian on the sphere using icosahedral point sets. Errors are given in relative two-norms (first column) and max-norms (second column). Markers correspond to different ℓ: filled markers are GMLS and open markers are RBF-FD. Dash-dotted lines without markers correspond to 2nd, 4th, and 6th order convergence with $1/\sqrt{N}$. β are the measured order of accuracy computed using the lines of best fit to the last three reported errors.

notable exception is on the cubed sphere points, where the convergence rates of the max-norm errors for the GMLS approximations of the Laplacian are noticeably larger. The results in the figure also show that the measured convergence rates in the two-norm for the surface gradient and divergence approximations are close to the expected rates of ℓ for all the point sets. However, when looking at the convergence rates of the surface Laplacian, we see from Figures A.4 & A.5 that the icosahedral and cubed sphere points have higher rates than for the Poisson disk points in Figure A.6. These improved convergence rates have been referred to as superconvergence in the GMLS literature and rely on the point set being structured so that the stencils have certain symmetries [22]. When these symmetries do not exist, as is the case for the Poisson disk points, then the convergence rates for the surface Laplacian follow more closely the expected rates of $\ell - 1$.

A.7.2 Convergence comparison: Torus

The convergence comparisons on the torus are based on the target function

$$u(\mathbf{x}) = \frac{x}{8}(x^4 - 10x^2y^2 + 5y^4)(r^2 - 60z^2),\ \mathbf{x} \in \mathbb{T}^2, \tag{A.28}$$

where $r = \sqrt{x^2 + y^2}$. This function has also been used in other studies of RBF methods for surfaces [16]. As with the sphere example, the surface gradient of u can be computed as $\nabla_{\mathcal{M}} u = \nabla u - \boldsymbol{\eta}(\boldsymbol{\eta} \cdot \nabla u)$, where $\boldsymbol{\eta}$ is the unit outward normal to $\mathbb{T}^2$, which can be computed from the implicit equation (A.25). The surface Laplacian of (A.28) is given in [16] as

$$\Delta_{\mathcal{M}} u(\mathbf{x}) = -\frac{3x}{8r^2}(x^4 - 10x^2y^2 + 5y^4)(10248r^4 - 34335r^3 + 41359r^2 - 21320r + 4000),\ \mathbf{x} \in \mathbb{T}^2.$$

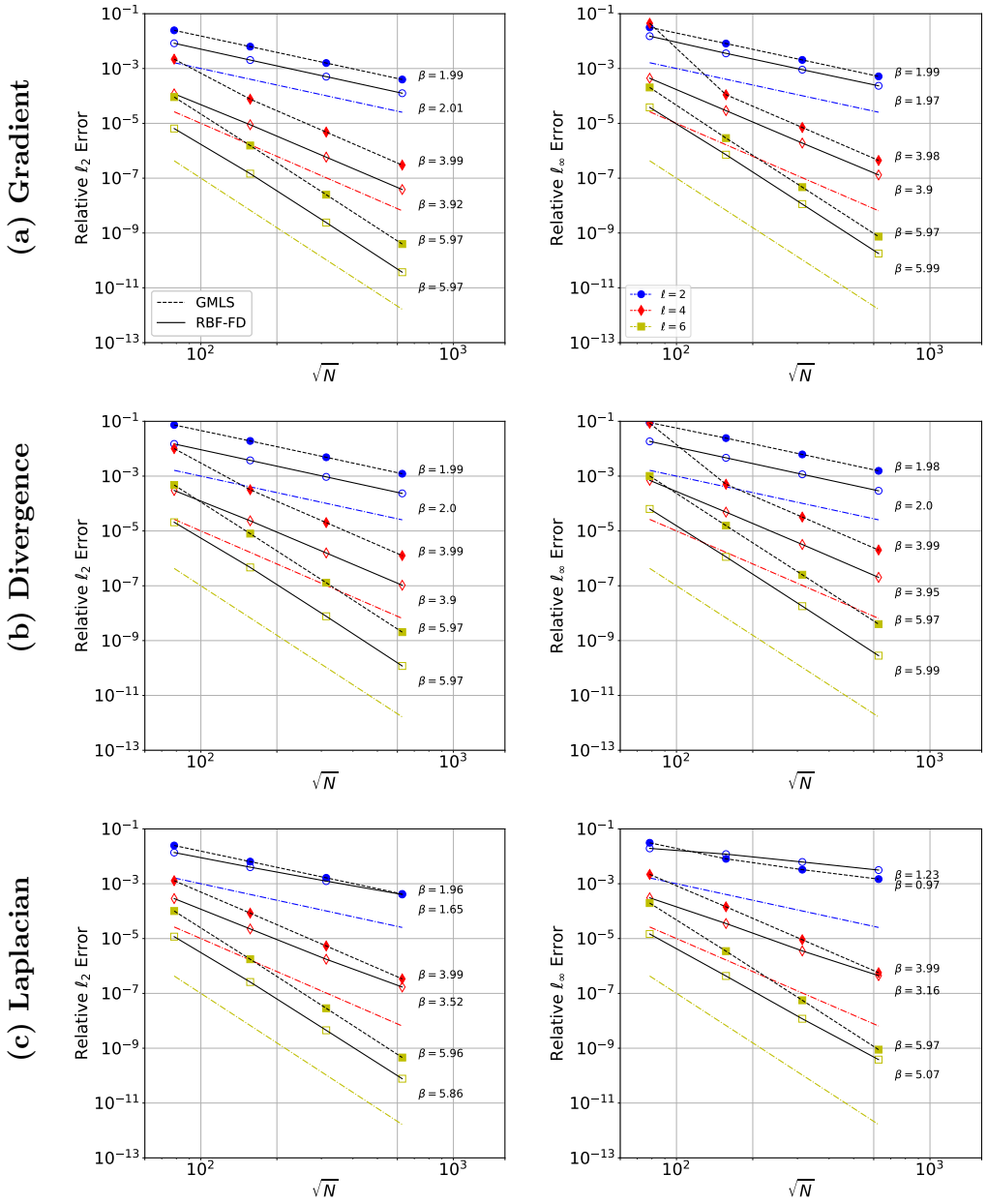


Figure A.5: Same as Figure A.4, but for the cubed sphere points.

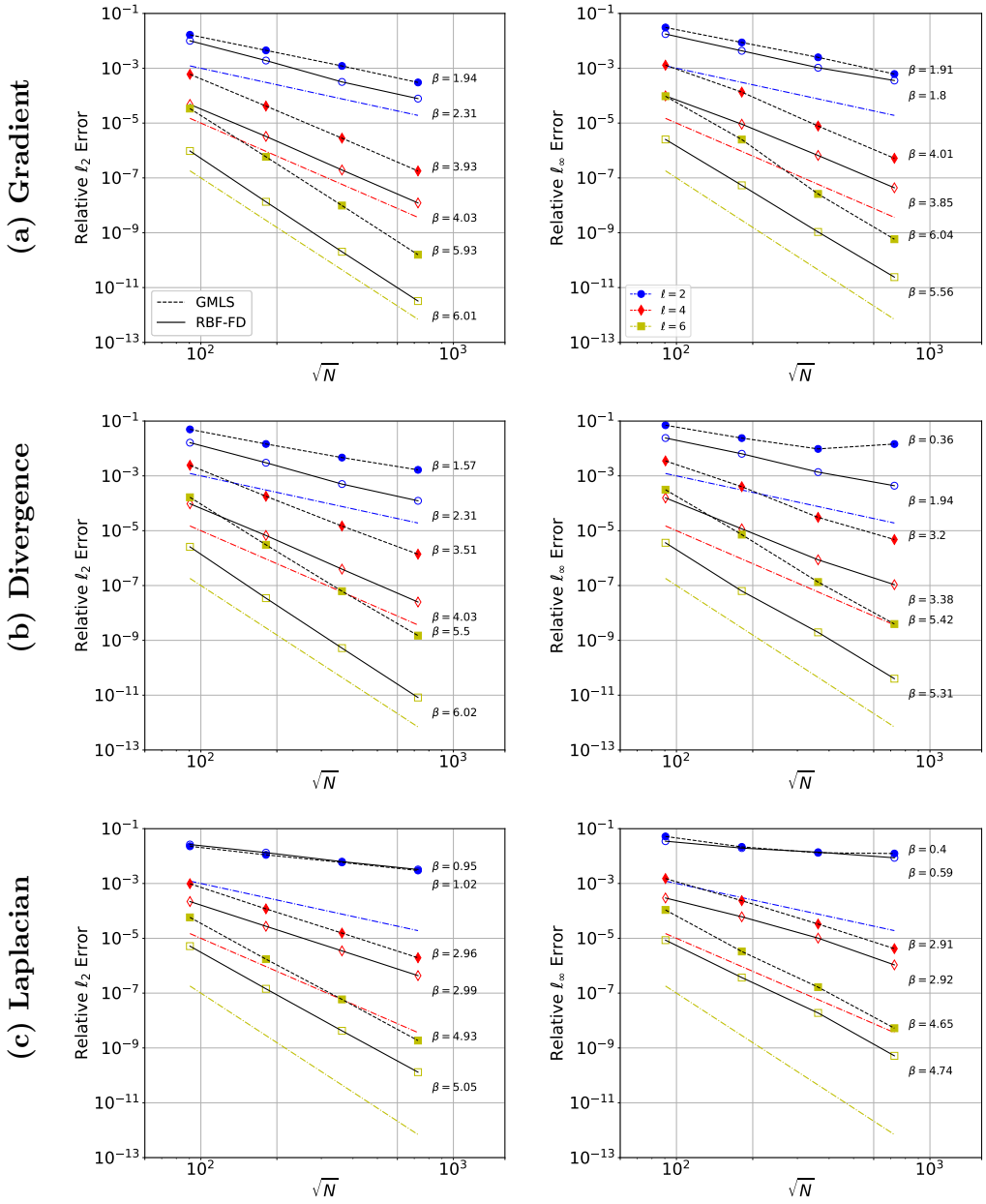


Figure A.6: Same as Figure A.4, but for the Poisson disk points on the sphere.

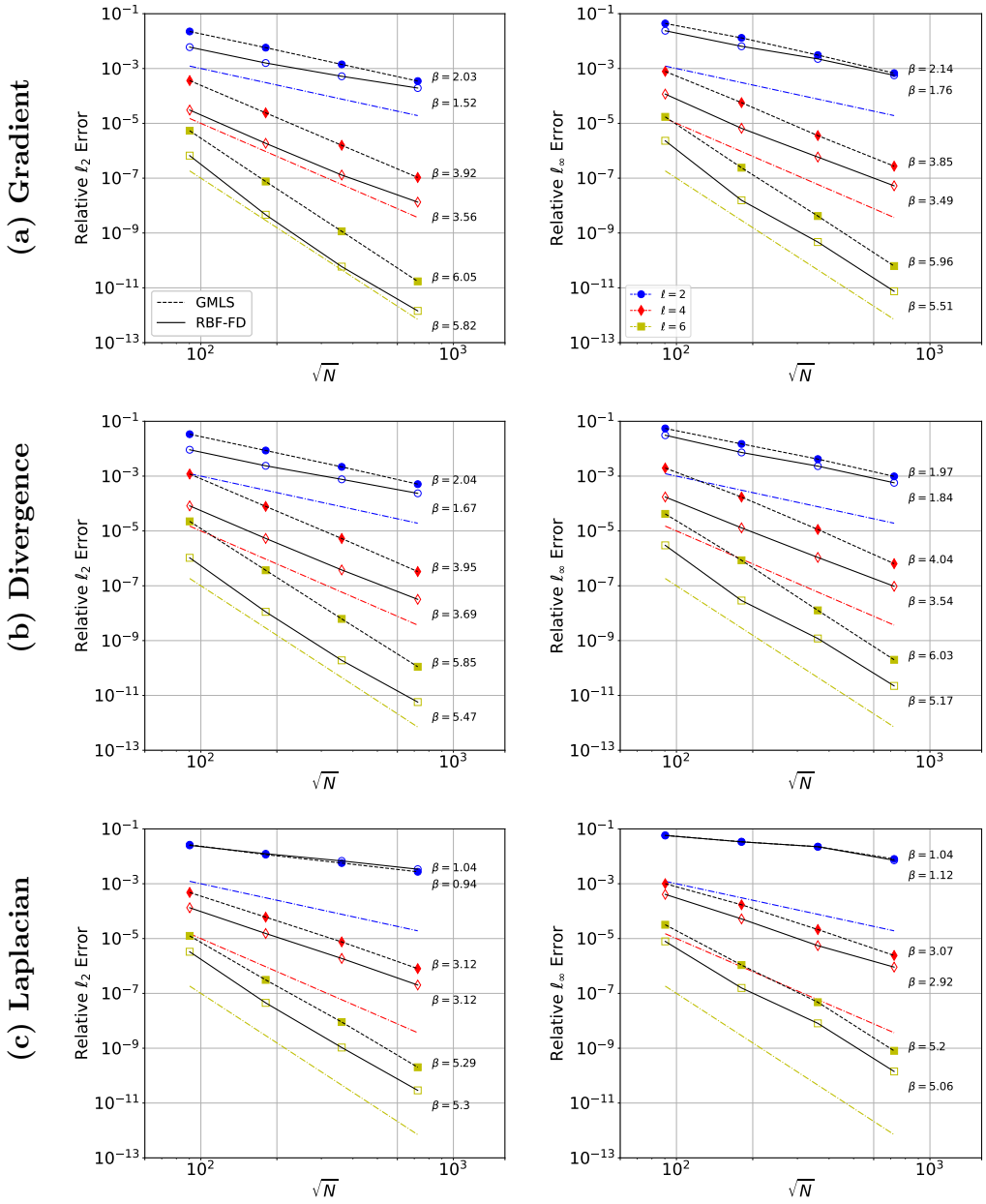


Figure A.7: Same as Figure A.4, but for torus using Poisson disk points.

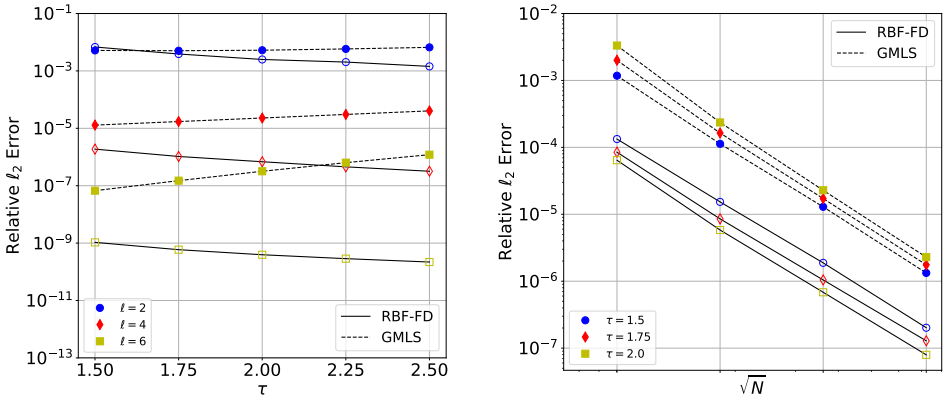


Figure A.8: Relative two-norm errors of the surface Laplacian approximations as the stencil radius parameter τ varies. Left figure shows errors for several different values of τ and a fixed $N = 130463$. Right figure shows the convergence rates of the methods for different τ and a fixed $\ell = 4$.

Similar to the sphere, we use samples of $\nabla_{\mathcal{M}} u$ in the tests of the divergence and compare the results with $\Delta_{\mathcal{M}} u$ above.

We first study the convergence rates with the stencil radius scaling $\tau = 1.5$ and approximate the tangent space, as we did with the sphere tests. Figure A.7 displays the results for the surface gradient, divergence, and Laplacian. We see that errors for RBF-FD are again smaller than the errors for GMLS in almost all cases over the range of N tested. However, GMLS has a slightly higher convergence rates in the case of the surface gradient and divergence, but not for the Laplacian. Both methods have convergence rates that are close to the expected rates of ℓ for these surface gradient and divergence and $\ell - 1$ for the Laplacian.

Next we investigate how the approximation properties of the two methods change when τ is increased, which results in larger stencil sizes. We focus on approximating the surface Laplacian as similar results were found for the other SDOs. In the left plot of Figure A.8, we show the relative two-norm errors of the approximations for a

fixed N as τ varies from 1.5 to 2.5. We see that increasing τ has opposite effects on the two methods: the errors decrease for RBF-FD and increase with GMLS. We see similar results in the right plot of Figure A.8, where we show the convergence of the methods with increasing N for different fixed values of τ (and ℓ fixed at 4). While the convergence rates do not appear to change with τ, the overall errors decrease for RBF-FD and increase for GMLS.

These results make sense when one considers the different types of approximations the methods are based on: RBF-FD is based on interpolation, while GMLS is based on least squares approximation. As the stencil sizes increase, RBF-FD has a larger approximation space consisting of more shifts of PHS kernels, which can reduce the errors [10]. However, GMLS has the same fixed approximation space of polynomials of degree ℓ regardless of the stencil size. This gives another parameter that can be tuned for RBF-FD to give better results for the same fixed polynomial degree.

Finally, we compare the errors when the exact and approximate tangent spaces are used in the two methods. We focus only on the surface Laplacian and for $\ell = 4$ since similar results were obtained for the other operators and other ℓ. Table A.2 shows the results for both methods. The approximate tangent spaces were computed using the methods from Sections A.4.4 (GMLS) and A.5.4 (RBF-FD) also using the polynomial degree $\ell = 4$. We see from the table that the differences between using the exact or the approximate tangent spaces is minor.

	GMLS		RBF-FD	
N	Exact	Approx.	Exact	Approx.
8153	4.7984e-04	4.8004e-04	1.3311e-04	1.3312e-04
32615	6.0457e-05	6.04654e-05	1.5321e-05	1.5322e-05
130463	7.5486e-06	7.5488e-06	1.8811e-06	1.8811e-06
521855	8.0158e-07	8.0159e-07	2.0177e-07	2.0176e-07

Table A.2: Comparison of the relative ℓ_2 errors for the surface Laplacian on the torus using the exact tangent space for the torus and approximations to it based on the methods from Sections A.4.4 (GMLS) and A.5.4 (RBF-FD). In all cases, $\ell = 4$ and the points are based on Poisson disk sampling.

A.7.3 Efficiency comparison

The results in Section B.6 demonstrate that RBF-FD and GMLS have similar asymptotic convergence rates for the same ℓ, but that RBF-FD can achieve lower errors for the same N and stencil sizes. In this section, we consider which of the methods is more computationally efficient in terms of error per computational cost. We examine both the efficiency when the setup costs are included and when just the evaluation costs are included, as measured by (A.23) and (A.24), respectively. We limit this comparison to $\tau = 1.5$ since this gave the best results for GMLS. Figure **??** displays the results of this examination for the case of computing the surface Laplacian on the tours discretized with Poisson disk sampling. Similar results were obtained for other SDOs and for the sphere, so we omit them. We see from the figure that GMLS is more efficient when the setup costs are included, but that RBF-FD is more efficient when only evaluation costs are included. For problems where the point sets are fixed and approximations to a SDO are required to be performed multiple times—as occurs when solving a time-dependent surface PDEs—the setup costs are not as important

as the evaluation costs since they are amortized across all time-steps. In this scenario RBF-FD is the more efficient method.

A.8 Concluding remarks

We presented a thorough comparison of the GMLS and RBF-FD methods for approximating the three most common SDOs: the gradient, divergence, and Laplacian (Laplace-Beltrami). Our analysis of the two different formulations of SDOs used in the methods revealed that if the exact tangent space for the surface is used, these formulations are identical. Our numerical investigation of the methods showed that they appear to converge at similar rates when the same polynomial degree ℓ is used, but that RBF-FD generally gives lower errors for the same N and ℓ. We further examined the dependency of the stencil size on the methods (as measured by the τ parameter) and found that the errors produced by GMLS deteriorate as the stencil size increases. The errors for RBF-FD, contrastingly, appear to keep improving as the stencil size increases. However, we don't expect this trend to continue indefinitely, as eventually the tangent plane formulation breaks down when the stencil size becomes too large. Finally, we investigated the computational efficiency of the methods in terms of error versus computational cost and found GMLS to be more efficient when setup costs are included and RBF-FD to be more efficient when only considering evaluation costs.

Acknowledgements

AMJ was partially supported by US NSF grant CCF-1717556. PAB & PAK were supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR) Program and Biological and Environmental Research (BER) Program under a Scientific Discovery through Advanced Computing (SciDAC

4) BER partnership pilot project. PAK was additionally supported by the Laboratory Directed Research & Development (LDRD) program at Sandia National Laboratories and ASCR under Award Number DE-SC-0000230927. AMJ was also partially supported by the Climate Model Development and Validation (CMDV) program, funded by BER. Part of this work was conducted while AMJ was employed at the Computer Science Research Institute at Sandia National Laboratories. GBW was partially supported by U.S. NSF grants CCF-1717556 and DMS-1952674.

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2022-XXXXJ.

REFERENCES

[1] Álvarez, Diego, González-Rodríguez, Pedro, and Kindelan, Manuel. 2021. A Local Radial Basis Function Method for the Laplace–Beltrami Operator. *J. Sci. Comput.*, **86**(3), 28.

[2] Banerjee, Sanjoy, Lakehal, Djamel, and Fulgosi, Marco. 2004. Surface divergence models for scalar exchange between turbulent streams. *International Journal of Multiphase Flow*, **30**(7), 963–977. A Collection of Papers in Honor of Professor G. Yadigaroglu on the Occasion of his 65th Birthday.

[3] Bayona, Victor. 2019a. Comparison of Moving Least Squares and RBF+Poly for

Interpolation and Derivative Approximation. *J. Sci. Comput.*, **81**(1), 486–512.

[4] Bayona, Víctor. 2019b. An insight into RBF-FD approximations augmented with polynomials. *Comput. Math. Appl.*, **77**(9), 2337–2353.

[5] Bayona, Victor, Flyer, Natasha, Fornberg, Bengt, and Barnett, Gregory A. 2017. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs. *J. Comput. Phys.*, **332**, 257–273.

[6] Bayona, Víctor, Flyer, Natasha, and Fornberg, Bengt. 2019. On the role of polynomials in RBF-FD approximations: III. Behavior near domain boundaries. *J. Comput. Phys.*, **380**, 378–399.

[7] Bertalmío, M., Cheng, L., Osher, S., and Sapiro, G. 2001. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, **174**, 759–780.

[8] Bosler, P.A., Wang, L., Jablonowski, C., and Krasny, R. 2014. A Lagrangian particle/panel method for the barotropic vorticity equations on a rotating sphere. *Fluid Dyn. Res.*, **46**.

[9] Cui, Jianjun, and Freeden, Willi. 1997. Equidistribution on the Sphere. *SIAM J. Sci. Stat. Comput*, **18**, 595–609.

[10] Davydov, Oleg, and Schaback, Robert. 2019. Optimal stencils in Sobolev spaces. *IMA J. Num. Analy.*, **39**(1), 398–422.

[11] Demanet, Laurent. 2006. Painless, highly accurate discretizations of the Laplacian on a smooth manifold. *Technical report, Stanford University.*

[12] Dziuk, Gerhard, and Elliott, Charles M. 2013. Finite element methods for surface PDEs. *Acta Numerica*, **22**, 289–396.

[13] Fasshauer, G. E. 2007. *Meshfree Approximation Methods with MATLAB, Interdisciplinary Mathematical Sciences*. Singapore: World Scientific Publishers.

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

[15] Fornberg, Bengt, and Flyer, Natasha. 2015. *A Primer on Radial Basis Functions with Applications to the Geosciences*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

[16] Fuselier, Edward J, and Wright, Grady B. 2013. A high-order kernel method for diffusion and reaction-diffusion equations on surfaces. *J. Sci. Comput.*, **56**(3), 535–565.

[17] Gowan, Evan J., Zhang, Xu, Khosravi, Sara, Rovere, Alessio, Stocchi, Paolo, Hughes, Anna L. C., Gyllencreutz, Richard, Mangerud, Jan, Svendsen, John-Inge, and Lohmann, Gerrit. 2021. A new global ice sheet reconstruction for the past 80 000 years. *Nature Communications*, **12**(1), 1199.

[18] Gross, B. J., Trask, N., Kuberry, P., and Atzberger, P. J. 2020. Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: A Generalized Moving Least-Squares (GMLS) approach. *J. Comput. Phys.*, **409**, 109340.

[19] Gunderman, David, Flyer, Natasha, and Fornberg, Bengt. 2020. Transport

schemes in spherical geometries using spline-based RBF-FD with polynomials. *J. Comput. Phys.*, **408**, 109256.

[20] Kuberry, Paul, Bosler, Peter, and Trask, Nathaniel. 2019 (Jan.). *Compadre Toolkit.*

[21] Lehto, E, Shankar, V, and Wright, G. B. 2017. A radial basis function (RBF) compact finite difference (FD) scheme for reaction-diffusion equations on surfaces. *SIAM J. Sci. Comput.*, **39**, A219–A2151.

[22] Liang, Jian, and Zhao, Hongkai. 2013. Solving Partial Differential Equations on Point Clouds. *SIAM J. Sci. Comput.*, **35**(3), A1461–A1486.

[23] Liu, Jinghui, Totz, Jan F., Miller, Pearson W., Hastewell, Alasdair D., Chao, Yu-Chen, Dunkel, Jörn, and Fakhri, Nikta. 2021. Topological braiding and virtual particles on the cell membrane. *Proc. Natl. Acad. Sci. U.S.A.*, **118**(34).

[24] Mahadevan, Vijay S, Guerra, Jorge E, Jiao, Xiangmin, Kuberry, Paul, Li, Yipeng, Ullrich, Paul, Marsico, David, Jacob, Robert, Bochev, Pavel, and Jones, Philip. 2022. Metrics for Intercomparison of Remapping Algorithms (MIRA) protocol applied to Earth system models. *Geosci. Model Dev.*, **15**(17), 6601–6635.

[25] Mikkelsen, Morten S. 2020. Surface Gradient–Based Bump Mapping Framework. *Journal of Computer Graphics Techniques (JCGT)*, **9**(4), 60–91.

[26] Mirzaei, Davoud. 2016. Error bounds for GMLS derivatives approximations of Sobolev functions. *J. Comput. Appl. Math*, **294**, 93–101.

[27] Mirzaei, Davoud, Schaback, Robert, and Dehghan, Mehdi. 2011. On generalized moving least squares and diffuse derivatives. *IMA J. Numer. Anal.*, **32**, 983–1000.

[28] O'neill, Barrett. 2006. *Elementary Differential Geometry*. Second edn. Elsevier.

[29] Petras, Argyrios, Ling, Leevan, and Ruuth, Steven J. 2018. An RBF-FD closest point method for solving PDEs on surfaces. *J. Comput. Phys.*, **370**, 43–57.

[30] Piret, Cécile, and Dunn, Jarrett. 2016. Fast RBF OGr for solving PDEs on arbitrary surfaces. *AIP Conference Proceedings*, **1776**(1), 070005.

[31] Shankar, Varun, Wright, Grady B., Kirby, Robert M., and Fogelson, Aaron L. 2014. A Radial Basis Function (RBF)-Finite Difference (FD) Method for Diffusion and Reaction-Diffusion Equations on Surfaces. *J. Sci. Comput.*, **63**(3), 745–768.

[32] Shankar, Varun, Narayan, Akil, and Kirby, Robert M. 2018. RBF-LOI: Augmenting Radial Basis Functions (RBFs) with Least Orthogonal Interpolation (LOI) for solving PDEs on surfaces. *J. Comput. Phys.*, **373**, 722–735.

[33] Shaw, Sage. 2019. *Radial Basis Function Finite Difference Approximations of the Laplace-Beltrami Operator*. M.Phil. thesis, Boise State University, USA. 1587.

[34] Stoop, Norbert, Lagrange, Romain, Terwagne, Denis, Reis, Pedro M, and Dunkel, Jörn. 2015. Curvature-induced symmetry breaking determines elastic surface patterns. *Nature Materials*, **14**(3), 337–342.

[35] Suchde, Pratik, and Kuhnert, Jörg. 2019. A meshfree generalized finite difference method for surface PDEs. *Comp. Math. Appl.*, **78**(8), 2789–2805.

[36] Trask, Nathaniel, and Kuberry, Paul. 2020. Compatible meshfree discretization of surface PDEs. *Computational Particle Mechanics*, **7**, 271–277.

[37] Trask, Nathaniel, Perego, Mauro, and Bochev, Pavel. 2017. A high-order staggered meshless method for elliptic problems. *SIAM J. Sci. Comput.*, **39**(2), A479–A502.

[38] Vallis, Geoffrey K. 2006. *Atmospheric and Oceanic Fluid Dynamics: Fundamentals and Large-scale Circulation*. Cambridge University Press.

[39] Walker, Shawn W. 2015. *The Shapes of Things*. Philadelphia: Society for Industrial and Applied Mathematics.

[40] Wendland, Holger. 2005. *Scattered Data Approximation*. Vol. 17. Cambridge: Cambridge University Press.

[41] Wendland, Holger, and Künemund, Jens. 2020. Solving partial differential equations on (evolving) surfaces with radial basis functions. *Advances in Computational Mathematics*, **46**, 64.

[42] Williamson, David L. 2007. The Evolution of Dynamical Cores for Global Atmospheric Models. *J. Meteorol. Soc. Jpn.*, **85B**, 241–269.

[43] Wright, G. B., Jones, A. M., and Shankar, V. 2022. MGM: A meshfree geometric multilevel method for systems arising from elliptic equations on point cloud surfaces. *SIAM J. Sci. Comput.*, **Accepted**.

[44] Yuksel, Cem. 2015. Sample Elimination for Generating Poisson Disk Sample Sets. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2015)*, **34**(2), 25–32.

APPENDIX B:

PAPER 2: MGM: A MESHFREE GEOMETRIC MULTILEVEL METHOD FOR LINEAR SYSTEMS ARISING FROM ELLIPTIC EQUATIONS ON POINT CLOUD SURFACES

Authors: Grady B. Wright, Andrew M. Jones, Varun Shankar

B.1 Abstract

We develop a new meshfree geometric multilevel (MGM) method for solving linear systems that arise from discretizing elliptic PDEs on surfaces represented by point clouds. The method uses a Poisson disk sampling-type technique for coarsening the point clouds and new meshfree restriction/interpolation operators based on polyharmonic splines for transferring information between the coarsened point clouds. These are then combined with standard smoothing methods in a V-cycle iteration. MGM is applicable to discretizations of elliptic PDEs based on various localized meshfree methods, including RBF finite differences (RBF-FD) and generalized finite differences (GFD). We test MGM both as a standalone solver and preconditioner for Krylov subspace methods on several test problems using RBF-FD and GFD, and numerically analyze convergence rates, efficiency, and scaling with increasing point cloud sizes. We also perform a side-by-side comparison to algebraic multigrid (AMG) methods for solving the same systems. Finally, we further demonstrate the effectiveness of MGM by applying it to three challenging applications on complicated surfaces: pattern formation, surface harmonics, and geodesic distance.

B.2 Introduction

Partial differential equations (PDEs) defined on surfaces (or manifolds) arise in many areas of science and engineering, where they are used to model, for example, atmospheric flows [50], chemical signaling on cell membranes [28], morphogenesis [42], and textures for computer graphics [46]. Solutions of these models can rarely be achieved by analytical means and must instead be approximated using numerical techniques. While numerical methods for PDEs on the sphere have been developed since the

1960s [50], development of methods for PDEs on more general surfaces only began in the late 1980s [15], with interest growing considerably in the early 2000s [16]. These techniques include surface finite element (SFE) [16], embedded finite element (EFE) [8**?**], and closest point (CP) [29] methods. More recently, various meshfree (or meshless) methods have also been developed for PDEs on general surfaces that use a local stencil approach, including radial basis function-finite differences (RBF-FD) [2, 26, 39, 35, 40, 48], generalized finite differences (GFD) [43], and generalized moving least squares (GMLS) [27, 44, 20]. These methods can be applied for surfaces represented only by point clouds and do not require a surface triangulation like SFE methods or a level-set representation of the surface like EFE methods. Additionally, these meshfree methods approximate the solutions directly on the point cloud and do not extend the PDEs into the embedding space like the EFE and CP methods.

In this paper, we concentrate on local meshfree methods for elliptic PDEs on surfaces, which are challenging to solve with iterative methods because of the poor conditioning of the systems. We specifically focus on the surface Poisson and shifted (or screened) surface Poisson problems, which, for example, in surface hydrodynamics [20], computer graphics [36], and time-implicit discretizations of surface reaction diffusion equations [39]. We focus on two methods for these PDEs: polyharmonic spline-based RBF-FD with polynomials and GFD. These meshfree discretizations result in large, sparse, non-symmetric, linear systems of equations that need to be solved. Direct solvers for these systems have most commonly been used, but these do not scale well to large point clouds and high-orders of accuracy, motivating the need for efficient and robust iterative methods.

Multigrid methods are known to be effective solvers and preconditioners for linear

systems that arise from discretizing elliptic PDEs (e.g., [45]). These methods can be classified into two types: geometric and algebraic. While algebraic multigrid (AMG) methods are general purpose solvers/preconditioners, geometric methods, when they can be developed, generally converge faster and work as better preconditioners for Krylov subspace methods. Geometric multigrid methods have been developed for SFE and CP discretizations (e.g., [24] and [10]) and it is the aim of this paper to also develop these methods for meshfree discretizations.

The basic components of geometric multigrid methods that need to be developed are (1) techniques for coarsening the grid or mesh, (2) constructing interpolation/restriction operators for transferring the information between levels, (3) discretizing the differential operator on the coarser levels, (4) smoothing the approximate solution, and (5) solving the system of the coarsest level. The first component presents a challenge for meshfree surface PDEs as there is only a point cloud available and no grid or mesh to create a hierarchy of coarser levels. To overcome this challenge we use the weighted sample elimination (WSE) method from [52], which is a general purpose method for selecting quasi-uniformly spaced subsets of points from a point cloud and falls into the general category of Poisson disk sampling methods [9]. The lack of a grid or mesh also presents a challenge for component (2) as standard transfer operators cannot be used. To overcome this challenge, we use RBFs to construct the interpolation operators for transferring the defect from coarser to finer levels. For the restriction operators, we simply use the transpose of the interpolation operators, which is a standard choice [45]. With these transfer operators, we generate component (3) using a Galerkin projection, often referred to as the Galerkin coarse grid operator. Finally, for component (4) we use standard Gauss-Seidel smoothing and for

(5) we use a direct solver. We combine all of these components in a V-cycle iteration and apply it both as a solver and preconditioner. The resulting method is entirely meshfree and we refer to it as the meshfree geometric multilevel[1] (MGM) method.

The new MGM method has some similarities to the meshfree *multicloud* methods [22, 54], but also some key differences. The first major difference is that multicloud methods have been developed for PDEs posed in planar domains, whereas MGM is for surface PDEs. Another difference is with the choice of transfer operators. The method of [54] uses one-point, piecewise constant operators, while the method of [22] uses two-point, inverse-distance weighted interpolation and restriction operators. It is not clear how these latter transfer operators should be generalized to surfaces. MGM instead uses transfer operators based on RBFs, which are well suited for interpolation on surfaces [17]. A second difference is the strategy for geometric coarsening of the given point cloud. Multicloud methods use a graph coloring-type scheme for finding maximally-independent subsets of vertices to determine the coarser levels. This does not allow the size of the point clouds on the coarser levels to be controlled precisely, and it limits the coarsening factors to approximately four (for 2D problems). MGM instead uses WSE [52], which allows for arbitrary coarsening factors and for the sizes of the points in the coarser levels to be controlled exactly. A third difference is that MGM can handle degenerate PDEs (e.g., the surface Poisson equation), while the multicloud methods have been tailored to non-degenerate PDEs (e.g. planar Poisson equation with mixed Dirichlet-Neumann boundary conditions). Finally, multicloud methods have only been tested on second order accurate discretizations of PDEs; these discretizations typically only use small stencils. We demonstrate that MGM

[1]We use the term multilevel rather than multigrid, since this method does not depend on a grid.

works for discretizations at least up to sixth order accurate with large stencil sizes.

The remainder of the paper is organized as follows. In Section B.3, an overview is given of the two meshfree methods for surface PDEs that the MGM algorithm is used to solve. The next section presents the RBF-based transfer operators. Section B.5 describes the remaining components of the MGM algorithm, including a discussion of the changes necessary to solve the degenerate surface Poisson problem. Section B.6 presents an extensive array of numerical results for the MGM method, including a comparison with algebraic multigrid (AMG) methods. Section B.7 then uses the method in three challenging applications to further demonstrate its effectiveness. Finally, the paper concludes with some final remarks on the method and some future directions in Section B.8.

B.2.1 Assumptions and notation

Throughout the manuscript we let $\mathcal{M}$ be a smooth embedded manifold of co-dimension one in ^{3} with no boundary and let $\Delta_{\mathcal{M}}$ denote the Laplace-Beltrami operator (LBO) (or surface Laplacian) on $\mathcal{M}$. When referencing points on $\mathcal{M}$, we assume they are represented as coordinates in $\mathbb{R}^3$, e.g., for $\mathbf{x} \in \mathcal{M}$, and we write $\mathbf{x} = (x, y, z)$. For a point $\mathbf{x} \in \mathcal{M}$, we let $T_{\mathbf{x}}\mathcal{M}$ denote the tangent plane to $\mathcal{M}$ at $\mathbf{x}$. We denote normal vectors to $\mathcal{M}$ as $\mathbf{n}$ and assume that they are available either analytically or using some approximation technique (see for example [23]).

We use sub/superscripts h and H on variables to indicate whether they are associated with the fine or coarse level point clouds, respectively. For example, X_h and X_H denote the set of points in the fine level and coarse level point clouds, respectively. This is meant to mimic the notation that is used in traditional grid based geometric multigrid methods [45], but these parameters do not relate to anything specific about

the spacing of the points and do not need to be computed.

The focus of this study is on the elliptic equation

$$\mathcal{L}u = f, \tag{B.1}$$

where $u : \mathcal{M} \to \mathbb{R}$ is unknown and $f : \mathcal{M} \to \mathbb{R}$ is known. Here $\mathcal{L} = \Delta_{\mathcal{M}}$ or $\mathcal{L} = \mathcal{I} - \mu\Delta_{\mathcal{M}}$, where $\mathcal{I}$ is the identity operator and $\mu > 0$, which correspond to the surface Poisson and shifted (or screened) surface Poisson equation (B.1), respectively. For the surface Poisson problem, we assume f satisfies the *compatibility condition* $\int_{\mathcal{M}} f \, dA = 0$, which is a necessary and sufficient condition for (B.1) to have a solution. In this case, any solution is unique up to the addition of a constant since constants satisfy the homogeneous equation.

B.3 Localized meshfree discretizations

Several localized meshfree methods have been developed for approximating the solution of (B.1), e.g., [27, 44, 2, 26, 39, 35, 40]. For the sake of brevity, we limit the focus of this study to two localized meshfree methods: polyharmonic spline (PHS)-based RBF-FD with polynomials and GFD. Both of these methods use the so-called tangent plane approach, but differ in the approximation spaces used. They should be sufficient to demonstrate the general applicability of the MGM method.

The RBF-FD and GFD methods are based on approximating the strong form of the equation, and amount to discretizing the LBO $\Delta_{\mathcal{M}}$ over a local stencil of points on the surface. This stencil based approach can generally be described as follows. Let $\mathbf{X}_h = \{\mathbf{x}_i\}_{i=1}^{N_h}$ denote the global point cloud (node set) discretizing $\mathcal{M}$. For each $\mathbf{x}_i \in \mathbf{X}_h$, $i = 1, \ldots, N_h$, let σ_i^h denote the set of indices of the $n_i > 1$ nearest neighbor

nodes in $\mathbf{X}_h$ to $\mathbf{x}_j$. Here we use the Euclidean distance in ^{3} to define the nearest neighbor distances. The points $\mathbf{X}_h^i = \{\mathbf{x}_j\}_{j\in\sigma_i^h}$ are the *stencil* for $\mathbf{x}_i$, and $\mathbf{x}_i$ is the *stencil center*. The stencil based approximation to $\Delta_{\mathcal{M}}u$ at $\mathbf{x}_j$ then takes the form

$$\Delta_{\mathcal{M}}u\big|_{\mathbf{x}_i} \approx \sum_{j\in\sigma_h^i} c_{ij}u(\mathbf{x}_j), \tag{B.2}$$

where c_{ij} are some set of weights determined by the RBF-FD or GFD methods discussed below. These weights can then be assembled into a global N_h-by-N_h (sparse) differentiation matrix D_h and an approximate solution to (B.1) is given as a solution to the linear system

$$L_h u^h = f^h, \tag{B.3}$$

where $u^h, f^h \in \mathbb{R}^{N_h}$ contain the unknown solution and known right hand side of (B.1), respectively, sampled at $\mathbf{X}_h$. The matrix L_h is given by $L_h = D_h$ or $L_h = I_h - \mu D_h$, where I_h is the N_h-by-N_h identity matrix. Note that the latter L_h arises from time-implicit discretizations of surface diffusion-type problems. The MGM method will be used for solving the system (B.3).

In the remainder of this section, we give specific details on determining the stencil weights in (B.2) for the LBO using RBF-FD and GFD methods. Since both of these methods use the tangent plane technique, we review it first.

B.3.1 Tangent plane method

The tangent plane idea was introduced by Demanet [13] and recently further refined by Suchde & Kuhnert [43], Shaw [41], and, in the case of the unit two-sphere, by

Gunderman et. al. [21]. The central idea of the method is to approximate the LBO at the center of each stencil $\mathbf{X}_h^i$ using an approximation to the standard Laplacian on the plane tangent to the surface at the stencil center, $T_{\mathbf{x}_i}\mathcal{M}$. The approximation is constructed from a projection of the stencil points to $T_{\mathbf{x}_i}\mathcal{M}$. In this work, we use the projection advocated in [43], which is known as an orthographic projection when $\mathcal{M}$ is the unit sphere.

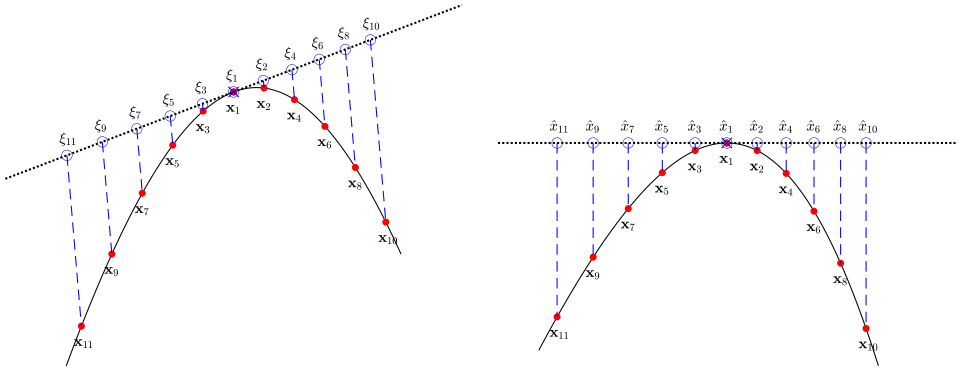


Figure B.1: Illustration of the tangent plane method for a 1D surface (curve). The solid black lines indicates the surface, the solid red circles mark the $n = 11$ stencil nodes, the open blue circles mark the projected nodes, and the $\times$'s marks the stencil center. (a) Direct projection of the stencil points according to (B.4)**. (b) Rotation and projection of the stencil points according to** (B.6)**.**

With out loss of generality, we describe the method for the first stencil $\mathbf{X}_h^1$ with index set σ_h^1. To simplify notation, we assume $\sigma_h^1 = \{1, \ldots, n\}$, so that the stencil points are simply $\mathbf{X}_h^1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The tangent plane method from [43] projects these points onto $T_{\mathbf{x}_1}\mathcal{M}$ along the normal vector $\mathbf{n}_1$ (the normal to the surface at $\mathbf{x}_1$). This projection is illustrated in Figure B.1 (a) for a one dimensional surface (curve)

in ^{2}. The projected points can be computed explicitly by

$$\boldsymbol{\xi}_j = (I - \mathbf{n}_1\mathbf{n}_1^T)(\mathbf{x}_j - \mathbf{x}_1),\ j = 1, \ldots, n, \tag{B.4}$$

where we have shifted the projected points so that $\boldsymbol{\xi}_1$ is at the origin. For a two dimensional surface, the projected points can be expressed in terms of orthonormal vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ that span $T_{\mathbf{x}_1}\mathcal{M}$ as follows

$$\boldsymbol{\xi}_j = \underbrace{\begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 \end{bmatrix}}_{R} \underbrace{\begin{bmatrix} \hat{x}_j \\ \hat{y}_j \end{bmatrix}}_{\hat{\mathbf{x}}_j},\ j = 1, \ldots, n. \tag{B.5}$$

The 2D coordinates $\hat{\mathbf{x}}_j$ in the tangent plane for the projected points are what will be used for constructing the approximations to the Laplacian. These can be computed directly from the relationship (B.4) and (B.5) as

$$\hat{\mathbf{x}}_j = R^T(\mathbf{x}_j - \mathbf{x}_1),\ j = 1, \ldots, n, \tag{B.6}$$

where we have used $R^T(I - \mathbf{n}_1\mathbf{n}_1^T) = R^T$. We denote the the projected stencil as $\hat{\mathbf{X}}_h^1 = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n\}$. The above procedure is repeated for every stencil $\mathbf{X}_h^i$, to obtain the projected stencils $\hat{\mathbf{X}}_h^i$, $i = 1, \ldots, N_h$.

We note that, geometrically speaking, (B.6) amounts to first shifting the stencil points so the center is at the origin, rotating them so the normal $\mathbf{n}_1$ is orthogonal to the xy-plane, and then dropping the third component. This is illustrated in Figure B.1 (b) for the case of a 1D curve.

B.3.2 PHS-based RBF-FD with polynomials

The RBF-FD method for determining the weights in (B.2) can be derived by constructing an RBF interpolant over each of the projected stencil points to the tanget plane, applying the standard 2D Laplacian to the interpolants, and then evaluating them at the stencil center. In this study we focus on interpolants constructed from PHS kernels and polynomials [5, 14, 41]. Without loss of generality, we again describe the method for the first stencil, with $\mathbf{X}_h^1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, to simplify notation.

For the stencil $\mathbf{X}_h^1$, the PHS interpolant to the projected stencil $\hat{\mathbf{X}}_h^1$ takes the form

$$s(\hat{\mathbf{x}}) = \sum_{i=1}^{n} a_i \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_i\|^{2k+1} + \sum_{j=1}^{L} b_j p_j(\hat{\mathbf{x}}), \tag{B.7}$$

where $\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} & \hat{y} \end{bmatrix}^T \in T_{\mathbf{x}_1}\mathcal{M}$, $\|\cdot\|$ denotes the Euclidean norm, k is the order of the PHS kernel, and $\{p_1, \ldots, p_L\}$ is a basis for bivariate polynomials in $T_{\mathbf{x}_1}\mathcal{M}$ of degree ℓ (so that $L = (\ell+1)(\ell+2)/2$). The order k controls the smoothness of the PHS and is chosen such that $0 \leq k \leq \ell$. We note that the polynomials can be chosen to be the standard bivariate monomials in the components of $\hat{\mathbf{x}}$. For samples $\{u_1, \ldots, u_n\}$ of an arbitrary function at the stencil points $\mathbf{X}_h^1$, the coefficients for the interpolant in the tangent plane are determined by the conditions

$$s(\hat{\mathbf{x}}_i) = u_i,\ i = 1, \ldots, n \quad \text{and} \quad \sum_{i=1}^{n} a_i p_j(\hat{\mathbf{x}}_i) = 0,\ j = 1, \ldots, L, \tag{B.8}$$

which can be written as the following linear system

$$\begin{bmatrix} A & P \\ P^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} = \begin{bmatrix} \underline{u} \\ \underline{0} \end{bmatrix}, \tag{B.9}$$

where $A_{ij} = \|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|^{2k+1}$ $(i, j = 1, \ldots, n)$, $P_{ij} = p_j(\hat{\mathbf{x}}_i)$ $(i = 1, \ldots, n,\ j = 1, \ldots, L)$, and underlined terms denote vectors containing the corresponding terms in (B.8). If the stencil nodes $\hat{\mathbf{X}}_h^1$ are unisolvent with respect to the space of bivariate polynomials of degree ℓ (i.e., $\text{rank}(P) = L$), then (B.9) has a unique solution, so that (B.7) is well-posed [47]. This is a mild condition on the stencil nodes, especially for "scattered" nodes on the tangent plane.

The stencil weights c_{1j} in (B.2) are determined from the approximation

$$\begin{aligned} \Delta_{\mathcal{M}} u\big|_{\mathbf{x}_1} \approx \hat{\Delta} s\big|_{\hat{\mathbf{x}}_1} &= \sum_{i=1}^{n} a_i \hat{\Delta}(\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_i\|^{2k+1})\big|_{\hat{\mathbf{x}}_1} + \sum_{j=1}^{L} b_j \hat{\Delta}(p_j(\hat{\mathbf{x}}))\big|_{\hat{\mathbf{x}}_1} \\ &= \begin{bmatrix} \hat{\Delta}\underline{s} & \hat{\Delta}\underline{p} \end{bmatrix}^T \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix}, \end{aligned}$$

where $\hat{\Delta} = \partial_{\hat{x}\hat{x}} + \partial_{\hat{y}\hat{y}}$, $\hat{\Delta}\underline{s}$ and $\hat{\Delta}\underline{p}$ are vectors containing the entries $\hat{\Delta}(\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_i\|^{2k+1})\big|_{\hat{\mathbf{x}}_1}$, $i = 1, \ldots, n$, and $\hat{\Delta}(p_j(\hat{\mathbf{x}}))\big|_{\hat{\mathbf{x}}_1}$, $j = 1, \ldots, L$, respectively. Using (B.9) in the above expression, the stencil weights are given as the solution to the following linear system

$$\begin{bmatrix} A & P \\ P^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \underline{c} \\ \underline{\lambda} \end{bmatrix} = \begin{bmatrix} \hat{\Delta}\underline{s} \\ \hat{\Delta}\underline{p} \end{bmatrix}, \tag{B.10}$$

where $\underline{c}$ contains c_{1j} and $\underline{\lambda}$ are unused.

We note that one can interpret (B.10) as the solution to an equality constrained optimization problem where the weights are determined by enforcing they are exact for $\hat{\Delta}(\|\hat{\mathbf{x}}-\hat{\mathbf{x}}_i\|^{2k+1})\big|_{\hat{\mathbf{x}}_1}$, $j=1,\ldots,n$, subject to the constraint that they are also exact for $\hat{\Delta}(p_j(\hat{\mathbf{x}}))\big|_{\hat{\mathbf{x}}_1}$, $j=1,\ldots,L$. Under this interpretation, $\underline{\lambda}$ is the vector of Lagrange multipliers [5].

In this work, we choose the order of the PHS as $k=\ell$ and fix the stencil size $n_j=n$, $j=1,\ldots,N$ as $n=\lceil 2L \rceil$, which is a common choice for RBF-FD methods [5]. The degree ℓ of the appended polynomial can then be used to control the approximation order of the method, with larger ℓ leading to higher orders [12].

B.3.3 GFD

This method is similar to the RBF-FD method, but instead of using an interpolant, the method is based on a (weighted) polynomial least squares approximant. Using the same notation and assumptions as the previous section and again focusing only on the first stencil $\mathbf{X}_h^1$, the approximant for the projected stencil $\hat{\mathbf{X}}_h^1$ takes the form

$$q(\hat{\mathbf{x}}) = \sum_{j=1}^{L} b_j p_j(\hat{\mathbf{x}}). \tag{B.11}$$

The coefficients of the approximant are determined from the samples $\{u_1,\ldots,u_n\}$ according to the the following weighted least squares problem:

$$\underline{b}^* = \underset{\underline{b}\in\mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{n} w(\hat{\mathbf{x}}_i)(q(\hat{\mathbf{x}}_i)-u_i)^2 = \underset{\underline{b}\in\mathbb{R}^n}{\operatorname{argmin}} \|W^{1/2}(P\underline{b}-\underline{u})\|_2^2, \tag{B.12}$$

where $W=(w(\hat{\mathbf{x}}_i))$. Here we again assume the stencil nodes $\hat{\mathbf{X}}_h^1$ are unisolvent with respect to bivariate polynomials of degree L so that (B.12) has a unique solution.

There are many different options for selecting the weight function w in the literature. In this work, we follow [43] and use the following Gaussian function:

$$w(\hat{\mathbf{x}}_i) = \exp\left(-\alpha \frac{\|\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_i\|^2}{\rho_1^2 + \rho_i^2}\right),$$

where ρ_k is the support of the kth projected stencil $\hat{\mathbf{X}}_h^i$, i.e. the radius of the minimum ball centered at $\hat{\mathbf{x}}_i$ that encloses all the points in $\hat{\mathbf{X}}_h^i$. The parameter $\alpha > 0$ is used for controlling the shape of the weight function and is typically chosen in an ad hoc manner [43].

The stencil weights in (B.2) are determined from the approximation

$$\Delta_{\mathcal{M}} u\big|_{\mathbf{x}_1} \approx \hat{\Delta} q\big|_{\hat{\mathbf{x}}_1} = (\hat{\Delta}\underline{p})^T \underline{b}^*.$$

Using the normal equation solution of (B.12) in the above expression, the vector of stencil weights $\underline{c}$ is given as

$$\underline{c} = WP(P^T W P)^{-1}(\hat{\Delta}\underline{p}). \tag{B.13}$$

In practice, a QR factorization of WP is used instead of the normal equations to improve the numerical conditioning of (B.13).

In this work, we choose the number of points in the stencils $\mathbf{X}_h^i$ for this method in the same manner as the RBF-FD technique. Note that, similar to RBF-FD, increasing the polynomials degree ℓ also increases the order of accuracy of the GFD method.

B.4 Multilevel transfer operators using RBFs

Operators for transferring information between coarse and fine levels are one of the key components of multilevel methods. The interpolation transfer operators are used to transfer information from coarse level to a finer level, while the restriction operators are used for the reverse. Let $\mathbf{X}_h = \{\mathbf{x}_i\}_{i=1}^{N_h} \subset \mathcal{M}$ denote the fine set of nodes and $\mathbf{X}_H = \{\mathbf{y}_j\}_{j=1}^{N_H} \subset \mathcal{M}$ the coarse set, where $N_H < N_h$. We denote the interpolation operator by I_H^h and the restriction operator by I_h^H. These can be represented of as (sparse) matrices, so that for a vector u_H of data on the coarse nodes $\mathbf{X}_H$, the vector containing the interpolation of u^H to $\mathbf{X}_h$ is given as $u^h = I_H^h u_H$. In this section, we discuss a novel meshfree method for constructing I_H^h based on RBF interpolation. For the restriction operator, we use $I_h^H = (I_H^h)^T$, which is a standard choice, especially in AMG methods [45, Appendix A].

Similar to the discrete LBO, we compute the interpolation operator using a stencil based approach. Letting σ_H^i be the indicies of the m_i nearest neighbors in $\mathbf{X}_H$ to $\mathbf{x}_i$, the interpolation of $\{u_j^H\}_{j\in\sigma_i^H}$ to u_i^h, the entry in u_h corresponding to $\mathbf{x}_i$, is given as

$$u_i^h = \sum_{j\in\sigma_H^i} d_{ij} u_j^H. \tag{B.14}$$

We again use the Euclidean distance in $\mathbb{R}^3$ to define the index set σ_H^i. The weights for each stencil can be assembled to form the (sparse) interpolation matrix I_H^h.

We use local RBF interpolants about each stencil $\mathbf{X}_H^i = \{\mathbf{y}_j\}_{j\in\sigma_i^H}$ to determine the weights in (B.14) and form these interpolants in the embedding space $\mathbb{R}^3$. This is considerably simpler than using intrinsic coordinates to $\mathcal{M}$ and the resulting interpolants have good approximation properties [17]. One could alternatively use interpolants in

the tangent plane about each stencil center similar to Section B.3.2, but these are only accurate when the surface is well discretized by the underlying point cloud. This will not necessarily be the case for the nodes $\mathbf{X}_H$ as we coarsen the the finer levels. We again use a PHS kernel to form the interpolants and describe the method for the first stencil $\mathbf{X}_H^1$, which, to simplify the notation, we assume to consist of the nodes $\{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$.

For the stencil $\mathbf{X}_H^1$, the PHS interpolant takes the same form as (B.7), but with $\hat{\mathbf{x}}$ replaced by $\mathbf{x}$ and $\{\hat{\mathbf{x}}_i\}$ is replaced by $\{\mathbf{y}_i\}$. Additionally, we only consider the PHS kernel with $k = 0$ and a constant term appended to the interpolant (i.e. $\ell = 0$). The weights d_{1j} in (B.14) are determined by evaluating this interpolant at $\mathbf{x}_1$ and can be computed in a similar procedure to that used in deriving the system (B.10). The linear system for the interpolation weights takes the form

$$\begin{bmatrix} A & \underline{1} \\ \underline{1}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \underline{d} \\ \lambda \end{bmatrix} = \begin{bmatrix} \underline{s} \\ 1 \end{bmatrix}, \tag{B.15}$$

where $A_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|$ $(i, j = 1, \ldots, m)$, $\underline{1}$ is the vector of length n with all ones, and $\underline{s}$ has entries $\|\mathbf{x}_1 - \mathbf{y}_i\|$, $i = 1, \ldots, m$. Again, λ is unused.

While higher order PHS kernels ($k > 0$) and higher degree polynomials ($\ell > 0$) could be used in constructing the interpolation weights, we found that the simple formulation above gave good results, while also being efficient, for the range of problems we considered. This formulation also has the added benefit that the system (B.15) has a unique solution, provided the points are distinct [47]. When using larger k and ℓ this may not be the case as the points must be unisolvent with respect to the space of trivariate polynomials of degree ℓ, i.e., $(P) = L$. Since the interpolation is done

Algorithm 2: Two-Level Cycle

1 Pre-smooth initial guess: $u^h \leftarrow \text{presmooth}(L_h, u^h, f^h, \nu_1)$;
2 Compute residual: $r^h = f^h - L_h u^h$;
3 Restrict the residual to $\mathbf{X}_H$: $r^H = I_h^H r^h$;
4 Solve for the defect: $L_H e^H = r^H$;
5 Interpolate the defect to $\mathbf{X}_h$: $e^h = I_H^h e^H$;
6 Correct the approximation: $u^h \leftarrow u^h + e^h$;
7 Post-smooth the approximation: $u^h \leftarrow \text{postsmooth}(L_h, u^h, f^h, \nu_2)$;

in the embedding space, this can be an issue for certain algebraic surfaces (e.g., the sphere with $\ell \geq 2$).

B.5 Meshfree geometric multilevel (MGM) method

In this section we present the MGM method for solving the discrete problem (B.3). We first present the MGM method in terms of a two-level cycle, which is summarized in Algorithm 2, describing its primary components: coarsening the point cloud $\mathbf{X}_h \to \mathbf{X}_H$, forming the coarse level operator L_H, smoothing the approximation, and solving for the defect on the coarse level. The interpolation/restriction operators are described in the previous section. We then focus on some modifications to the algorithm that are necessary when (B.3) corresponds to the surface Poisson problem. This is followed by a description of the multilevel extension of the method. Finally, we comment on using the method as a preconditioner for Krylov subspace methods.

B.5.1 Node coarsening

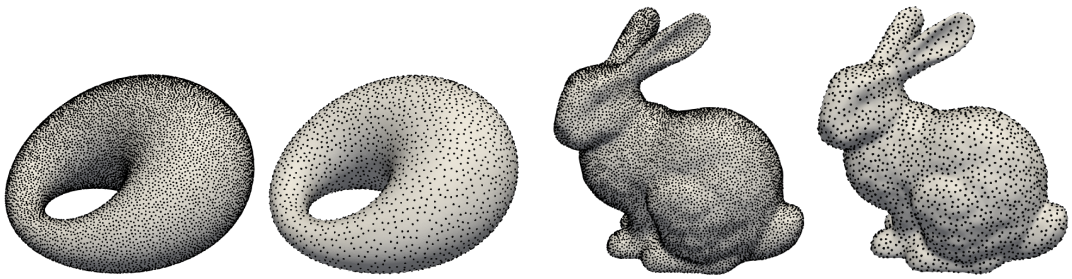
The technique we propose for generating the coarser point clouds on general surfaces is based on (WSE) method from [52]. This algorithm falls into the category of Poisson disk sampling methods, which produce quasiuniformly spaced point sets [9].

The WSE method approximates the solution to the following optimization problem: Given a point cloud $\mathbf{X}_h$ with N_h samples, determine a subset $\mathbf{X}_H$ of $\mathbf{X}_h$ with N_H samples that has maximal Poisson disk radius. The Poisson disk radius is defined as one half the minimum distance between neighboring points in the set (which is called the separation radius in the meshfree methods literature [47]). This optimization problem is NP complete, but the WSE algorithm approximates the solution in $N_h - N_H$ steps with a theoretical complexity of $\mathcal{O}(N_h \log N_h)$ operations [52]. The method works for point clouds defined on many different sampling domains, including arbitrary manifolds, where it uses the Euclidean norm in $\mathbb{R}^3$ to define nearest neighbor distances. We use the implementation by the author of the WSE method, called `cySampleElimination`, that is provided in the `cyCodeBase` [53].

In this work, we coarsen the point cloud $\mathbf{X}_h$ by a fixed factor of 4, so that $\mathbf{X}_H$ has $N_H = \lfloor N_h/4 \rfloor$ points. This mimics the standard coarsening of geometric multigrid for two-dimensional domains. We tested other coarsening factors, but found that coarsening by 4 generally gave the best results in terms of iteration count and wall clock time for the multilevel method. Figure B.2 illustrates the coarse point clouds $\mathbf{X}_H$ with this coarsening factor computed from the WSE algorithm for two example surfaces.

B.5.2 Coarse level operator

There are two main approaches to constructing the coarse level operator in multilevel methods. The first is *direct discretization*, where the differential operator is discretized directly on the coarse level points $\mathbf{X}_H$. The second is based on a Galerkin projection



(a) $\mathbf{X}_h \rightarrow \mathbf{X}_H$ (b) $\mathbf{X}_h \rightarrow \mathbf{X}_H$

Figure B.2: Illustration of the WSE algorithm for generating a coarse level set $\mathbf{X}_H$ of $N_H = \lfloor N_h/4 \rfloor$ points from a fine level set $\mathbf{X}_h$ of N_h points. Here $N_h = 14561$ & $N_H = 3640$ for the cyclide (a) and $N_h = 14634$ & $N_H = 3658$ for the Stanford Bunny (b) .

involving the interpolation I_H^h and restriction I_h^H operators, and is defined as follows:

$$L_H = I_h^H L_h I_H^h. \tag{B.16}$$

This latter operator, referred to as the *Galerkin coarse grid operator*, provides a simple means of coarsening L_h and has been shown to be robust for a large class of problems, especially those where a direct discretization on the coarse grid does not adequately represent the approximation on the fine grid [45, §7.7.4]. It also gives rise to a variational principle that is exploited in the analysis of algebraic multigrid (AMG) [45, §A.2.4] methods. While this latter result relies on the matrix being symmetric positive define, modifications to this theory have also been developed for non-symmetric problems, which involve choosing the restriction operator differently than the transpose of the interpolation operator [30]. We use the Galerkin approach for forming L_H, as we have found that it approximates the fine grid operator on the coarser levels better than the direct discretization technique and it makes for a more robust solver/preconditioner. While L_h is not symmetric for our discretizations, we

have nonetheless found that simply choosing $I_h^H = (I_H^h)^T$ works well over a large array of test problems.

One disadvantage of the Galerkin approach is that L_H has to be formed explicitly through sparse matrix-matrix multiplication, which is more computationally expensive in terms of time and memory than the direct discretization approach. Several researchers have developed methods to reduce this cost on parallel architectures (e.g.,[6, 4]), but we have not used these methods in our implementation. We simply construct L_H as part of a set-up phase using a sparse matrix library. However, we do some minor alterations to improve the computational performance. These include reordering the rows and columns of L_h to decrease its bandwidth using the reverse Cuthill-McKee (RCM) algorithm prior to forming L_H. This essentially leads to a reordering of the nodes $\mathbf{X}_h$, which in turn leads to a reordering of the interpolation operator I_H^h. We also use RCM to reorder the rows and columns of L_H after it is formed, which leads to a re-ordering of the nodes $\mathbf{X}_h$ and of the columns of I_H^h. We have found that these matrix reorderings not only reduce the wall-clock time of MGM, but also the number of iterations to reach convergence.

B.5.3 Smoother and coarse level solver

For the smoothing operator we use classical Gauss-Seidel (GS) method. One application of the smoother can be written as

$$u^h \leftarrow u^h + B_h^{-1}(f^h - L_h u^h), \tag{B.17}$$

where B_h is the lower triangular part (called forward GS) or upper triangular part (called backward GS) of L_h. In some cases we vary the version of the smoother for the

pre- and post- smoothing operations (e.g., forward GS for pre-smooth and backward GS for the post-smooth). We denote the number of applications of the smoother for the pre- and post-phases of the cycle as ν_1 and ν_2, respectively.

To solve for the defect on the coarse level, we use a direct solver based on a sparse LU factorization of L_H (e.g., SuiteSparse or SuperLU).

B.5.4 Modifications to the two-level cycle for the surface Poisson problem

When (B.3) corresponds to the discretization of a surface Poisson problem ($L_h = D_h$), the system is singular and some modifications to the two level cycle in Algorithm 2 are necessary. To understand the nature of the singularity, we can look at the continuous problem (B.1). As discussed in Section B.2.1, this problem has a solution if and only if the right hand side satisfies the compatibility condition. Furthermore, the solution is only unique up to the addition of a constant. The degeneracy in the continuous problem manifests in the discrete problem as a one dimensional null space of L_h corresponding to constant vectors. The discrete analog of the consistency condition is that (B.3) has a solution if and only if f^h is orthogonal to the left null vector of L_h (i.e., f^h is in the range of L_h). Also, similar to the continuous case, any solution of (B.3) is only unique up to the addition of a constant vector.

The primary issue that arises with using multilevel methods (and other iterative methods) for these types of singular systems stems from the fact that, in practice, f_h is rarely in the range of L_h. This can cause the iterations to fail to converge to a suitable approximation. Three standard approaches to bypass this issue include the following. First, one can project f^h into the range of L_h. However, this requires computing the

left null vector, which can be computationally expensive[2]. It also requires modifying the coarse level solver to use the pseudoinverse (or some approximate inverse). A second approach is to impose that the solution is zero at one point. This fixes the non-uniqueness issue and transforms the problem into solving a non-singular system of one dimension smaller. However, this can lead to a deterioration of the convergence of the multilevel method since the pointwise condition is not well approximated on coarser levels [49]. Additionally, the solution to this approach can be less accurate and less smooth than the projection approach [51]. The third approach is to enforce a global constraint on the solution, such as the discrete mean of u_h is zero [45, §5.6.4]. This constraint can be enforced using a Lagrange multiplier, which transforms the linear system into the constrained system

$$\underbrace{\begin{bmatrix} L_h & b_h^T \\ b_h & 0 \end{bmatrix}}_{\tilde{L}_h} \underbrace{\begin{bmatrix} u^h \\ \lambda^h \end{bmatrix}}_{\tilde{u}^h} = \underbrace{\begin{bmatrix} f^h \\ 0 \end{bmatrix}}_{\tilde{f}^h}, \tag{B.18}$$

where b_h is a row vector of length N with all of its components set to 1 (i.e., the summation operator), and λ^h is the Lagrange multiplier. Provided b_h is not orthogonal to the left null space of L_h (which is likely to be true because of the compatibility condition for the continuous problem), this constrained system will have a unique solution [45, Lemma 5.6.1]. Furthermore, if this condition holds, the solution will be the same (up to a constant) as the projection approach, since $f^h - \lambda^h b_h^T$ is then necessarily in the range of L_h. We use the third approach in the MGM method.

Some modifications to the two-level cycle are required to handle the constrained

[2]Note that L_h is not symmetric, so the constant vector is not necessarily the left null vector

system (B.18). First, the transfer operators have to also transfer the Lagrange multipler through the fine and coarse levels and the Galerkin coarse grid operator has to include the transferred constraint. We follow the approach from [1] and modify these operators according to the following definitions:

$$\tilde{L}_H = \underbrace{\begin{bmatrix} I_h^H & 0 \\ 0 & 1 \end{bmatrix}}_{\tilde{I}_h^H} \underbrace{\begin{bmatrix} L_h & b_h^T \\ b_h & 0 \end{bmatrix}}_{\tilde{L}_h} \underbrace{\begin{bmatrix} I_H^h & 0 \\ 0 & 1 \end{bmatrix}}_{\tilde{I}_H^h} = \begin{bmatrix} I_h^H L_h I_H^h & I_h^H b_h^T \\ b_h I_H^h & 0 \end{bmatrix}, \tag{B.19}$$

where $\tilde{I}_H^h$ and $\tilde{I}_h^H$ are the modified interpolation and restriction operators, respectively, and $\tilde{L}_H$ is the modified Galerkin operator. These modified transfer operators simply pass the Lagrange multiplier between levels without alteration.

For the smoother of the constrained system (B.18), we use the approach discussed in [45, §5.6.5], where only the solution u_h is smoothed and the constraint is left alone. We again use GS for smoothing u_h and one application of the modified smoother takes the form

$$u^h \leftarrow u^h + B_h^{-1}(f^h - L_h u^h - b_h \lambda^h),$$

where B_h is the same as (B.17). This smoother is equivalent to one iteration of the undamped inexact Uzawa method with the Schur complement set equal to zero [7].

Finally, we use a direct solve to compute the defect e^h and Lagrange multiplier λ^H on the coarse level. This system takes the form $\tilde{L}_H \tilde{e}^H = \tilde{r}^H$, where $\tilde{e}^H = \begin{bmatrix} e^H & \lambda^H \end{bmatrix}^T$ and $\tilde{r}^H$ is the restricted residual for the modified system: $\tilde{r}^H = \tilde{I}_h^H(\tilde{f}^h - \tilde{L}_h \tilde{u}^h)$. When solving a Poisson problem, we use the modifications described above in Algorithm 2.

Algorithm 3: MGM preprocessing phase

1 **Input:** Fine level nodes $\mathbf{X}_1$ and operator L_1; minimum number of coarse level points $N_{\min}$;
2 Re-order rows and columns of L_1 using RCM;
3 Re-order $\mathbf{X}_1$ according to the RCM ordering;
4 Compute number of levels: $p = \lfloor \log(N_1/N_{\min})/\log(4) \rfloor + 1$;
5 **for** $j = 1 \dots p-1$ **do**
6 Generate coarse point cloud $\mathbf{X}_{j+1}$ with $N_{j+1} = \lfloor N_1/4^j \rfloor$ points ;
7 Generate interpolation operator I_{j+1}^j from $\mathbf{X}_{j+1}$ to $\mathbf{X}_j$;
8 Set the restriction operator to $I_j^{j+1} = (I_{j+1}^j)^T$;
9 Generate Galerkin coarse level operator $L_{j+1} = I_{j+1}^j L_j I_j^{j+1}$;
10 Re-order rows and columns of L_{j+1} using RCM;
11 Re-order rows of I_{j+1}^j and columns of I_j^{j+1} according to the RCM ordering;
12 **end**
13 Compute sparse LU decomposition of L_p;

B.5.5 Multilevel extension

The multilevel extension of the two-level cycle can be obtained by applying it recursively until a sufficiently coarse level is reached to make a direct solver practical. To simplify the notation in describing the multilevel cycle, we replace the h/H superscript/subscript notation with a number corresponding to the level, with $j = 1$ being the finest level. For example, for the jth level, $\mathbf{X}_j$ denotes the point cloud, N_j denotes its size, L_j denotes the operator, r^j denotes the residual, and I_j^{j+1} is the restriction to level $j+1$.

Before the multilevel cycle begins, we compute all the coarse point clouds, transfer operators, and Galerkin coarse level operators in a preprocessing step, which is outlined in Algorithm 3. The number of levels, p, depends on the number of fine level nodes and minimum number of nodes on the coarsest level, $N_{\min}$, and is determined on line 4 of this algorithm. This guarantees that the number of nodes on the coarsest

level satisfies $N_{\min} \leq N_p < 4N_{\min}$. We note that when using WSE to generate the coarse point cloud $\mathbf{X}_j$ on line 6 of the preprocessing algorithm, we use the finer point cloud $\mathbf{X}_{j+1}$, rather than the finest node $\mathbf{X}_1$. This reduces the cost in performing this step.

The multilevel cycle is outlined in Algorithm 4 in non-recursive form. This algorithm is what we call the MGM method and corresponds to a traditional V-cycle in multigrid methods, which is typically denoted V(ν_1, ν_2) corresponding to the number of pre-/post smoothing operations. Other cycling methods can also be used (e.g., F- or W-cycle [45, §2.4]), but we limit our focus to the V-cycle. While this algorithm is described for a shifted Poisson problem, it can be easily modified for solving a Poisson problem following the modifications discussed in Section B.5.4.

Algorithm 4: MGM V(ν_1, ν_2)-cycle

Input: Right hand side f^1; Initial guess u^1; Number levels p; $\{L_j\}_{j=1}^{p-1}$; $\{I_{j+1}^j\}_{j=1}^{p-1}$; $\{I_j^{j+1}\}_{j=1}^{p-1}$;
RCM re-orderings; Sparse LU factorization of L_p;
Re-order f^1 and u^1 according to RCM re-ordering of L_1;
Presmooth initial guess: $u^1 \leftarrow \text{presmooth}(L_1, u^1, f^1, \nu_1)$;
Compute/restrict residual: $r^1 = I_1^2(f^1 - L_1 u^h)$
for $j = 2 \ldots p-1$ **do**
    Presmooth defect: $e^j = \text{presmooth}(L_j, 0, r^j, \nu_1)$;
    Compute/restrict residual: $r^{j+1} = I_j^{j+1}(r^j - L_j e^j)$;
end
Compute defect: Solve $L_p e^p = r^p$ using sparse LU decomposition of L_p ;
for $j = p-1, \ldots, 2$ **do**
    Interpolate/correct defect: $e^j \leftarrow e^j + I_{j+1}^j e^{j+1}$;
    Post smooth defect: $e^j \leftarrow \text{postsmooth}(L_j, e^j, r^j, \nu_2)$;
end
Interpolate defect/correct approximation: $u^1 \leftarrow u^1 + I_2^1 e^2$;
Post smooth approximation: $u^1 \leftarrow \text{postsmooth}(L_1, u^1, f^1, \nu_2)$;
Undo re-ordering of u^1 from RCM of re-ordering of L_1;

B.5.6 Preconditioner for Krylov subspace methods

The MGM method has the benefit of being relatively straightforward to implement. However, as shown in the numerical experiments in the next section, it may converge slowly when using it as a standalone solver, especially for higher order discretizations of the LBO on more irregular point clouds. A common approach to bypassing these issues for standard geometric and algebraic multigrid methods is to combine them with a Krylov subspace method (e.g., [45, §7.8] or [33, 18]). In this case, multigrid is viewed as preconditioner for the Krylov method. We also take this approach with MGM, using it a preconditioner for two Krylov methods: generalized minimum residual (GMRES) and bi-conjugate gradient stabilized (BiCGSTAB) [37]. This combination appears to result in an efficient and robust method for solving the discretized surface Poisson and shifted Poisson equations on quite complicated surface.

B.6 Numerical Results

In this section, we analyze the MGM method as a solver and preconditioner for the Poisson and shifted Poisson problem on two surfaces: the unit sphere and the cyclide. The latter is shown in Figure B.2 and the implicit equation describing the surface is given in [26]. We test the method on both RBF-FD and GFD discretizations using the parameters given in the first part of Table B.1. In all the tests, we are interested in how the method scales to higher order discretizations, and thus give results for polynomial degrees $\ell = 3$, 5, and 7, which correspond to approximately second, fourth, and sixth order accuracy, respectively [41, 43]. For the sphere tests, we generate the point clouds from the vertices of icosahedral node sets, which are used extensively in numerical weather prediction [**?**]. For the cyclide, we use point

Variable	Description	Value(s)
Parameters for the discretization the LBO		
ℓ	Poly. degree for discretizing the LBO with RBF-FD or GFD	3, 5, or 7
k	Order of the PHS kernel for discretizing the LBO with RBF-FD	ℓ
α	Weighting parameter for the Gaussian kernel in GFD	4 or 5
n	Stencil size for discretizing the LBO with RBF-FD or GFD	$\lceil(\ell+1)(\ell+2)\rceil$
Parameters for MGM		
$N_{\min}$	Minimum number of nodes on the coarsest level	250
B_h	Pre- and post-smoother (see (B.17))	Forward GS
ν_1, ν_2	Number of applications of the pre- and post-smoother	1
m	Stencil size of the interpolation/restriction operators	3

Table B.1: Description of parameters and their values used in the numerical results.

clouds produced from Poisson disk sampling of the surface. This latter approach results in much more unstructured point clouds than the sphere case (see Figure B.2 for an illustration).

Unless otherwise specified, the parameters of the MGM method are set according to those given in the second part of Table B.1. We tested the method with different combinations of these parameters and found that the ones listed in the table generally gave the best results in terms of iteration count and wall-clock time. Additionally, when using MGM with Krylov methods, we use it as a right preconditioner, which is generally recommended [18]. Finally, all the MGM results presented were obtained from a MATLAB implementation of the method, with a MEX interface to the WSE method, which is implemented in C++.

In the first several experiments, we compare MGM to AMG, as implemented in the Python package PyAMG [32]. In addition to being very popular blackbox

solvers and preconditioners for a wide range of problems, AMG methods have been used previously for solving linear systems associated with meshfree discretizations of elliptic PDEs in the plane [38] and on surfaces [20]. We use the smoothed aggregation version of AMG, as we found it performed better than classical AMG. Additionally, we use one application of symmetric GS as the pre- and post-smoother, a V-cycle for the multilevel cycle, and sparse LU for the coarse level solver. We experimented with other combinations of parameters and again found these generally gave the best results in terms of iteration count and wall-clock time. Additionally, when using PyAMG with GMRES, we use it as a right preconditioner (with the fgmres option), while for BiCGSTAB we use it as a left preconditioner (as this is the only option). Finally, in the comparisons with AMG, we focus on the shifted Poisson problem as PyAMG does not offer a specialized way to deal with the constrained system (B.18).

B.6.1 Standalone solver vs. preconditioner

In the first set of tests, we compare MGM and PyAMG both as standalone solvers and preconditioners. For the latter approaches we refer to these solvers as MGM GMRES, MGM BiCGSTAB, PyAMG GMRES, and PyAMG BiCGSTAB, to indicate the type of Krylov method employed. We use these solvers on the shifted Poisson problem on the unit sphere and cyclide with N_h=2,621,422 and N_h=2,097,152 nodes, respectively. For the BiCGSTAB results, we count the number of applications of the preconditioner as the iterations since each step of this method applies the preconditioner twice, whereas GMRES applies it once.

Figure B.3 displays the results in terms of relative residual vs. iteration count for RBF-FD, while Figure B.4 displays the results for GFD. For the RBF-FD results, we see that the methods using MGM converge more rapidly than the methods based on

PyAMG for both surfaces. For the sphere, MGM works very well as a standalone solver and preconditioner even as ℓ increases, but for the cyclide the convergence rates of MGM as a standalone solver decrease considerably. This may be due to the more irregular nature of the cyclide point cloud. We note, however, that the preconditioned versions of MGM only have a very mild decrease in convergence rates for the cyclide. The figures also show that the methods using PyAMG do not converge as rapidly as the corresponding MGM methods, with the fastest converging PyAMG method taking more than double the number of iterations as the fastest MGM method when $\ell = 3$ and triple when $\ell = 5$ and 7. We see similar patterns in the GFD results, but the methods based on both MGM and PyAMG generally converge faster in this case and the gap between the fastest converging MGM and PyAMG methods is not as wide. Finally, we note that MGM BiCGSTAB seems to converge at a very similar rate to MGM GMRES, whereas this does not hold for PyAMG. This is a promising result for large systems since the storage requirements of BiCGSTAB are fixed, whereas they grow with the size of the Krylov subspace for GMRES [18].

These experiments also indicate that, while MGM can be an effective standalone solver for small ℓ (lower order discretizations), it is more robust for larger ℓ (higher order discretizations) and when used as a preconditioner. This also seems to be the case when applying it to different surfaces and point clouds based on regular nodes (like the sphere) and irregular nodes (like the cyclide). From the PyAMG results, it is clear that it should be used as a preconditioner to get the most robust results, which is generally the case for AMG methods applied to nonsymmetric systems [18].

Sphere						
	PyAMG			MGM		
N	$\ell = 3$	$\ell = 5$	$\ell = 7$	$\ell = 3$	$\ell = 5$	$\ell = 7$
10242	30 (38)	39 (56)	46 (70)	18 (19)	18 (18)	18 (20)
40962	35 (42)	43 (54)	53 (78)	19 (19)	19 (20)	19 (20)
163842	40 (50)	49 (64)	59 (84)	19 (21)	19 (20)	19 (20)
655362	45 (62)	55 (80)	68 (90)	20 (20)	20 (20)	20 (21)
2621442	52 (66)	64 (102)	75 (94)	20 (21)	20 (22)	20 (22)
Cyclide						
	PyAMG			MGM		
N	$\ell = 3$	$\ell = 5$	$\ell = 7$	$\ell = 3$	$\ell = 5$	$\ell = 7$
8192	25 (30)	34 (48)	42 (62)	17 (19)	19 (20)	20 (22)
32768	32 (40)	39 (48)	46 (64)	19 (20)	22 (22)	24 (26)
131072	35 (46)	45 (66)	54 (74)	20 (21)	23 (26)	28 (31)
524288	41 (50)	49 (68)	60 (78)	21 (22)	25 (27)	30 (39)
2097152	45 (54)	56 (78)	65 (90)	20 (21)	24 (25)	27 (29)

Table B.2: Comparison of the number of PyAMG and MGM preconditioned GMRES/BiCGSTAB iterations required to reach a relative residual tolerance of 10^{-12} for solving the shifted Poisson problem with RBF-FD discretizations. The numbers not in parenthesis are for GMRES, while the numbers in parenthesis are for BiCGSTAB.

B.6.2 Scaling with problem size

In the next set of tests, we examine how both the MGM and PyAMG methods scale as the size of the point clouds N_h increases. We focus on the preconditioned versions of these methods and test them again on the sphere and cyclide. Tables B.2 and B.3 display the results for the RBF-FD and GFD methods, respectively, in terms of number of iterations required to reach a relative residual of 10^{-12}. We see from these tables that the preconditioned MGM methods appear to scale much better than the PyAMG methods, both in terms of N_h and ℓ. For RBF-FD discretizations, the increase in the iteration count for MGM is more mild with increasing ℓ than for GFD. However, in all cases but $\ell = 7$ on the sphere, the iteration count is lower for the

GFD discretizations; we examine this further in Section B.6.4.

Sphere						
	PyAMG			MGM		
N	$\ell = 3$	$\ell = 5$	$\ell = 7$	$\ell = 3$	$\ell = 5$	$\ell = 7$
10242	15 (18)	20 (26)	26 (34)	10 (10)	14 (14)	20 (21)
40962	17 (20)	23 (26)	30 (42)	10 (11)	15 (16)	23 (25)
163842	20 (22)	27 (36)	34 (42)	11 (12)	15 (17)	26 (29)
655362	23 (26)	30 (42)	39 (48)	12 (13)	16 (17)	27 (28)
2621442	27 (30)	35 (46)	43 (60)	13 (15)	16 (17)	27 (29)
Cyclide						
	PyAMG			MGM		
N	$\ell = 3$	$\ell = 5$	$\ell = 7$	$\ell = 3$	$\ell = 5$	$\ell = 7$
8192	13 (16)	17 (20)	23 (28)	10 (11)	14 (14)	18 (19)
32768	16 (20)	21 (24)	26 (30)	12 (12)	17 (18)	24 (26)
131072	19 (24)	24 (32)	30 (36)	12 (13)	18 (21)	25 (27)
524288	22 (26)	27 (34)	34 (40)	13 (14)	18 (18)	26 (29)
2097152	25 (30)	32 (40)	39 (50)	14 (14)	17 (18)	24 (26)

Table B.3: Same as Table B.2, but for GFD discretizations. For these results, we set $\alpha = 4$ for all N_h, but the largest, where we set $\alpha = 5$.

In Figure B.5 we display the wall-clock times for the results in Tables B.2 and B.3 for GMRES PyAMG and MGM. These results were run on a Linux Workstation with Intel i9-9900X 3.5 GHz processor (with no explicit parallelization) and do not include the set-up times. We see from Figure B.5 that MGM has a lower wall-clock time than PyAMG for all but the first N_h in the case of the sphere. Furthermore, for the largest N_h, MGM is between 3 and 5 times faster. Additionally, the dotted line in these scaling plots marks perfect linear scaling and we see that the results for both methods have a very similar slope to this line. Finally, we note that the timing results for BiCGSTAB follow a similar trend to GMRES, so we omitted displaying the results. However, the gap between the MGM and PyAMG timings were larger in this case.

B.6.3 Spectrum analysis

The previous two sections showed the preconditioned MGM methods outperforming the PyAMG methods. To better understand these results, we investigate the spectrum (eigenvalues) of the preconditioned matrices from both methods. Letting M_h denote the matrix representation for applying one V-cycle of either MGM or PyAMG, we can write the (right) preconditioned system as $L_h M_h z^h = f^h$, where $z^h = (M_h)^{-1} u^h$. The convergence behavior of Krylov methods can be understood by analyzing the spectrum $L_h M_h$. As discussed in, for example [34], the more clustered this spectrum is to one, the faster the Krylov methods will converge. In Figure B.6 we display the complete spectrum of the preconditioned matrix $L_h M_h$ of both MGM and PyAMG for the RBF-FD discretizations on the sphere and cyclide. Due to the cost of this eigenvalue computation, we were only able to compute the results for with $N_h = 10242$ and 8192, respectively. We see from the figure that spectra for MGM are more clustered around one than PyAMG for both surfaces and increasing ℓ, which explains the better iteration counts in Table B.2. We omit the results for GFD, but note that the spectra were similar to RBF-FD, but were even more clustered near one.

B.6.4 Iteration vs. accuracy

In the final set of tests we focus on solving the (discretized) Poisson problem with MGM GMRES and examine how the accuracy of the RBF-FD and GFD discretizations depend on the iteration count for increasing N_h and ℓ. We restrict our attention to the sphere, for which it is easy to construct test problems with exact solutions based on spherical harmonics. For the test problem in the experiments, we use the Y_5^4 spherical harmonic, which can be written in Cartesian coordinates as

$Y_5^4(x,y,z) = z(x^4 - 6x^2y^2 + y^4)$. We fix the number of iterations of MGM GMRES for solving the discretized systems to $1, 5, 10, 15, 20, 25, 30$, and compute both the relative residual and relative errors (in the 2-norm) in the approximate solutions. Figure B.7 displays the results from these experiments. We see from the figure that in almost all cases the minimum error for either the RBF-FD and GFD is reached before the minimal residual is reached. Additionally, the results indicate that while the residuals for GFD converge faster than RBF-FD, the errors for a given N_h and ℓ are smaller for RBF-FD. So the cost per error for both methods is much more comparable than the previous experiments indicated and favor RBF-FD.

B.7 Applications

In this section we demonstrate the performance of MGM on three different applications involving complicated surfaces represented by relatively large point clouds; see Figure B.8. All these applications involve solving discrete (shifted) surface Poisson problems, for which we use the RBF-FD method to approximate the LBO and MGM GMRES to solve the resulting linear systems.

B.7.1 Surface harmonics

We first consider approximating the first several eigenvalues and eigenfunctions of the LBO on the Chinese Guardian Lion model. The eigenfunctions of the LBO or the "surface harmonics" have been used in various applications in data analysis. For example, Reuter et. al. [36] used the low frequency surface harmonics for shape segmentation and registration.

The LBO eigenvalue problem is given as $\Delta_{\mathcal{M}} u = \lambda u$. To approximate the solutions of this problem we use the RBF-FD method with $\ell = 5$ to approximate the LBO and

ARPACK [25] (accessed through the `eigs` function in MATLAB) to solve the discrete system for the first several eigenpairs that are smallest in magnitude. ARPACK uses the Arnoldi method on the shifted inverse of a matrix to find the eigenpairs closest to the shift σ, which, for the surface problem, requires a routine for repeatedly solving systems of the form $(L_h - \sigma I_h)v^h = f^h$, for different f^h. We use MGM GMRES to solve these linear systems with $\sigma = -1$ and set the tolerance to 10^{-10}. Figure B.9 displays the first 10 non-zero harmonics computed with this technique. The ARPACK routine used 49 linear system solves to determine the eigenpairs; the median number of MGM GMRES iterations required to solve these systems was only 15 and the max was 16.

B.7.2 Pattern formation

We next consider solving two coupled reaction-diffusion (RD) equations on the Stanford Bunny model. These types of equations arise, for example, in phenomenological models of color patterns in animal coats [31]. We consider the Gierer-Meinhardt two-species RD system [19] given as follows:

$$\frac{\partial u}{\partial t} = D_u \Delta_{\mathcal{M}}\, u + A - Bu + \frac{u^2}{v(1+Cu^2)}, \tag{B.20a}$$

$$\frac{\partial v}{\partial t} = D_v \Delta_{\mathcal{M}}\, v + u^2 - v. \tag{B.20b}$$

By altering the parameters A, B, C, D_u, and D_v appropriately, this system can produce solutions that converge to spot or labyrinth patterns at "steady-state" [31]. For the bunny model, we set $A = 0.08$, $B = 1.5$, $C = 0.45$, $D_u = 5 \times 10^{-5}$, and $D_v = 10^{-3}$ to produce the labyrinth pattern. We use a random initial condition, where at each point in X_h the values of u and v are selected from a uniformly random

distribution in the interval $[0, 1]$.

To approximate the solution of (B.20) we use the RBF-FD method with $\ell = 3$ to approximate the LBO and apply the second-order accurate semi-implicit backward difference scheme (SBDF2) [3] as the time-stepping method that treats the diffusion implicitly and reactions explicitly. We set the time-step to $\Delta t = 0.05$. The temporal discretization results in two decoupled (discrete) screened Poisson problems that need to be solved at each time-step for which we use GMRES preconditioned with MGM. For the GMRES method we set the tolerance on the relative residual to 10^{-8} and use the previous time-step as the initial guess. We set the final integration to 300 time units, which resulted in a near steady-state pattern. Figure B.10 displays the results of the simulations. Included in the figure are the iterations required by the preconditioned GMRES method as a function of time. We see from the figure that the maximum iteration count is 6 for the u variable and 11 for the v variable, and decreases to 2 and 3, respectively as the solutions approach steady-state. The larger iteration count for the v variable is expected since the diffusion coefficient is larger in (B.20b).

B.7.3 Geodesic distance

Lastly, we consider the classic problem of approximating the geodesic distance from a given point on a surface to all other points. We use the *heat method* introduced by Crane et. al. [11] to solve this problem. This method transforms the non-linear geodesic distance problem, typically formulated in terms of the eikonal equation, into solving a pair of linear parabolic and elliptic problems. The heat method is comprised of the three steps:

1. Solve $u_t = \Delta_{\mathcal{M}} u$, with $u_0 = \delta(\mathbf{x}^*)$, to some time $t_{\text{final}} > 0$

2. Compute the vector field $\boldsymbol{\eta} = -\nabla_{\mathcal{M}} u / |\nabla_{\mathcal{M}} u|$

3. Solve the Poisson problem $\Delta_{\mathcal{M}} \varphi = \nabla_{\mathcal{M}} \cdot \boldsymbol{\eta}$

Here $\mathbf{x}^* \in$ denotes the target point on the surface $\mathcal{M}$ to compute the distance from, $\nabla_{\mathcal{M}}$ denotes the surface gradient, $\nabla_{\mathcal{M}}\cdot$ is the surface divergence, and δ denotes the Dirac delta function. As discussed in [11], the function φ approximates the geodesic distance and converges to the exact distance as $t_{\text{final}} \to 0$.

We apply the heat method on the Armadillo model. We again use the RBF-FD method with $\ell = 3$ to approximate the LBO in steps 1 and 3 above. To approximate the surface gradient and divergence, we also use the RBF-FD method formulated in the tangent plane similar to the method described in [43] for GFD. For these approximations, we use $\ell = 2$, which result in a second-order approximation. We discretize the heat equation in the first step with backward Euler in time with a time-step of $\Delta t = 10^{-3}$ and set $t_{\text{final}} = 3\Delta t$. To solve the linear systems associated with this implicit discretization and the system from the discretized Poisson equation in step 3, we use GMRES preconditioned with MGM, setting the tolerance to 10^{-8}. The results for a point $\mathbf{x}^*$ on the chest of the Armadillo are displayed in the first to images of Figure B.11. The last image in this figure displays the iterations of the preconditioned GMRES method for solving the systems from the heat equation discretization for three time-steps and the Poisson system to determine φ. We see that the iteration count remains low for all these systems.

B.8 Concluding remarks

We have presented a new geometric multilevel method, MGM, for solving linear systems associated with discretizations of elliptic PDEs on point clouds. The method

is entirely meshfree and uses the WSE algorithm for coarsening the point clouds, interpolation/restrictions operators based on polyharmonic spline RBFs, Galerkin coarsening of the operator, and standard smoothers. All of these choices make MGM particularly straightforward to implement. We numerically analyzed the method as a standalone solver and preconditioner on test problems for the sphere and cyclide discretized using RBF-FD and GFD methods, and found that it compares favorably to AMG methods in terms of convergence rates and wall-clock time. When using MGM as a preconditioner, we also found that it scaled well as both the problem size and accuracy of the discretizations increased. Finally, we demonstrated that the method can be used in three challenging applications involving large systems of equations.

There are several extensions of MGM that we plan to pursue in the future. One is to test the method on other discretizations. MGM is agnostic to the underlying discretization and could be used even for (nodal) mesh-based discretizations. Here the nodal points of the mesh could be treated as a point cloud and WSE could be applied, or if there is a natural way to coarsen the mesh, then this could be used instead. A second idea we plan to pursue is extending MGM to domains with boundaries, which in principle should be straightforward. Finally, we plan to look into parallel implementations of the method to further improve the performance.

Acknowledgements

AMJ and GBW's work was partially supported by US NSF grants CCF-1717556 and DMS-1952674. VS's work was partially supported by US NSF grants CCF-1714844. We benefitted from discussions with Drs. Cem Yuksel (on the WSE algorithm) and Hari Sundar (on multigrid methods), both from the University of Utah. The Stanford Bunny and Armadillo data were obtained from the Stanford University 3D Scanning

Repository. The Chinese Guardian Lion data was obtained from the AIM@SHAPE-VISIONAIR Shape Repository.

REFERENCES

[1] Adams, Mark F. 2004. Algebraic multigrid methods for constrained linear systems with applications to contact problems in solid mechanics. *Numerical Linear Algebra with Applications*, **11**, 141–153.

[2] Álvarez, Diego, González-Rodríguez, Pedro, and Kindelan, Manuel. 2021. A Local Radial Basis Function Method for the Laplace–Beltrami Operator. *J. Sci. Comput.*, **86**(3), 28.

[3] Ascher, Uri M., Ruuth, Steven J., and Wetton, Brian T. R. 1997. Implicit-Explicit Methods For Time-Dependent PDEs. *SIAM J. Num. Anal*, **32**, 797–823.

[4] Ballard, Grey, Siefert, Christopher, and Hu, Jonathan. 2016. Reducing Communication Costs for Sparse Matrix Multiplication within Algebraic Multigrid. *SIAM J. Sci. Comput.*, **38**(3), C203–C231.

[5] Bayona, Victor, Flyer, Natasha, Fornberg, Bengt, and Barnett, Gregory A. 2017. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs. *J. Comput. Phys.*, **332**, 257–273.

[6] Bell, Nathan, Dalton, Steven, and Olson, Luke N. 2012. Exposing Fine-Grained Parallelism in Algebraic Multigrid Methods. *SIAM J. Sci. Comput.*, **34**(4), C123–C152.

[7] Benzi, Michele, Golub, Gene H, and Liesen, Jörg. 2005. Numerical solution of saddle point problems. *Acta Numerica*, **14**, 1–137.

[8] Bertalmío, M., Cheng, L., Osher, S., and Sapiro, G. 2001. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, **174**, 759–780.

[9] Bridson, Robert. 2007. Fast Poisson Disk Sampling in Arbitrary Dimensions. *In: ACM SIGGRAPH 2007 Sketches*. SIGGRAPH '07. New York, NY, USA: ACM.

[10] Chen, Yujia, and Macdonald, Colin B. 2015. The closest point method and multigrid solvers for elliptic equations on surfaces. *SIAM Journal on Scientific Computing*, **37**(1), A134–A155.

[11] Crane, Keenan, Weischedel, Clarisse, and Wardetzky, Max. 2013. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)*, **32**(5), 1–11.

[12] Davydov, Oleg, and Schaback, Robert. 2019. Optimal stencils in Sobolev spaces. *IMA J. Num. Analy.*, **39**(1), 398–422.

[13] Demanet, Laurent. 2006. Painless, highly accurate discretizations of the Laplacian on a smooth manifold. *Technical report, Stanford University*.

[14] Duchon, J. 1977. Splines minimizing rotation-invariant semi-norms in Sobolev space. *Pages 85–100 of:* Schempp, W., and K., Zeller (eds), *Constructive Theory of Functions of Several Variables. Lecture Notes in Mathematics (vol 571)*. Berlin: Springer.

[15] Dziuk, G. 1988. Finite elements for the Beltrami operator on arbitrary surfaces. *In:* Hildebrandt, S., and Leis, R. (eds), *Partial Differential Equations and Calculus of Variations*. Lecture Notes in Mathematics, vol. 1357. Berlin: Springer.

[16] Dziuk, Gerhard, and Elliott, Charles M. 2013. Finite element methods for surface PDEs. *Acta Numerica*, **22**, 289–396.

[17] Fuselier, E, and Wright, G B. 2012. Scattered Data Interpolation on Embedded Submanifolds with Restricted Positive Definite Kernels: Sobolev Error Estimates. *SIAM J. Numer. Anal.*, **50**(3), 1753–1776.

[18] Ghai, Aditi, Lu, Cao, and Jiao, Xiangmin. 2018. A comparison of preconditioned Krylov subspace methods for large-scale nonsymmetric linear systems. *Numerical Linear Algebra with Applications*, **26**(10).

[19] Gierer, Alfred, and Meinhardt, Hans. 1972. A theory of biological pattern formation. *Kybernetik*, **12**(1), 30–39.

[20] Gross, B. J., Trask, N., Kuberry, P., and Atzberger, P. J. 2020. Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: A Generalized Moving Least-Squares (GMLS) approach. *J. Comput. Phys.*, **409**, 109340.

[21] Gunderman, David, Flyer, Natasha, and Fornberg, Bengt. 2020. Transport schemes in spherical geometries using spline-based RBF-FD with polynomials. *J. Comput. Phys.*, **408**, 109256.

[22] Katz, Aaron, and Jameson, Antony. 2009. Multicloud: Multigrid convergence with a meshless operator. *J. Comput. Phys.*, **228**(14), 5237–5250.

[23] Klasing, K., Althoff, D., Wollherr, D., and Buss, M. 2009. Comparison of surface normal estimation methods for range sensing applications. *Pages 3206–3211 of: IEEE International Conference on Robotics and Automation.*

[24] Landsberg, Christoph, and Voigt, Axel. 2010. A multigrid finite element method for reaction-diffusion systems on surfaces. *Computing and visualization in science*, **13**(4), 177–185.

[25] Lehoucq, Richard B, Sorensen, Danny C, and Yang, Chao. 1998. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. Philadelphia, PA, USA: SIAM.

[26] Lehto, E, Shankar, V, and Wright, G. B. 2017. A radial basis function (RBF) compact finite difference (FD) scheme for reaction-diffusion equations on surfaces. *SIAM J. Sci. Comput.*, **39**, A219–A2151.

[27] Liang, Jian, and Zhao, Hongkai. 2013. Solving Partial Differential Equations on Point Clouds. *SIAM J. Sci. Comput.*, **35**(3), A1461–A1486.

[28] Liu, Jinghui, Totz, Jan F., Miller, Pearson W., Hastewell, Alasdair D., Chao, Yu-Chen, Dunkel, Jörn, and Fakhri, Nikta. 2021. Topological braiding and virtual particles on the cell membrane. *Proc. Natl. Acad. Sci. U.S.A.*, **118**(34).

[29] MacDonald, C. B., and Ruuth, S. J. 2009. The implicit closest point method for the numerical solution of partial differential equations on surfaces. *SIAM J. Sci. Comput.*, **31**, 4330–4350.

[30] Manteuffel, Tom, and Southworth, Ben S. 2019. Convergence in Norm of Non-symmetric Algebraic Multigrid. *SIAM Journal on Scientific Computing*, **41**(5), S269–S296.

[31] Miyazawa, Seita, Okamoto, Michitoshi, and Kondo, Shigeru. 2010. Blending of animal colour patterns by hybridization. *Nature communications*, **1**(1), 1–6.

[32] Olson, L. N., and Schroder, J. B. 2018. *PyAMG: Algebraic Multigrid Solvers in Python v4.0*. Release 4.0.

[33] Oosterlee, C. W., Wienands, R., Washio, T., and Gaspar, F. J. 2000. The acceleration of multigrid convergence by recombination techniques. *Pages 34–43 of: Multigrid methods, VI (Gent, 1999)*. Lect. Notes Comput. Sci. Eng., vol. 14. Berlin: Springer.

[34] Oosterlee, Cornelis W, and Washio, Takumi. 1998. An evaluation of parallel multigrid as a solver and a preconditioner for singularly perturbed problems. *SIAM Journal on Scientific Computing*, **19**(1), 87–110.

[35] Piret, Cécile, and Dunn, Jarrett. 2016. Fast RBF OGr for solving PDEs on arbitrary surfaces. *AIP Conference Proceedings*, **1776**(1), 070005.

[36] Reuter, Martin, Biasotti, Silvia, Giorgi, Daniela, Patanè, Giuseppe, and Spagnuolo, Michela. 2009. Discrete Laplace–Beltrami operators for shape analysis and segmentation. *Comput. Graph.*, **33**(3), 381–390.

[37] Saad, Y. 2003. *Iterative Methods for Sparse Linear Systems*. Philadelphia: SIAM.

[38] Seibold, Benjamin. 2010. Performance of algebraic multigrid methods for non-symmetric matrices arising in particle methods. *Numerical Linear Algebra with Applications*, **17**(2-3), 433–451.

[39] Shankar, Varun, Wright, Grady B., Kirby, Robert M., and Fogelson, Aaron L. 2014. A Radial Basis Function (RBF)-Finite Difference (FD) Method for Diffusion and Reaction-Diffusion Equations on Surfaces. *J. Sci. Comput.*, **63**(3), 745–768.

[40] Shankar, Varun, Narayan, Akil, and Kirby, Robert M. 2018. RBF-LOI: Augmenting Radial Basis Functions (RBFs) with Least Orthogonal Interpolation (LOI) for solving PDEs on surfaces. *J. Comput. Phys.*, **373**, 722–735.

[41] Shaw, Sage. 2019. *Radial Basis Function Finite Difference Approximations of the Laplace-Beltrami Operator*. M.Phil. thesis, Boise State University, USA. 1587.

[42] Stoop, Norbert, Lagrange, Romain, Terwagne, Denis, Reis, Pedro M, and Dunkel, Jörn. 2015. Curvature-induced symmetry breaking determines elastic surface patterns. *Nature materials*, **14**(3), 337–342.

[43] Suchde, Pratik, and Kuhnert, Jörg. 2019. A meshfree generalized finite difference method for surface PDEs. *Comp. Math. Appl.*, **78**(8), 2789–2805.

[44] Trask, Nathaniel, and Kuberry, Paul. 2020. Compatible meshfree discretization of surface PDEs. *Computational Particle Mechanics*, **7**, 271–277.

[45] Trottenberg, Ulrich, Oosterlee, Cornelis W., and Schüller, Anton. 2001. *Multigrid*. Texts in Applied Mathematics., vol. 33. Academic Press.

[46] Turk, Greg. 1991. Generating textures on arbitrary surfaces using reaction-diffusion. *Comput. Graph.*, **25**(4), 289–298.

[47] Wendland, Holger. 2005. *Scattered data approximation*. Cambridge Monogr. Appl. Comput. Math., vol. 17. Cambridge: Cambridge University Press.

[48] Wendland, Holger, and Künemund, Jens. 2020. Solving partial differential equations on (evolving) surfaces with radial basis functions. *Advances in Computational Mathematics*, **46**, 64.

[49] Wesseling, P. 1992. *An Introduction to Multigrid Methods*. New York: John Wiley & Sons.

[50] Williamson, David L. 2007. The Evolution of Dynamical Cores for Global Atmospheric Models. *J. Meteorol. Soc. Jpn.*, **85B**, 241–269.

[51] Yoon, Myoungho, Yoon, Gangjoon, and Min, Chohong. 2016. On Solving the Singular System Arisen from Poisson Equation with Neumann Boundary Condition. *J. Sci. Comput.*, **69**, 391–405.

[52] Yuksel, Cem. 2015. Sample Elimination for Generating Poisson Disk Sample Sets. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2015)*, **34**(2), 25–32.

[53] Yuksel, Cem. 2021. *cyCodeBase*.

[54] Zamolo, Riccardo, Nobile, Enrico, and Šarler, Božidar. 2019. Novel multilevel techniques for convergence acceleration in the solution of systems of equations arising from RBF-FD meshless discretizations. *J. Comput. Phys.*, **392**, 311–334.

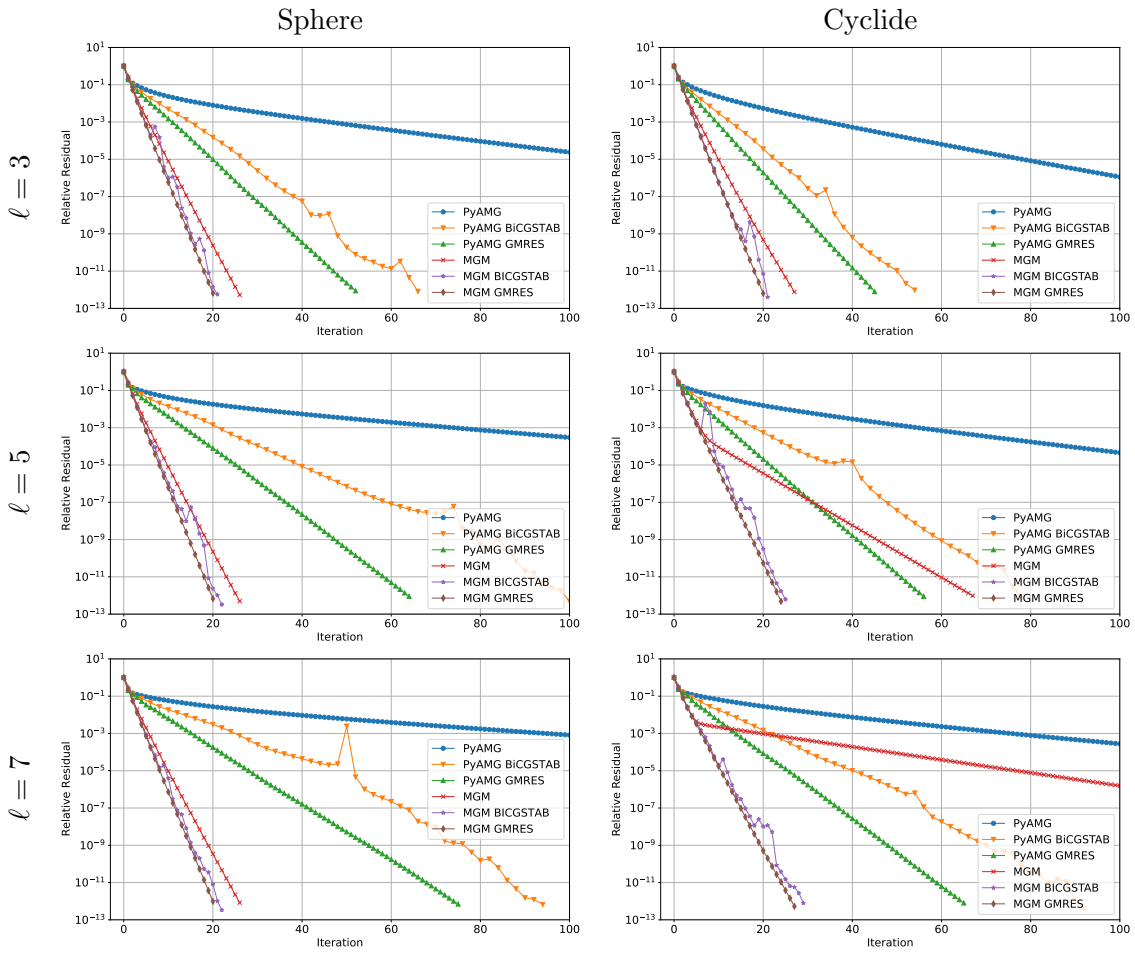


Figure B.3: Convergence results for MGM and PyAMG based solvers for RBF-FD discretizations of a shifted Poisson problem with random right hand side. The sphere results are for $N_h = 2621442$, while for the cyclide $N_h = 2097152$.

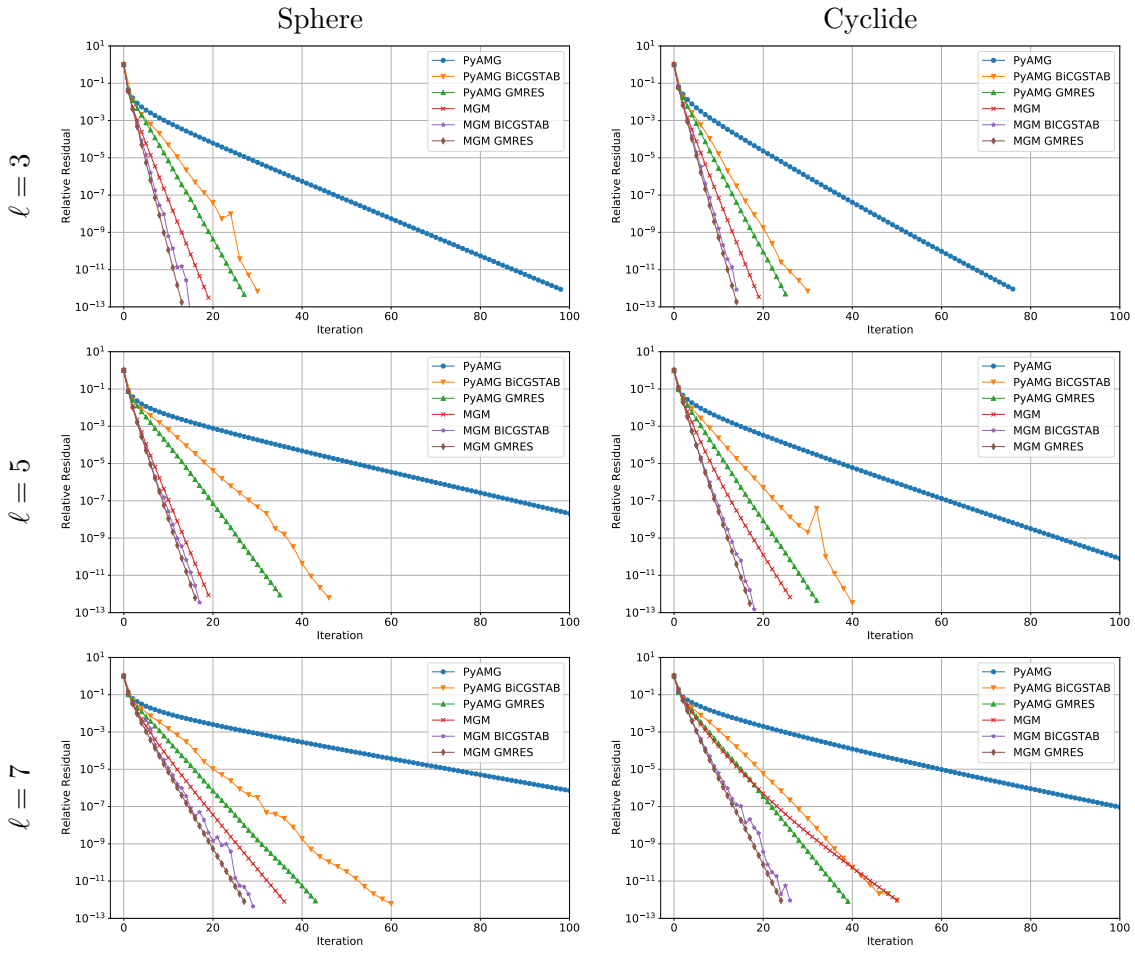


Figure B.4: Same as Figure B.3, but for GFD discretizations of the shifted surface Poisson problem.

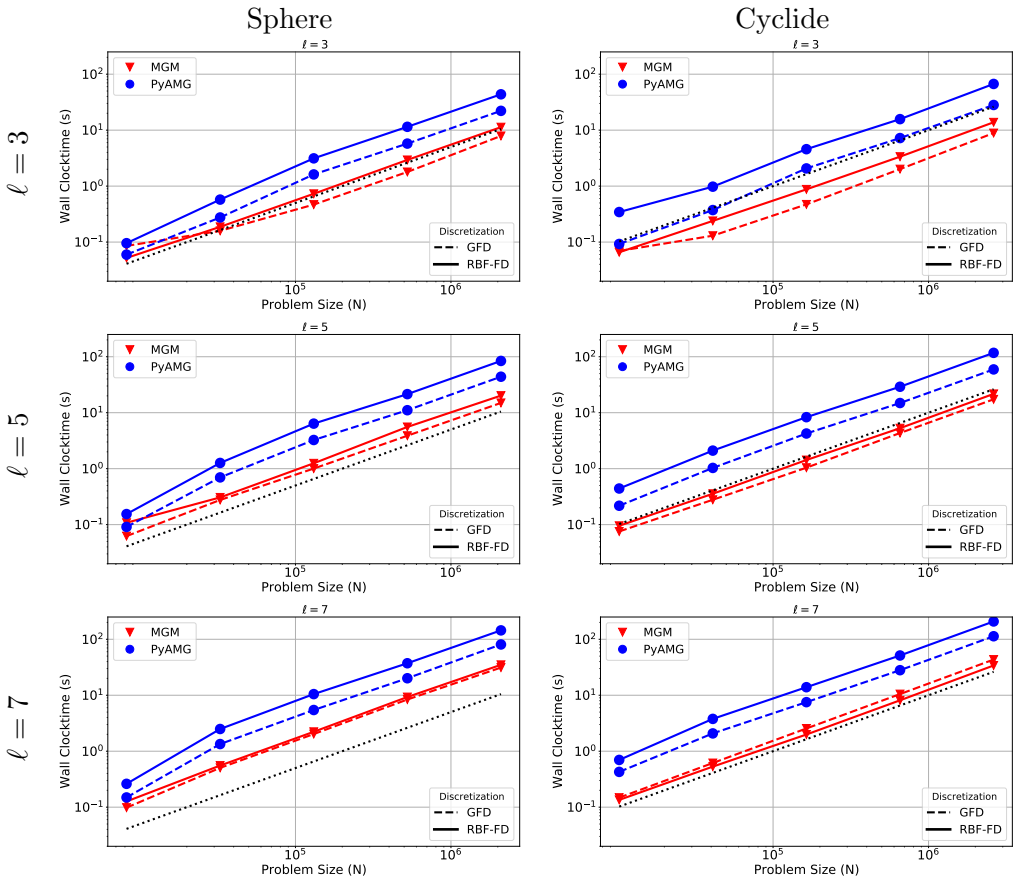


Figure B.5: Wall-clock time (in seconds) for PyAMG GMRES and MGM GMRES to converge to a relative residual of 10^{-12} problem size (N_n) increases for RBF-FD (solid line) and GFD (dashed line) discretizations. The black dotted line marks linear scaling, $\mathcal{O}(N_h)$, for reference.

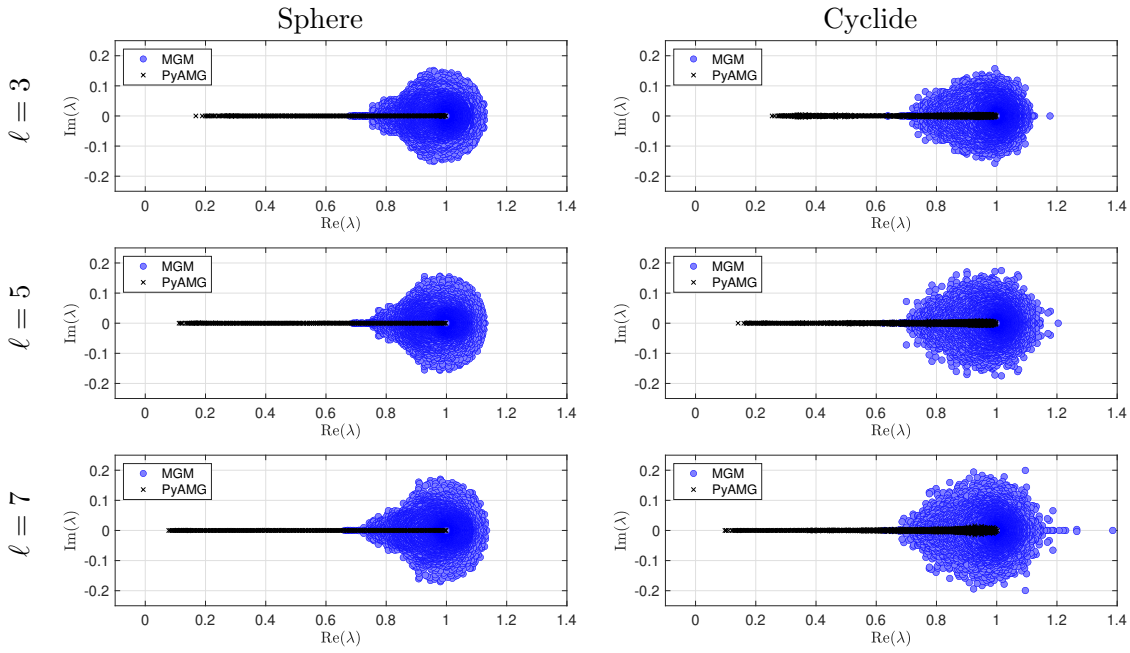


Figure B.6: The spectra of the preconditioned matrix for shifted Poisson problem discretized with RBF-FD.

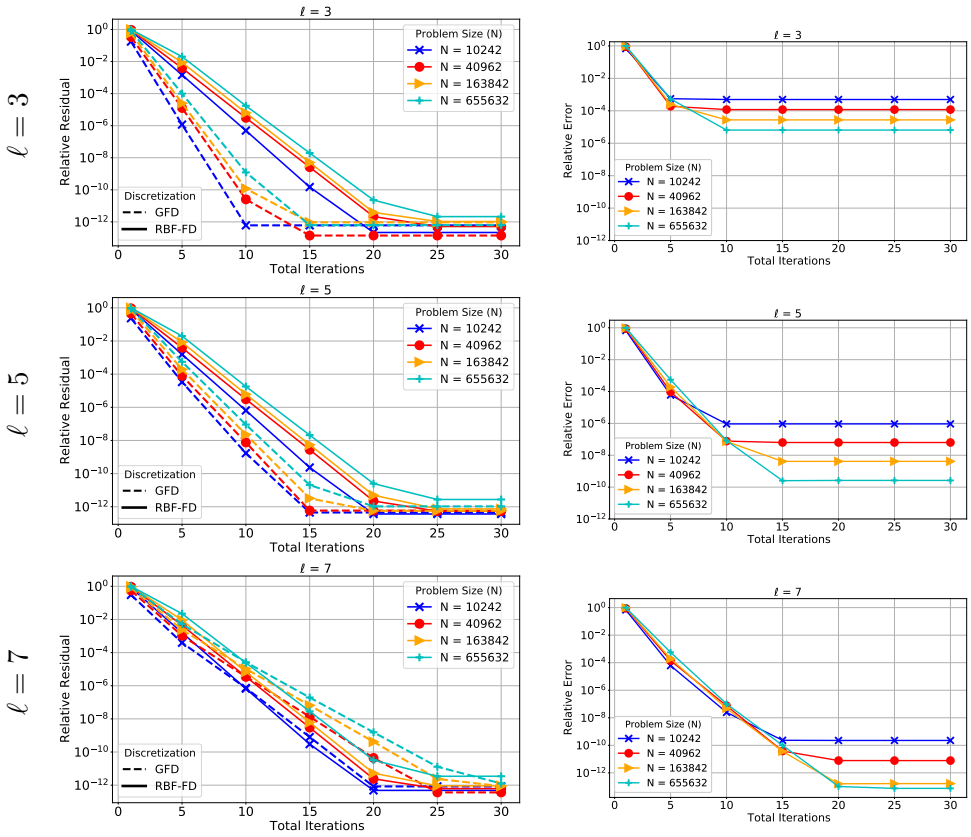


Figure B.7: Relative residuals (left) and relative 2-norm errors (right) for solving a Poisson problem on the sphere with MGM GMRES. Solid lines correspond to RBF-FD discretizations, while dashed lines correspond to GFD.

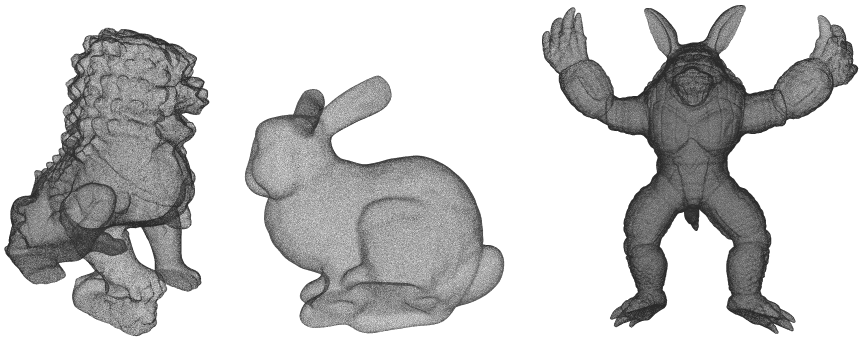


Figure B.8: Point clouds for the surfaces considered in the applications: Chinese Guardian Lion ($N_h = 436605$), Stanford Bunny ($N_h = 291804$), and Armadillo ($N_h = 872773$).

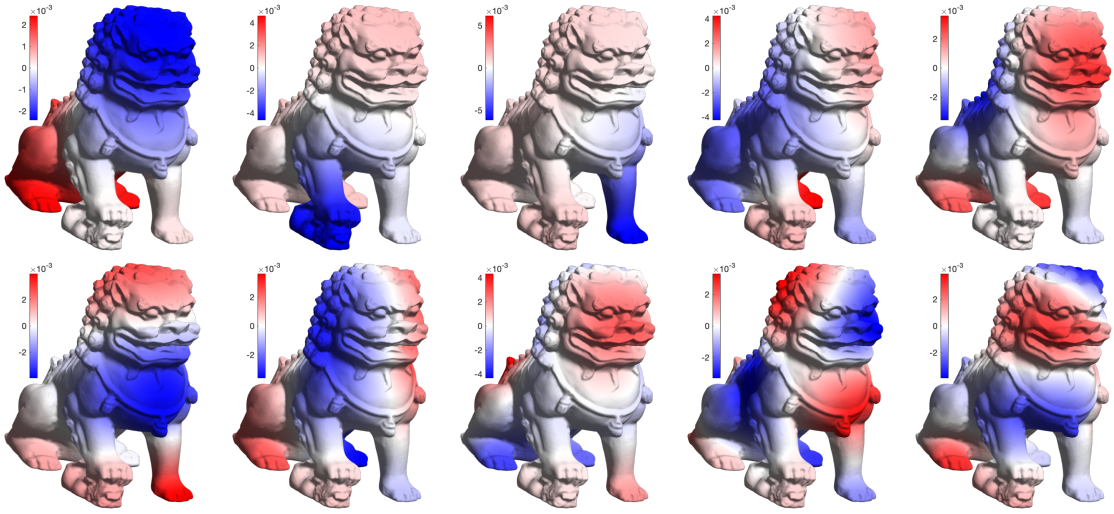


Figure B.9: Left: pseudocolor map of the first 10 non-zero surface harmonics of the Chinese Guardian Lion model.

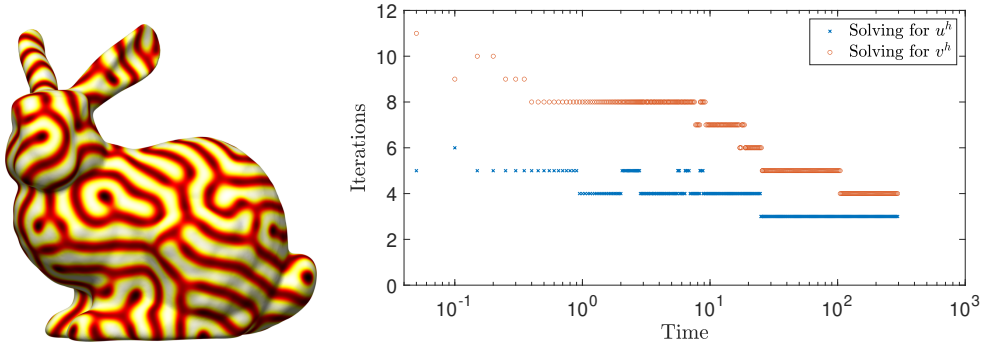


Figure B.10: Left: pseudocolor map of the u variable in the numerical solution of (B.20) on the Stanford Bunny model; the colors transition from white to yellow to red to black, with white corresponding to $u = 0$ and black to $u = 1$. Right: iteration count of MGM GMRES for solving the linear systems associated with u and v variables at each time-step in the semi-implicit scheme for (B.20).

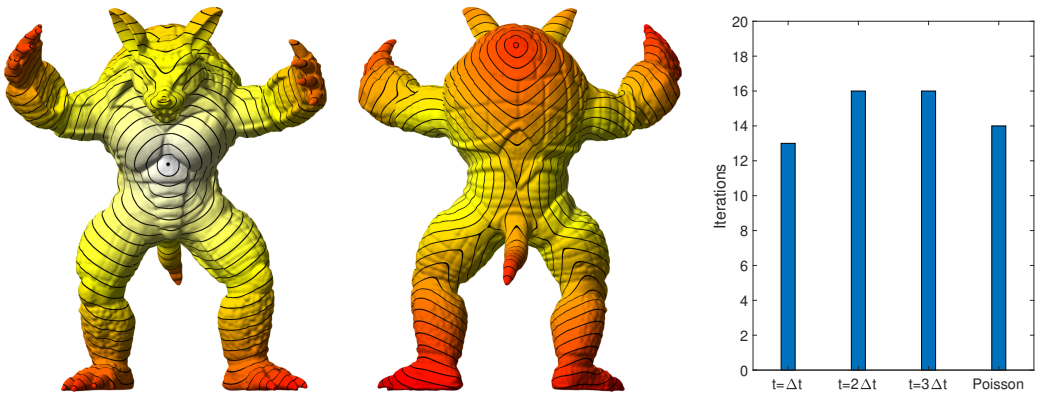


Figure B.11: Left: pseudocolor map of the approximate geodesic distance from the solid circle on the chest of the Armadillo model (viewed from the front and backside) computed with the heat method. Solid black lines mark the contours of the distance field and the colors transition from white to yellow to red with increasing distance from the solid circle. Right: iteration count of GMRES preconditioned with MGM for solving the linear systems associated with the heat method.