RADIAL BASIS FUNCTION FINITE DIFFERENCE APPROXIMATIONS OF THE LAPLACE-BELTRAMI OPERATOR

by

Sage Byron Shaw

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Mathematics

Boise State University

August 2019

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Sage Byron Shaw

Thesis Title: Radial Basis Function Finite Difference Approximations of the Laplace-Beltrami Operator

Date of Final Oral Examination: 21 June 2019

The following individuals read and discussed the thesis submitted by student Sage Byron Shaw, and they evaluated the 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
Varun Shankar, Ph.D.	Co-Chair, Supervisory Committee
Jodi Mead, Ph.D.	Member, Supervisory Committee

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

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisors Grady Wright and Varun Shankar. Their clear instruction, patience, and advice has helped me not only to produce this thesis, but also to properly begin my career in applied mathematics. I would also like to acknowledge the financial support through their awarded grants: NSF-CCF 1717556 and NSF-CCF 1714844. Next, I would like to thank Boise State University and, in particular, the Department of Mathematics, for their invaluable instruction and support.

Special thanks goes to my friends and colleagues in the MS and PhD programs: Daniel, Kathryn, Andrew, and Kennedy. I have always enjoyed long nights in front of whiteboards and computer screens, though never as much as in their company.

Lastly I would like to thank my family. In particular, Kris and my grandparents: Sue, Lary, Patricia, Leo and Marry. Your love and counsel has made all the difference.

ABSTRACT

Partial differential equations (PDEs) are used throughout science and engineering for modeling various phenomena. Solutions to PDEs cannot generally be represented analytically, and therefore must be approximated using numerical techniques; this is especially true for geometrically complex domains. Radial basis function generated finite differences (RBF-FD) is a recently developed mesh-free method for numerically solving PDEs that is robust, accurate, computationally efficient, and geometrically flexible. In the past seven years, RBF-FD methods have been developed for solving PDEs on surfaces, which have applications in biology, chemistry, geophysics, and computer graphics. These methods are advantageous, as they are mesh-free, operate on arbitrary configurations of points, and do not introduce artificial singularities, as surface parameterizations are known to do. In this thesis, we develop a new RBF-FD method that uses projections on the tangent plane to approximate the Laplace-Beltrami operator (surface Laplacian). We then compare this method to two other previously developed RBF-FD methods: the Projected Gradient method and the Hermite RBF-FD method, on a set of benchmark problems posed on the sphere and torus. We also provide guidelines for choosing the various parameters involved in the methods.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Partial differential equations (PDEs) have been used for centuries to model problems in physics, geology, biology, economics, and many other scientific disciplines. Analytic solutions to PDEs are known for only the simplest problems on simple domains such as circles, and rectangles. Some problems, even on relatively simple domains, are proven not to have closed form solutions. Consequentially, there has been extensive research into techniques that can numerically approximate solutions to PDEs.

Finite difference methods are a common and effective technique for numerically solving PDEs. These methods involve finding approximate solutions at discrete points on uniform spatial grids. Therefore, they are limited to fairly simple domains, or at least require modification for each new domain to which they are applied. The computational cost of a given method is dependent on the number of points chosen in the domain, and as such, even modest refinement of grids for problems with relatively few spatial dimensions can be quite costly. These limitations strongly motivate the search for high-order, mesh-free methods, which are both highly-accurate and robust.

In 1990, Kansa [1] developed the so called *multiquadric method* — a radial basis function (RBF) method for numerically solving PDEs. RBF methods are high-order, mesh-free, and dimension-independent. With this success, several more RBF col-

location methods [2, 3] were developed in the 90's, however, these *global* methods do not scale well to large problems, as they become computationally expensive and ill-conditioned. Shortly after the turn of the millennium, the radial basis function generated finite difference method (RBF-FD) was independently introduced by a number of groups [4, 5, 6, 7] to address the issues with global RBF methods. RBF-FD is a *local* method that scales well to large problems (see Sections 2.3) while still giving high-order accurate solutions.

Over the last decade, researchers have shown increasing interest in applying RBF techniques to solving PDEs on *surfaces*. Applications of surface PDEs are numerous in biology, chemistry, materials science, and computer graphics, and include reaction-diffusion [8, 9, 10], chemical oscillator [11], and transport [12, 13, 14] problems. While several RBF-FD methods for solving PDEs on surfaces have been developed, there have yet to be any studies that directly compare the local techniques on the same test problems. Here we do exactly this. Specifically we compare the Projected Gradient method, the Hermite RBF-FD method, and a new Tangent Plane method to approximate the Laplace-Beltrami operator — a surface differential operator present in many surface PDEs.

In the early 2010's, Flyer and Wright introduced the *Projected Gradient method* to solve the shallow water equations on the sphere [12, 13]. Fuselier and Wright generalized it to arbitrary surfaces using global RBFs [11] and this was later extended by Shankar et al. [15, 16] to local RBF-FD. We discuss the Projected Gradient method at length in Section 3.2.2.

In 2012, Piret developed two global RBF techniques called the low-order and high-order *Orthogonal Gradients methods* (OGr) [8], which were RBF adaptations of the Closest Point method [17]. In 2016, Piret and Dunnn used the Hermite interpolation

techniques of [18, 19] to create a local method, which they named Fast OGr [20]. Each of the three OGr methods are a combination of an algorithm to approximate normal vectors, and an algorithm to generate finite difference weights. We refer to the portion of Fast OGR that generates finite difference weights as Hermite RBF-FD and discuss it in Section 3.2.3.

In 2006, Demanet [21] introduced a technique for using 2-dimensional Euclidean finite difference stencils to approximate surface differential operators. This is done by projecting points on the surface into the tangent plane and finding finite difference weights for the co-planar points. We combine Demanet's technique with 2-dimensional planar RBF-FD to create a new *Tangent Plane* method, which we describe in detail in Section 3.2.1.

The remainder of this thesis is outlined as follows. Chapter 2 contains background information on RBF interpolation and finite differences. Additionally, it reviews some vector calculus identities regarding the Laplace-Beltrami operator. Chapter 3 details the three high-order methods that are used to generate finite difference weights for the Laplace-Beltrami operator. Chapter 4 discusses parameter selection, and showcases numerical results comparing the three RBF-FD methods. Lastly, chapter 5 summarizes these results and concludes with some ideas to be explored in the future.

CHAPTER 2

BACKGROUND

2.1 Finite Difference Weights from Lagrange Interpolation

Polynomial interpolation is foundational to much of the theory behind this work and finite differences in general. Generalizing polynomial interpolation to multi-dimensional data is not straightforward; in particular, minimal-degree polynomial interpolation is usually not unique. One of the principle advantages of RBF interpolation is that it easily generalizes to any dimension. Recently, there has been growing interest in incorporating polynomial basis terms into RBF techniques and thus, we begin with a review of relevant properties of polynomial interpolation as they pertain to finite difference weights.

Given a set of distinct points $\{x_n\}_{n=0}^N \subset \mathbb{R}$ and associated function values $\{f_n\}_{n=0}^N \subset \mathbb{R}$, the Lagrange form of the interpolating polynomial of degree N is given by

$$p_N(x) = \sum_{n=0}^{N} \ell_n(x) f_n \tag{2.1}$$

where

$$\ell_n(x) = \prod_{\substack{k=0 \\ k \neq n}}^{N} \frac{(x - x_k)}{(x_n - x_k)}, \qquad \text{for} \qquad n = 0, 1, \ldots, N \tag{2.2}$$

are the Lagrange polynomials associated with the points $\{x_n\}_{n=0}^N$. This form of polynomial interpolation can generate finite difference weights due to the following

theorem.

Theorem 2.1. Let $f \in C^{N+1}([-R,R])$, $\{x_n\}_{n=0}^N \subset (-R,R)$ and $m \leq N$. Suppose that we are given weights $\{w_n\}_{n=0}^N \subset \mathbb{R}$ such that for any polynomial p up to degree $d \leq N$

$$\frac{d^m}{dx^m} p(x)\bigg|_{x=0} = \sum_{n=0}^{N} w_n p(x_n).$$

Then for $h > 0$,

$$\frac{d^m}{dx^m} f(x)\bigg|_{x=0} = \frac{1}{h^m} \sum_{n=0}^{N} w_n f(hx_n) + \mathcal{O}(h^{N-m+1}).$$

Proof: Let $T_N(x)$ be the $(N+1)^{\text{st}}$ partial sum of the Taylor series of f about $x = 0$ so that

$$f(x) = T_N(x) + \mathcal{O}(x^{N+1}).$$

Then

$$\begin{aligned}
\sum_{n=0}^{N} w_n f(hx_n) &= \sum_{n=0}^{N} w_n T_N(hx_n) + \mathcal{O}(h^{N+1}) \\
&= \frac{d^m}{du^m} T_N(u)\bigg|_{u=0} + \mathcal{O}(h^{N+1}), \qquad \text{for } u = hx \\
&= h^m \frac{d^m}{dx^m} T_N(x)\bigg|_{x=0} + \mathcal{O}(h^{N+1}) \\
&= h^m f^{(m)}(0) + \mathcal{O}(h^{N+1})
\end{aligned}$$

and thus

$$\frac{d^m}{dx^m} f(x)\bigg|_{x=0} = \frac{1}{h^m} \sum_{n=0}^{N} w_n f(hx_n) + \mathcal{O}(h^{N-m+1})$$

as desired. □

Differentiating the Lagrange form of the interpolating polynomial (2.1) we obtain

$$\left.\frac{d^m}{dx^m} p_N(x)\right|_{x=0} = \sum_{n=0}^{N} \underbrace{\ell_n^{(m)}(0)}_{w_n} f(x_n)$$

where the weights $w_n = \ell_n^{(m)}(0)$ are only dependent on the locations of $\{x_n\}_{n=0}^N$. If f is any polynomial of degree $\leq N$ then $f = p_N$ and the weights are exact[1] – thus satisfying the premise of the theorem.

When applied to equally spaced points, this method yields the same weights as the usual derivation using the method of undetermined coefficients and the Taylor series expansion [23, p. 7]. Note that Theorem 2.1 guarantees a minimum order of accuracy, but the actual order may be higher. For example, the finite difference weights that approximate $\frac{d^2}{dx^2} f|_{x=0}$ for the points $-h, 0, h \in \mathbb{R}$ given by $h^{-2}, -2h^{-2}, h^{-2}$, are a second-order-accurate approximation while the theorem only guarantees first-order accuracy.

While Lagrange interpolation is effective for finding finite difference weights for arbitrary node layouts in one dimension, it does not generalize to arbitrary node layouts in higher dimensions. The theorem, however, does generalize to higher dimensions though it does not prescribe a method for finding such weights.

2.2 Radial Basis Functions

2.2.1 Mairhuber-Curtis Theorem

Polynomial interpolation in one dimension is quite powerful and leads to high order approximations, difference rules, and quadrature rules. It is very natural to try

[1] It is worth noting that a naive computation of finite difference weights using derivatives of Lagrange polynomials is unstable, however a stable algorithm is presented by Fornberg in [22].

extending its utility to multidimensional data. Indeed, multidimensional polynomial interpolation is possible, but there are a few difficulties that arise.

For one, polynomial interpolation suffers from the *curse of dimensionality*. As the dimension increases, the number of basis terms increases rapidly to span polynomials of the same degree. For example, second degree polynomials in two dimensions require six basis terms:

$$1, \quad x, \quad y, \quad x^2, \quad xy, \quad y^2,$$

while second degree polynomials in three dimensions require ten basis terms:

$$1, \quad x, \quad y, \quad z, \quad x^2, \quad xy, \quad xz, \quad y^2, \quad yz, \quad z^2.$$

In general the number of terms in the basis is given by $\binom{\text{deg}+\text{dim}}{\text{deg}}$. For a fixed dimension, this quantity grows like a polynomial of degree equal to the dimension.

There is also a question of linear independence. In one dimension, n polynomial basis terms are linearly independent over n points if the points are distinct. This is not the case in higher dimensions. In particular any distinct set of points that lie on an algebraic surface will give rise to linearly *dependent* basis terms if the basis is not chosen carefully. For example, the terms $1, x^2, y^2, z^2$ are linearly dependent on the sphere.

Of course if one knows that points come from a sphere then one can simply choose the spherical harmonics as a polynomial basis and linear independence (of the basis at least) is guaranteed. This begs the question: is there a choice of polynomial basis which will be linearly independent for any collection of points? The famous Mairhuber-Curtis theorem[2] [24, 25] settles this question, and the answer is *no*.

[2]Occasionally the theorem is called the Haar-Mairhuber-Curtis theorem.

Here we present a simplified version of the Mairhuber-Curtis theorem that is sufficient for our purposes.

Theorem 2.2. Let $\phi_1, \phi_2, \dots \phi_N$ be linearly independent functions from $\mathbb{R}^d$ to $\mathbb{R}$ with $d > 1$. Then there exists a distinct set of points $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N \in \mathbb{R}^d$ such that the matrix

$$\begin{bmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_N(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_N(\mathbf{x}_N) \end{bmatrix} \tag{2.3}$$

is singular.[3]

Proof: Choose any set of points $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N \in \mathbb{R}^d$ and suppose that (2.3) is not singular. In $\mathbb{R}^d$ with $d \geq 2$, a finite set of points cannot separate the space. Thus, there is some simple closed path that passes through the points $\mathbf{x}_1$ and $\mathbf{x}_2$ but none of the others. Let $\mathbf{x}(t)$ be a parameterization of this curve such that $\mathbf{x}(0) = \mathbf{x}_1, \mathbf{x}(1) = \mathbf{x}_2$, and $\mathbf{x}(2) = \mathbf{x}_1$ closing the curve. Consider the function

$$f(t) = \det \left(\begin{bmatrix} \phi_1(\mathbf{x}(t)) & \phi_2(\mathbf{x}(t)) & \dots & \phi_N(\mathbf{x}(t)) \\ \phi_1(\mathbf{x}(t+1)) & \phi_2(\mathbf{x}(t+1)) & \dots & \phi_N(\mathbf{x}(t+1)) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_n(\mathbf{x}_N) \end{bmatrix} \right),$$

and note that it is continuous in t, since the determinant is a continuous function of the matrix entries. Then we have that

[3]The matrix in (2.3) is called the interpolation matrix, because, if it is non-singular, then there is a unique linear combination of $\phi_1, \phi_2, \dots \phi_N$ that interpolates a given function at the points $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N$.

$$f(0) = \det\left(\begin{bmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_N(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_N(\mathbf{x}_N) \end{bmatrix}\right)$$

and

$$f(1) = \det\left(\begin{bmatrix} \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_N(\mathbf{x}_2) \\ \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_N(\mathbf{x}_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_N(\mathbf{x}_N) \end{bmatrix}\right) = -f(0)$$

since we have swapped the first and second rows. By the intermediate value theorem we know that there is some $t_0 \in [0,1]$ such that $f(t_0) = 0$ and thus the points $\mathbf{x}(t_0), \mathbf{x}(t_0+1), \mathbf{x}_3, \mathbf{x}_4, \dots \mathbf{x}_N$ satisfy our theorem. □

The importance of the Mairhuber-Curtis theorem (in our context at least) is that one cannot choose a set of basis functions independent of the point set and be guaranteed a uniquely solvable multidimensional interpolation problem. Naturally, one wonders: is there a good way to choose basis functions that *are* dependent on the point set? The answer is a resounding *yes*!

In 1971 Hardy [26] developed such a technique in the context of Cartography. Hardy was working with scattered elevation data and attempting to recreate surfaces. He was creating interpolants that he described as "*the summation of a single class of quadric surfaces*" which he called multiquadric surfaces, and they were of the form

$$\sum c_i \left[(x_i - x)^2 + (y_i - y)^2 + C\right]^{1/2}$$

where $C \geq 0$. This was the first instance of what we now call *radial basis function* (RBF) *interpolation*. Hardy did not prove any theoretical results, but since then,

there has been considerable mathematical work done in validating and generalizing Hardy's method.

2.2.2 Radial Basis Function Interpolation

When RBFs are discussed in the literature, their definition is given somewhat circumlocuitously or by example. They share a common form and are dubbed RBFs if they have desirable interpolation properties. The necessary conditions in general are not known [27, p. 27], however there has been extensive theory developed that describes sufficient conditions. We refer the reader to [27, 28] for a thorough discussion of these conditions and proceed with a description of the form of radial basis functions and provide several examples.

Definition 2.1. A function $\Phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a **radial kernel** with respect to the norm $\|\cdot\|$, if there exists a function $\phi : [0, \infty) \to \mathbb{R}$ such that $\Phi(\mathbf{x}, \mathbf{y}) = \phi(\|\mathbf{x} - \mathbf{y}\|)$.

Some functions ϕ are called RBFs, and Table 2.1 provides some select examples. In principle, one can choose any norm, but we restrict ourselves to the standard Euclidean norm for the entirety of this thesis.

Given a distinct set of points $X = \{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^d$ and a radial basis function ϕ, the corresponding radial kernel Φ generates functions $\phi_n : \mathbb{R}^d \to \mathbb{R}$ given by $\phi_n(\mathbf{x}) = \Phi(\mathbf{x}, \mathbf{x}_n) = \phi(\|\mathbf{x} - \mathbf{x}_n\|)$ for $n = 1, 2, \ldots, N$. These ϕ_n are radially symmetric about the point $\mathbf{x}_n$ and are linearly independent over $\mathbb{R}^d$, thus they form a basis for a space, and are also called radial basis functions.[4] Radial basis function interpolation is the process of finding interpolants that exist in this space.

[4] Some authors choose to call ϕ *basic functions* [27, p. 18] to distinguish them from ϕ_n. Others rely on context.

Given function values $\{f_n\}_{n=1}^N \subset \mathbb{R}$ that correspond to the points in X, we seek an interpolant of the form

$$s(\mathbf{x}) = \sum_{n=1}^{N} c_n \phi(\|\mathbf{x} - \mathbf{x}_n\|)$$

for some RBF ϕ, and some set of weights $\{c_n\}_{n=1}^N \subset \mathbb{R}$. For a given ϕ, finding s is equivalent to finding the weights $\{c_n\}_{n=1}^N$ that satisfy the linear system

$$\begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) & \dots & \phi(\|\mathbf{x}_1 - \mathbf{x}_N\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) & \dots & \phi(\|\mathbf{x}_2 - \mathbf{x}_N\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_N - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_N - \mathbf{x}_2\|) & \dots & \phi(\|\mathbf{x}_N - \mathbf{x}_N\|) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}. \tag{2.4}$$

The matrix in Equation (2.4) is called the RBF interpolation matrix. For many choices of RBF, and in particular for all of the infinitely smooth RBFs listed in table 2.1, the interpolation matrix is guaranteed to be non-singular, and the resulting interpolant s is called an RBF interpolant.

RBF interpolation can be used to approximate functions in a manner similar to

Table 2.1: Some select radial basis functions. For the infinitely smooth RBFs, ε is a parameter that controls the shape of the functions. Though the odd polyharmonic splines appear smooth as functions of r, the corresponding radial kernels only have finite smoothness at their centers. For example, using $\phi(r) = r^3$ in one dimension, the kernel centered at 0 is given by $|x - 0|^3$, which has a discontinuous third derivative.

Infinitely Smooth		**Polyharmonic Splines**	
Multiquadric	$\phi(r) = \sqrt{1 + (\varepsilon r)^2}$	Thin-plate spline	$\phi(r) = r^2 \log r$
Gaussian	$\phi(r) = e^{-(\varepsilon r)^2}$	Natural Cubic Spline	$\phi(r) = r^3$
Inverse Multiquadric	$\phi(r) = \sqrt{1 + (\varepsilon r)^2}^{-1}$	Even PHS	$\phi(r) = r^{2\ell} \log r$
Inverse Quadratic	$\phi(r) = \left(1 + (\varepsilon r)^2\right)^{-1}$	Odd PHS	$\phi(r) = r^{2\ell+1}$

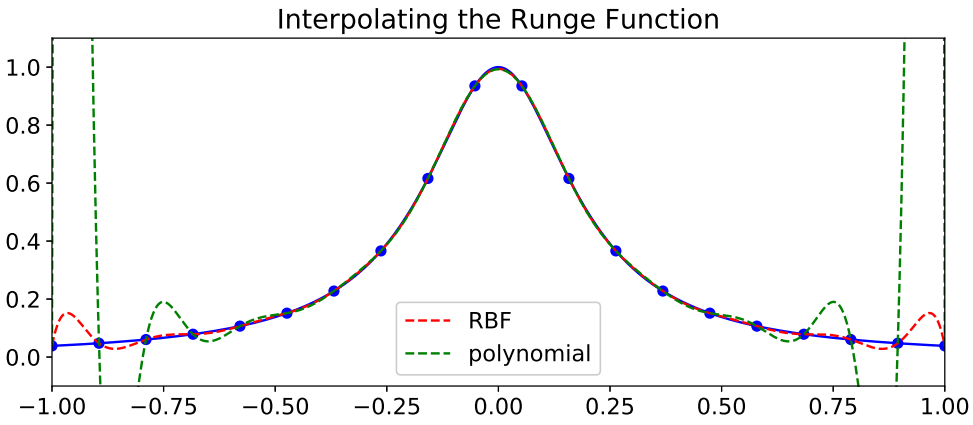


Figure 2.1: RBF interpolation (Gaussian $\varepsilon = 2.369$) of the Runge Function over $[-1, 1]$ at $N = 20$ equally-spaced points compared with polynomial interpolation.

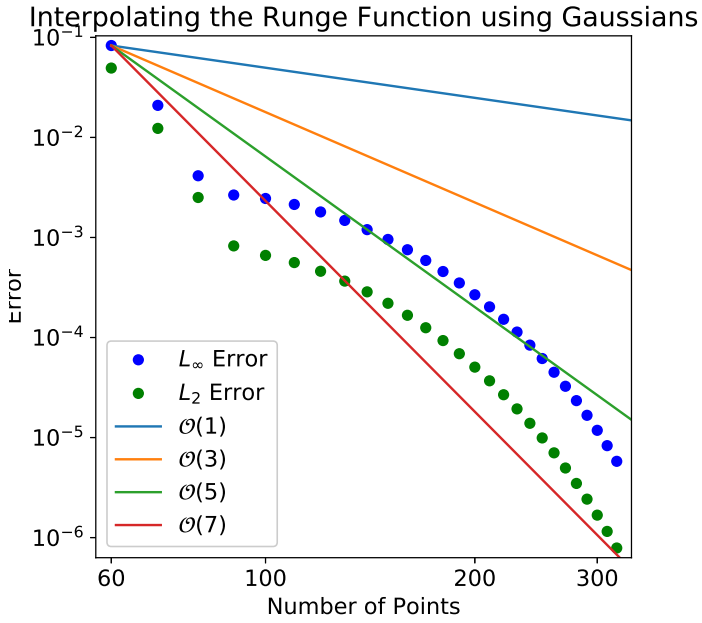


Figure 2.2: Error on a log-log scale when interpolating the Runge function with Gaussians ($\varepsilon = 47.06$). Note the apparent spectral convergence.

polynomial interpolation. One result that is particularly impressive is that some RBFs exhibit spectral convergence.[5] Figure 2.1 shows the Runge Function $f(x) = 1/(1+25x^2)$ interpolated over the interval $[-1,1]$ at $N = 20$ equally-spaced points using using Gaussian RBFs as compared to polynomial interpolation. Figure 2.2 shows that as N increases, the error in the RBF interpolant decreases geometrically.[6]

Of course one of the primary motivations of RBFs was to interpolate higher-dimensional data. A standard test case in two dimensions is *Franke's function* [27, p. 20] [30] given by

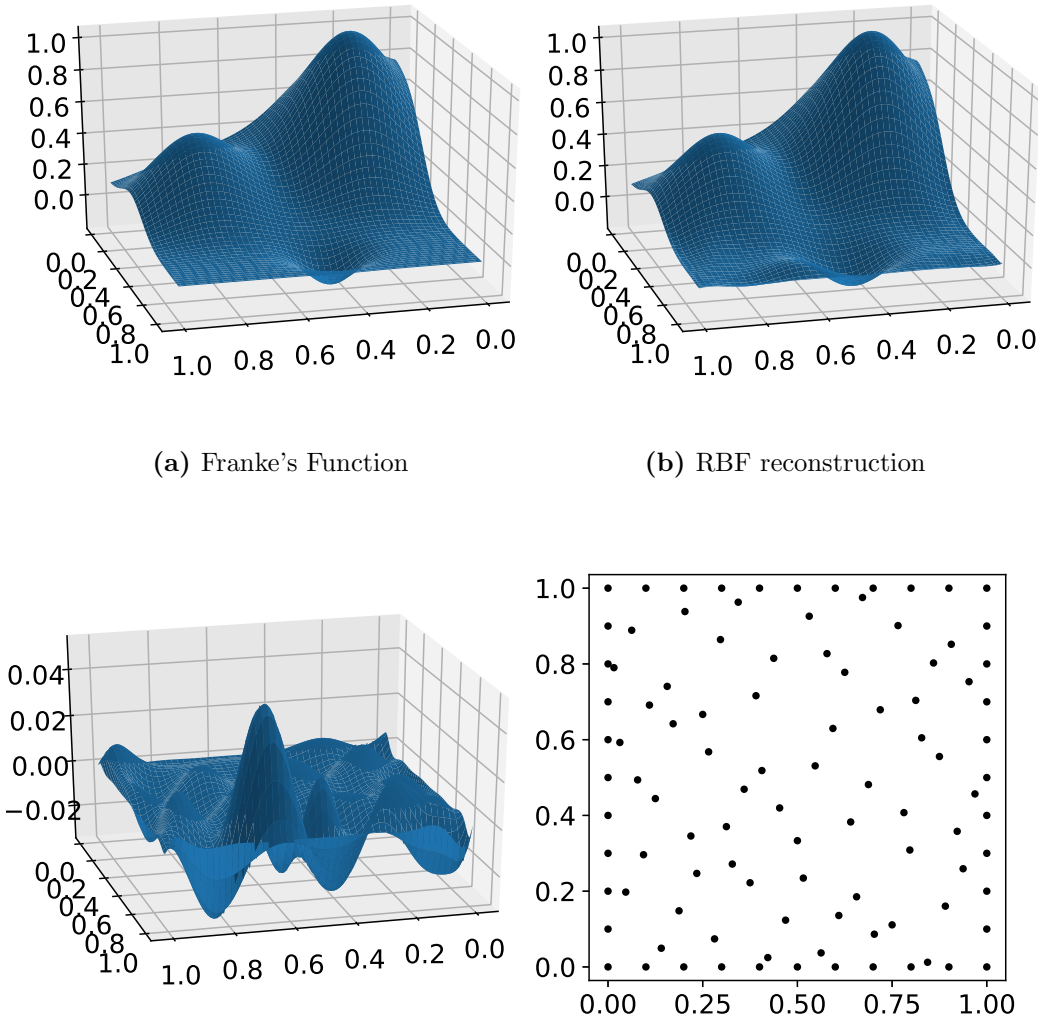
$$\begin{aligned} f(x,y) = \quad & \frac{3}{4}\exp\left(-\tfrac{1}{4}\left((9x-2)^2+(9y-2)^2\right)\right) \\ & + \frac{3}{4}\exp\left(-\tfrac{1}{49}(9x+12)^2 - \tfrac{1}{10}(9y+1)^2\right) \\ & + \frac{1}{2}\exp\left(-\tfrac{1}{4}\left((9x-7)^2+(9y-3)^2\right)\right) \\ & - \frac{1}{5}\exp\left(-(9x-4)^2-(9y-7)^2\right). \end{aligned}$$

Figure 2.3 shows a plot of Franke's function over $[0,1]^2$ along with an RBF reconstruction using only $N = 100$ points. Figure 2.4 shows that convergence of the RBF interpolant is again faster than seventh order.

In the preceding examples, the parameter ε has been chosen without explanation. In fact, selecting a good value for this parameter is critical for the success of RBF interpolation when using infinitely smooth RBFs. As seen in Figure 2.5, the parameter ε controls the "shape" of the RBF. As ε grows larger, the function becomes sharper or more peaked, and as ε grows smaller, the function becomes flatter. In particular, as $\varepsilon \to \infty$ the interpolation matrix converges to the identity and the condition number tends to 1 leading to a very stable interpolation problem. However, as the condition

[5] Specifically, RBFs that are infinitely smooth exhibit spectral convergence when interpolating certain classes of functions. See [27, 28, 29] for a thorough discussion.

[6] The error cannot decrease indefinitely. Eventually it will begin to increase due to ill-conditioning.



(a) Franke's Function

(b) RBF reconstruction

(c) Error

(d) Halton Points

Figure 2.3: (a) Franke's Function, (b) an RBF reconstruction using $N = 100$ Gaussians ($\varepsilon = 2.2$), (c) the error, (d) the sample points which are composed of Halton points [31] and equally spaced points along the boundary.

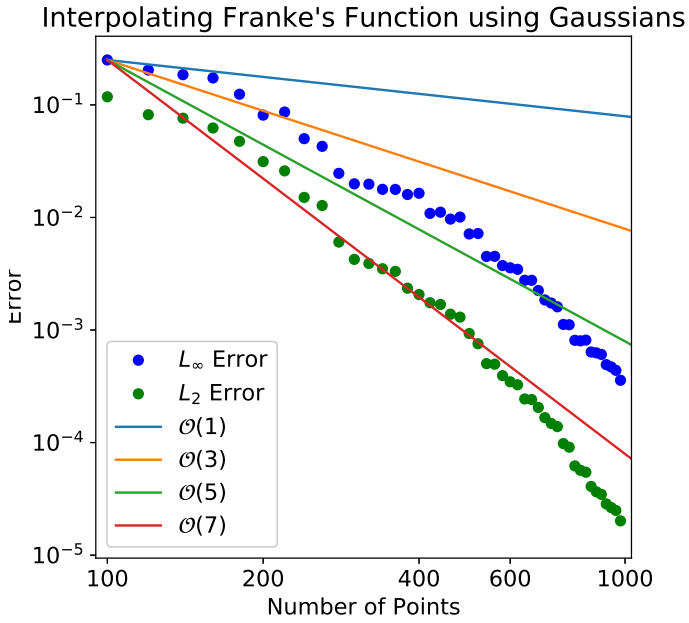


Figure 2.4: Error on a log-log scale when interpolating Franke's function over $[0, 1]^2$ with Gaussians ($\varepsilon = 10.6$). Note the apparent spectral convergence.

number becomes larger, the approximation accuracy of the RBF interpolant becomes poor. Figure 2.6 shows the Runge Function again interpolated with Gaussians, but this time with $\varepsilon = 100$ — much too high. The interpolant is essentially zero everywhere except near the interpolation points where it quickly spikes up to interpolate the function. Such an interpolant has been dubbed the "bed-of-nails interpolant" as the function visually appears to be resting on the sharp tips of the Gaussian.

The interpolation matrix is guaranteed to be non-singular for all $\varepsilon > 0$, however, as $\varepsilon \to 0$ the condition number of the interpolation matrix increases without bound. Figure 2.7 shows plots of the condition number and the maximum error as a function of ε for the problem of interpolating the Runge Function at $n = 100$ equally spaced points. As expected, for ε large the condition number is small and the error is

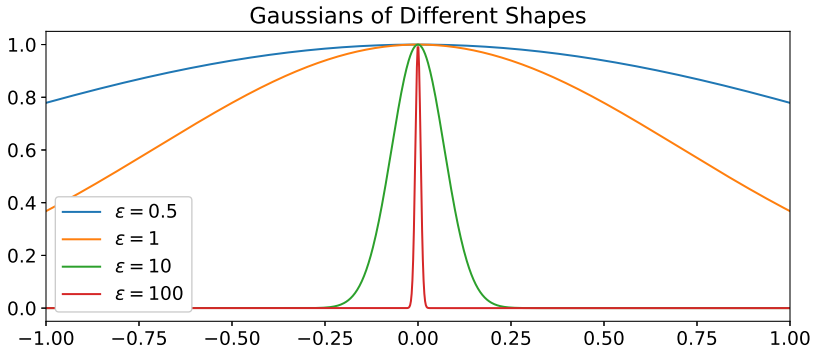


Figure 2.5: The Gaussian for different choices of ε.

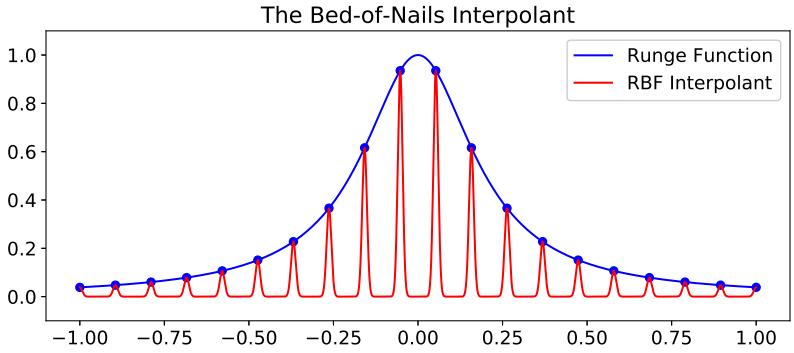
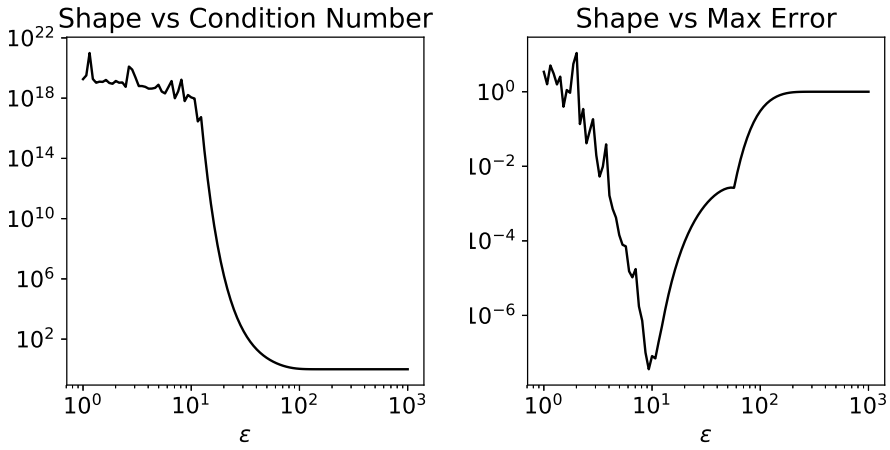


Figure 2.6: Interpolating the Runge Function at $n = 20$ equally spaced points using Gaussian RBFs with $\varepsilon = 100$ - the bed-of-nails interpolant.

large. For ε small, the condition number is large and the error is also large due to ill-conditioning. Finding effective algorithms and heuristics for choosing ε is an active area of study [32, 33]. There also exist stable algorithms that can compute the RBF interpolant for all values of $\varepsilon > 0$ and for the limiting case of $\varepsilon \to 0$. However, these are domain-specific, RBF-specific, or both [34, 35, 36, 37].

For the convergence plots in Figures 2.2 and 2.4, ε was optimized for interpolation over the maximum number of points listed in each plot. Figure 2.8 shows the same plots but includes larger values of N. Since ε is fixed, the interpolation matrix eventually becomes ill-conditioned. Decreasing ε will allow for larger interpolation



(a) Condition Number **(b)** Maximum Error

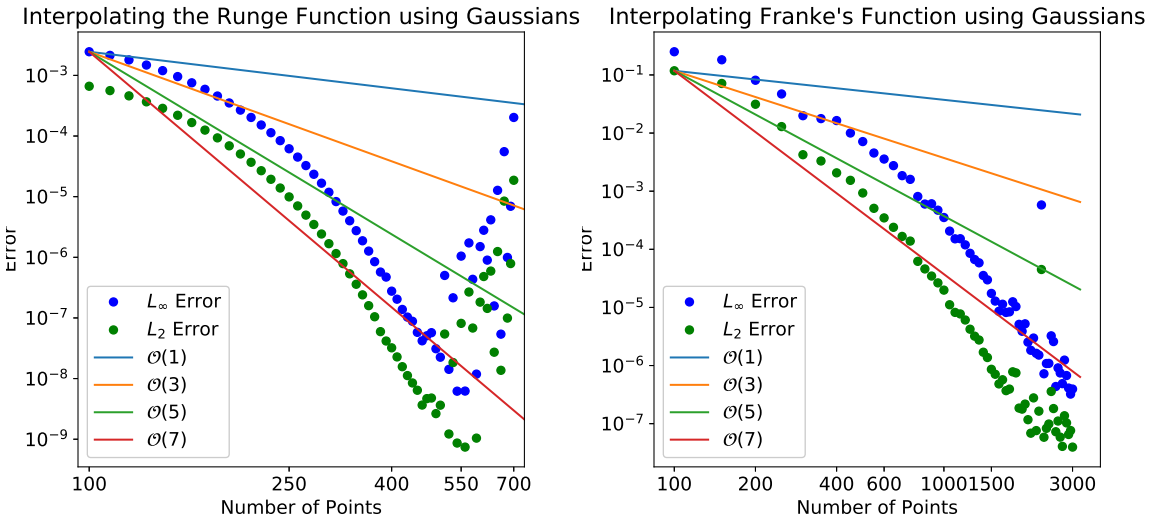
Figure 2.7: (a) The condition number of the interpolation matrix and (b) the maximum error of the interpolant when interpolating the Runge Function at $n = 100$ equally spaced points for varying values of ε. Though the condition number appears to plateau around 10^{18} this is simply numerical instability of the calculation. In reality, the condition number continues to grow rapidly.

matrices, however it will also reduce the accuracy of the interpolants, leading to little or no change in the error of the best approximation. Furthermore, it has been shown that, in the limit as $\varepsilon \to 0$, interpolation using Gaussian RBFs converges to polynomial interpolation and therefore suffers from Runge Phenomenon [38]. As a consequence, even stable algorithms may have a limit to their accuracy.

2.2.3 Local RBF Interpolation

A common strategy to keep computational costs down is to create an interpolant by patching together local interpolants to the function. This is known as *local* RBF interpolation in contrast to *global* RBF interpolation as described previously.

With local interpolation, a parameter $k \in \mathbb{N}$ is chosen to be smaller than N the total number of points. Then to evaluate the interpolant at a point $\mathbf{x}$ in the domain,



(a) Runge Function saturation

(b) Frankes Function saturation

Figure 2.8: Stagnation in the error as N increases for (a) Runge's Function, and (b) Franke's Function.

we first find the closest interpolation point

$$\mathbf{x}_{\text{cp}} = \underset{\mathbf{x}_n}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{x}_n\| .$$

We then reorder the points by distance from $\mathbf{x}_{\text{cp}}$ so that $\mathbf{x}_1 = \mathbf{x}_{\text{cp}}$ and

$$\|\mathbf{x}_2 - \mathbf{x}_1\| \leq \|\mathbf{x}_3 - \mathbf{x}_1\| \leq \cdots \leq \|\mathbf{x}_k - \mathbf{x}_1\| .$$

Under this local ordering, we form the set $\{\mathbf{x}_n\}_{n=1}^{k}$ which is called the *stencil*. We call $\mathbf{x}_1$ the *stencil center*, and k the *stencil size*. The stencil is said to contain the k nearest neighbors to $\mathbf{x}_1$ (itself included). We then perform RBF interpolation using only points in the stencil as interpolation points. Note that the resulting interpolant may be discontinuous at points that are equidistant from the two nearest interpolation points. Also note that the ordering of points is not necessary to the process, but is used here merely as a convenient way to express the stencil and stencil center.

To see that this is computationally more efficient, suppose that we wish to interpolate to M points. Finding the *global* RBF interpolant weights requires $\mathcal{O}(N^3)$ operations using Gaussian elimination. Then evaluating the interpolant at M points requires $\mathcal{O}(MN)$ operations for a total computational complexity of $\mathcal{O}(MN + N^3)$. Local interpolation requires that for each of the N interpolation points we form the stencil of the k nearest neighbors. This can be done using $\mathcal{O}(N \log N)$ operations with a KD-tree. Then we find the weights for each stencil by inverting N matrices of size $k \times k$ requiring $\mathcal{O}(Nk^3)$ operations. For each of the M points at which we wish to evaluate the interpolant, we find the closest interpolation point which is $\mathcal{O}(M \log N)$ when using the KD-tree. Lastly, evaluating the local interpolants is $\mathcal{O}(Mk)$ for a total of $\mathcal{O}(M \log N + Nk^3)$. For fixed M and k, local interpolation is more efficient and can be easily parallelized.

The drawback to local interpolation is that the convergence rate is tied to the stencil size and is no longer spectral for fixed k. Figure 2.9 shows the convergence rates of local RBF interpolation to the Runge Function. The convergence rates are approximately polynomial.

2.2.4 Polyharmonic Splines and Polynomials

Our focus is on using RBFs to approximate differential operators which are inherently local (opposed to integral operators for example). This coupled with the improved efficiency of local RBF interpolation makes it ideal for our purposes — sacrificing only spectral convergence rates. Since spectral convergence rates were the primary reason for choosing the infinitely smooth RBFs, it is natural to consider alternatives. In particular we turn our attention to *polyharmonic splines*.

The polyharmonic spline (PHS) RBFs are given by

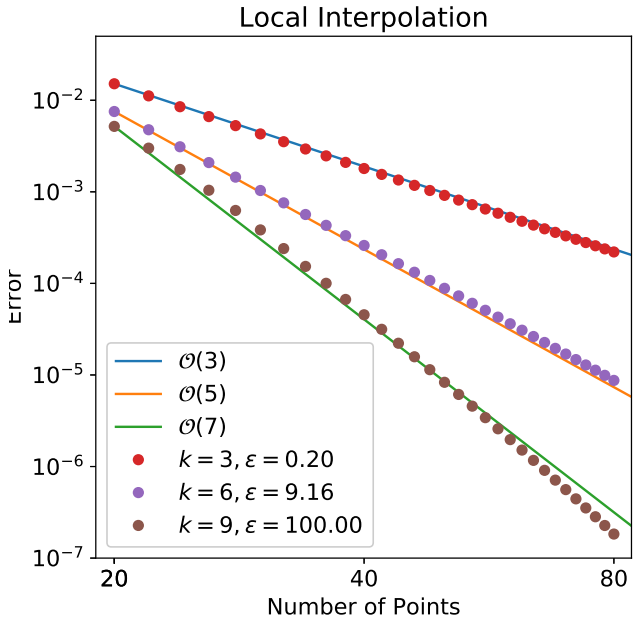


Figure 2.9: Convergence of local RBF interpolation to the Runge Function using Gaussians. Note that the order of convergence is polynomial (not spectral) and is dependent on the stencil size k.

$$\phi(r) = r^{2\ell+1} \qquad \text{or} \qquad \phi(r) = r^{2\ell} \log r$$

for $\ell \in \mathbb{N}$ where the exponent is referred to as the degree of the PHS. Unlike infinitely smooth RBFs, PHSs only attain polynomial convergence rates, even for global interpolation as seen in Figure 2.10, however, they do not require tuning a shape parameter. Furthermore, they do not guarantee non-singular interpolation matrices, but are rather *conditionally positive definite*.

Definition 2.2. [27, p. 63] A real-valued continuous even function f is called **conditionally positive definite**[7] of order m on $\mathbb{R}^d$ if

[7] Historically this was the definition given for *strictly conditionally positive definite* functions. Some texts continue to use this as the definition of strictly conditionally positive definite functions, and use a slightly different definition for conditionally positive definite functions.

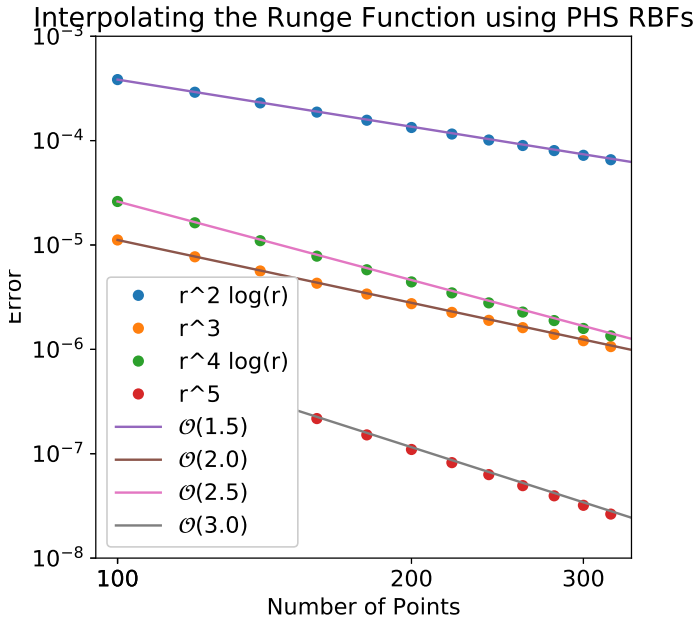


Figure 2.10: Convergence of global RBF interpolation to the Runge Function using PHSs (without appended polynomial terms).

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k f(\mathbf{x}_j - \mathbf{x}_k) \geq 0 \tag{2.5}$$

for any distinct points $\{\mathbf{x}_n\}_{n=1}^{N} \subset \mathbb{R}^d$ and non-zero $\mathbf{c} \subset \mathbb{R}^N$ satisfying

$$\sum_{j=1}^{N} c_j p(\mathbf{x}_j) = 0$$

for all polynomials p of degree less than or equal to $m-1$.

The PHSs $\phi(r) = r^{2\ell+1}$ and $\phi(r) = r^{2\ell}\log(r)$ are each CPD of order $\ell+1$. The key property of CPD RBFs of order $\ell+1$ is that their interpolation matrices will be non-singular if a basis of polynomials up to degree ℓ are added to the interpolant (provided that the polynomials are not linearly dependent on the set of interpolation points) and appropriate extra conditions are satisfied. Specifically, the new interpolant takes

the form

$$s(\mathbf{x}) = \sum_{n=1}^{N} c_n \phi(\|\mathbf{x} - \mathbf{x}_n\|) + \sum_{m=1}^{L} \lambda_m p_m(\mathbf{x})$$

and is subject to the *moment conditions*

$$\sum_{n=1}^{N} c_n p_m(\mathbf{x}_n), \quad \text{for} \quad m = 1, 2, \ldots, L$$

where L is the dimension of the polynomial space up to degree ℓ, and $\{p_m\}_{m=1}^{L}$ is a basis for this space. The interpolation and moment conditions can be expressed as the linear system

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}, \tag{2.6}$$

where A is the RBF matrix from (2.4), $P_{ij} = p_j(\mathbf{x}_i)$ and $f_i = f(\mathbf{x}_i)$. Here we say that the PHSs are augmented with a polynomial basis of degree ℓ, or equivalently, degree ℓ polynomials are appended [27, ch. 6-8].

A particularly satisfying result, is that interpolation in one dimension using $\phi(r) = r^3$ and augmenting with up to linear terms will exactly reproduce cubic splines with natural boundary conditions [39] [28, p. 10].

Reintroducing polynomials also reintroduces the difficulties of multivariate polynomial interpolation. In this setting, the number of basis terms is not equal to the number of interpolation points, so we are always free to choose a full basis of polynomials up to a given degree. Furthermore, we are not obligated to add polynomials as we add points, thus avoiding Runge phenomenon that can accompany high-degree polynomial interpolation. The more difficult barrier is the non-uniqueness of polynomial interpolants in higher dimensions. This manifests as the matrix P in (2.6) being column-rank deficient.

Experimentally this is not an issue if sufficiently many points are chosen pseudo-

randomly. If however, the points lie on an algebraic surface then polynomial terms will necessarily be linearly dependent. Even if the manifold is not an algebraic surface, it is likely that it can be approximated (at least locally) by an algebraic surface leading to ill-conditioning of (2.6). As our main focus here is on solving PDEs on manifolds, this must be addressed, and we do so in Section 3.1.

2.3 Radial Basis Function Finite Differences

Any interpolation scheme can be used to generate finite difference weights for differential operators. RBF interpolation is no exception, and the resulting method is called radial basis function finite differences (RBF-FD).

Suppose we are given a set of points $\{\mathbf{x}_n\}_{n=1}^N \subset \Omega$ and a differential operator $\mathcal{L}$. We would like a set of weights $\boldsymbol{\omega} \in \mathbb{R}^N$ such that for any function $f : \Omega \to \mathbb{R}$ we have that $\mathcal{L}f\big|_{\mathbf{x}=\mathbf{x}_1} \approx \sum_n \omega_n f(\mathbf{x}_n)$. For ease of notation, we will use $\mathcal{L}f$ to denote $\mathcal{L}f\big|_{\mathbf{x}=\mathbf{x}_1}$.

Suppose we are given a function $f : \Omega \to \mathbb{R}$. Then there exists an RBF interpolant of the form

$$s(\mathbf{x}) = \sum_{n=1}^{N} c_n \phi(\|\mathbf{x} - \mathbf{x}_n\|) + \sum_{m=1}^{L} \lambda_m p_m(\mathbf{x})$$

where the interpolation weights $\mathbf{c}, \boldsymbol{\lambda}$ satisfy (2.6):

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}.$$

Applying the linear operator to the interpolant, we have that

$$
\begin{aligned}
\mathcal{L}f &\approx \mathcal{L}s \\
&= \sum_n c_n \mathcal{L}\phi(\|\mathbf{x}-\mathbf{x}_n\|) + \sum_m \lambda_m \mathcal{L}p_m(\mathbf{x}) \\
&= \begin{bmatrix} \mathcal{L}\boldsymbol{\phi}^T & \mathcal{L}\boldsymbol{\pi}^T \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \boldsymbol{\lambda} \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \mathcal{L}\boldsymbol{\phi}^T & \mathcal{L}\boldsymbol{\pi}^T \end{bmatrix} \begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix}^{-1}}_{\begin{bmatrix} \boldsymbol{\omega}^T & \boldsymbol{\gamma}^T \end{bmatrix}} \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix} \\
&= \boldsymbol{\omega}^T \mathbf{f}.
\end{aligned} \tag{2.7}
$$

Thus our finite difference weights $\boldsymbol{\omega}$ are found by solving the system

$$
\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} \mathcal{L}\boldsymbol{\phi} \\ \mathcal{L}\boldsymbol{\pi} \end{bmatrix}
$$

where the values of $\mathcal{L}\boldsymbol{\phi}_i = \mathcal{L}\phi(\|\mathbf{x}-\mathbf{x}_i\|)$ and $\mathcal{L}\boldsymbol{\pi}_i = \mathcal{L}p_i$ are calculated analytically. Again, this direct approach requires that P is full column-rank.

We now have a theorem to complement Theorem 2.1.

Theorem 2.3. Suppose $\boldsymbol{\omega} \in \mathbb{R}^N$ are finite difference weights for the points $\{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^d$ that approximate the differential operator $\mathcal{L}(\cdot)\big|_{\mathbf{x}=\mathbf{x}_1}$ generated using RBF-FD augmented with a basis of polynomials up to degree ℓ. These weights are exact when applied to polynomials up to degree ℓ.

Proof: Let f be a d-variate polynomial of degree at most ℓ and let $p_1, p_2, \ldots p_L$ be a basis for d-variate polynomials up to degree ℓ. Then there exist $\boldsymbol{\lambda} \in \mathbb{R}^L$ such that

$$
f(\mathbf{x}) = \sum_{m=1}^{L} \lambda_m p_m(\mathbf{x})
$$

and the RBF interpolant is defined by the weights $\mathbf{c} = \mathbf{0}$ and $\boldsymbol{\lambda}$. From (2.7) we have

$$
\begin{aligned}
\boldsymbol{\omega}^T \mathbf{f} &= \begin{bmatrix} \mathcal{L}\boldsymbol{\phi}^T & \mathcal{L}\boldsymbol{\pi}^T \end{bmatrix} \begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{L}\boldsymbol{\phi}^T & \mathcal{L}\boldsymbol{\pi}^T \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \boldsymbol{\lambda} \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{L}\boldsymbol{\phi}^T & \mathcal{L}\boldsymbol{\pi}^T \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\lambda} \end{bmatrix} \\
&= \sum_{m=1}^{L} \lambda_m \mathcal{L} p_m(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_1} \\
&= \mathcal{L} \left(\sum_{m=1}^{L} \lambda_m p_m(\mathbf{x}) \right) \Big|_{\mathbf{x}=\mathbf{x}_1} \\
&= \mathcal{L} f \big|_{\mathbf{x}=\mathbf{x}_1}.
\end{aligned}
$$

□

Together these theorems imply that increasing the degree of the polynomial basis will ensure accuracy up to an order dependent on the degree and the order of the differential operator [40]. For a more extensive discussion of RBF-FD convergence rates, see [41, 42].

2.4 Vector Calculus Definitions and Identities

The Laplace operator (or the Laplacian) denoted Δ or ∇^2 is defined as the divergence of the gradient. For a function f, the Laplacian of f is given by

$$\Delta f \coloneqq \nabla \cdot (\nabla f) = \sum_k \frac{\partial^2}{\partial x_k^2} f$$

where x_k for $k = 1, 2, \ldots, d$ denote orthonormal coordinates for $\mathbb{R}^d$.

For a given manifold $\mathbb{M}$, the Laplace-Beltrami operator, denoted $\Delta_{\mathbb{M}}$ or $\nabla^2_{\mathbb{M}}$, is a generalization of the Laplacian to functions defined on $\mathbb{M}$. It can be expressed in terms of a parameterization of $\mathbb{M}$, however, forming parameterizations for arbitrary surfaces can be difficult and may lead to artificial coordinate singularities. In an effort

to avoid this, we will express $\Delta_{\mathbb{M}}$ in terms of the ambient Cartesian coordinate system. Specifically we will consider this in the context of 2-dimensional closed orientable manifolds embedded in $\mathbb{R}^3$.

For a point $\mathbf{x} \in \mathbb{M}$, let $\mathbf{n}(\mathbf{x})$ denote the outward-oriented unit normal to $\mathbb{M}$ at $\mathbf{x}$. Let $\tilde{f} : \mathbb{M} \to \mathbb{R}$ and let $f : \mathbb{R}^3 \to \mathbb{R}$ be any extension of $\tilde{f}$ such that it has continuous first partial derivatives. Then the surface gradient of $\tilde{f}$ is the component of the gradient of f that is tangent to the surface. Mathematically this is the component of the gradient of f that is orthogonal to the surface normal, or the projection into the plane tangent to the surface at $\mathbf{x}$. Symbolically that is

$$\begin{aligned} \nabla_{\mathbb{M}} \tilde{f}(\mathbf{x}) &= \nabla f - \text{Proj}_{\mathbf{n}} \nabla f \\ &= \nabla f - (\mathbf{n} \cdot \nabla f)\mathbf{n} \\ \nabla_{\mathbb{M}} \tilde{f} &= (I - \mathbf{n}\mathbf{n}^T)\nabla f \\ &= P \nabla f \end{aligned} \tag{2.8}$$

where P is the projection matrix $I - \mathbf{n}\mathbf{n}^T$ (note that P is dependent on $\mathbf{x}$). This formula holds when evaluating at points on $\mathbb{M}$, as $\nabla \tilde{f}$ is not defined off of the surface. Then we have that the Laplace-Beltrami operator is given by

$$\Delta_{\mathbb{M}} \tilde{f} = P \nabla \cdot (P \nabla f). \tag{2.9}$$

Equations (2.8) and (2.9) will be important to the Projected Gradient method described in Section 3.2.2.

Alternatively we can express the Laplace-Beltrami operator in terms of *local coordinates*. At a point $\mathbf{x} \in \mathbb{M}$ let $\mathbf{n}$ be the unit vector normal to the surface at $\mathbf{x}$. Let $\mathbf{t}_1, \mathbf{t}_2$ be orthogonal unit vectors that are orthogonal to $\mathbf{n}$. Now, $\mathbf{n}, \mathbf{t}_1, \mathbf{t}_2$ form a basis for $\mathbb{R}^3$, which we will refer to as the *local* coordinate system. Then for a function $\tilde{f} : \mathbb{M} \to \mathbb{R}$ and any extension $f : \mathbb{R}^3 \to \mathbb{R}$ with continuous second partial

derivatives, we have

$$\Delta_{\mathbb{M}}\tilde{f} = \frac{\partial^2 f}{\partial \mathbf{t}_1^2} + \frac{\partial^2 f}{\partial \mathbf{t}_2^2} \qquad \text{and} \qquad \Delta f = \frac{\partial^2 f}{\partial \mathbf{t}_1^2} + \frac{\partial^2 f}{\partial \mathbf{t}_2^2} + \frac{\partial^2 f}{\partial \mathbf{n}^2}.$$

Thus, we have the identity

$$\Delta_{\mathbb{M}} = \Delta - \frac{\partial^2}{\partial \mathbf{n}^2}. \tag{2.10}$$

Note that the local coordinates are dependent on $\mathbf{x}$, and thus $\frac{\partial^2}{\partial \mathbf{n}^2}$ is not simply the second derivative in the constant direction $\mathbf{n}$. Rather, we must differentiate the normal vector $\mathbf{n}$ to obtain

$$\begin{aligned} \frac{\partial^2}{\partial \mathbf{n}^2} &= \mathbf{n} \cdot \nabla(\mathbf{n} \cdot \nabla) \\ &= \underbrace{\big(\nabla \cdot \mathbf{n}\big)}_{\kappa} \big(\mathbf{n} \cdot \nabla\big) + \mathbf{n}^T H(\cdot)\mathbf{n}, \end{aligned}$$

where $\kappa = \nabla \cdot \mathbf{n}$ is the mean curvature. This equation converts the derivative with respect to the local coordinate system to an expression involving the gradient and the Hessian with respect to the global coordinate system. Equation (2.10) then gives the identity

$$\Delta_{\mathbb{M}} = \Delta - \kappa\big(\mathbf{n} \cdot \nabla\big) - \mathbf{n}^T H(\cdot)\mathbf{n} \tag{2.11}$$

which is proven in [43] and will be useful in Section 3.2.3.

Note that some authors (including the authors of [43]) use the notation $\frac{\partial^2}{\partial \mathbf{n}^2}$ to denote the derivative where $\mathbf{n}$ is fixed. If a vector $\mathbf{v}$ is fixed, then the directional derivatives of a function f with respect to that vector are given by

$$\begin{aligned}
\frac{\partial f}{\partial \mathbf{v}}(\mathbf{x}) &= \mathbf{v} \cdot \nabla f(\mathbf{x}) = \mathbf{v}^T \nabla f \\
\frac{\partial^2 f}{\partial \mathbf{v}^2}(\mathbf{x}) &= \frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T \nabla f \\
&= \mathbf{v}^T \nabla \big(\mathbf{v}^T \nabla f \big) \\
&= \mathbf{v}^T \nabla \big((\nabla f)^T \mathbf{v} \big) \\
&= \mathbf{v}^T H(f) \mathbf{v}.
\end{aligned}$$

For this reason, some authors give Equation (2.11) as $\Delta_{\mathbb{M}} = \Delta - \kappa \frac{\partial}{\partial \mathbf{n}} - \frac{\partial^2 f}{\partial \mathbf{n}^2}$.

CHAPTER 3

THREE RBF TECHNIQUES FOR APPROXIMATING THE LAPLACE-BELTRAMI OPERATOR

In this chapter we detail three RBF algorithms for finding finite difference weights that approximate the Laplace-Beltrami operator: the Projected Gradient method, the Hermite RBF-FD method (part of the Fast OGr method), and a new Tangent Plane method. All of these can be done using (almost) any RBF, however, our focus is on using PHSs which benefit greatly from appending polynomial basis terms. As noted in section 2.2.4, a full basis of polynomial terms will will be (nearly) linearly dependent when restricted to a manifold. Therefore we begin this chapter with a technique for overcoming this linear dependence.

3.1 Overcoming the Rank Deficiency of the Polynomial Basis

As mentioned in section 2.3, the addition of polynomial basis terms to the RBF interpolant leads to rank deficient or ill-conditioned interpolation matrices when interpolating functions on manifolds. This is due to linear dependence of polynomials on algebraic surfaces. For example, the functions $1, x^2, y^2, z^2$ are linearly dependent on a sphere and thus will generate linearly dependent columns in P from Equation (2.3). Even if the surface is not algebraic, it is likely that it will be approximately algebraic, at least locally, leading to ill-conditioned systems.

In [44] many methods to numerically bypass the rank deficiency of the polynomial basis terms are explored. Here we briefly elaborate on the most promising of these methods that employs the SVD and the Schur complement.

A more general form of of the system (2.6) is given by

$$\begin{bmatrix} A & P \\ P^T & Q \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \tag{3.1}$$

which we can express as two systems of equations

$$A\mathbf{c} + P\mathbf{d} = \mathbf{f} \qquad \text{and} \qquad P^T\mathbf{c} + Q\mathbf{d} = \mathbf{g}.$$

We first solve for $\mathbf{d}$ as follows:

$$\begin{aligned} Q\mathbf{d} - \mathbf{g} &= -P^T\mathbf{c} \\ &= -P^TA^{-1}A\mathbf{c} \\ &= -P^TA^{-1}(\mathbf{f} - P\mathbf{d}) \\ &= P^TA^{-1}P\mathbf{d} - P^TA^{-1}\mathbf{f} \\ P^TA^{-1}\mathbf{f} - \mathbf{g} &= (P^TA^{-1}P - Q)\mathbf{d} \\ \mathbf{d} &= (P^TA^{-1}P - Q)^{-1}(P^TA^{-1}\mathbf{f} - \mathbf{g}). \end{aligned} \tag{3.2}$$

Here, the matrix $P^TA^{-1}P - Q$ is called the *Schur complement* of the saddle point system (3.1), and may be rank deficient. In our case (with the exception of the Hermite method described in section 3.2.3), we have that $Q = 0$, and often find that P is column-rank deficient. Thus $P^TA^{-1}P - Q = P^TA^{-1}P$ will be rank deficient, and cannot be inverted. We instead use a pseudo-inverse of this matrix to solve for $\mathbf{d}$ in (3.2). Once $\mathbf{d}$ is computed, a simple substitution finds $\mathbf{c}$:

$$\mathbf{c} = A^{-1}(\mathbf{f} - P\mathbf{d}).$$

Note that the pseudo-inverse requires a tolerance parameter that is used to determine which singular values of the matrix are to be truncated. In practice we find

that choosing this tolerance to be 10^{-12} is effective.

3.2 Three RBF-FD Methods for Manifolds

Each of the methods investigated here seek to approximate a surface differential operator at a specified point on a surface, using information from nearby points on that surface. All of the methods in this section share some common notation that is discussed presently and summarized in Table 3.1.

Let $\mathbb{M} \subset \mathbb{R}^3$ denote a closed orientable manifold and let $\mathcal{L}_{\mathbb{M}}$ be a surface differential operator on $\mathbb{M}$. Let $X \subset \mathbb{M}$ be a discrete set of N nodes. Let $\mathbf{x}_1 \in X$ and let $\{\mathbf{x}_n\}_{n=1}^k \subset X$ denote the k nodes in X that are closest to $\mathbf{x}_1$ (itself included). We refer to the set $\{\mathbf{x}_n\}_{n=1}^k$ as the *stecil*, and to the point $\mathbf{x}_1$ as the *stencil center*. Our goal is to find finite difference weights $\{\omega_n\}_{n=1}^k \subset \mathbb{R}$ such that for any sufficiently smooth function $u : \mathbb{M} \to \mathbb{R}$

$$\mathcal{L}_{\mathbb{M}} u(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_1} \approx \sum_{n=1}^{k} \omega_n u(\mathbf{x}_n).$$

Each of the methods explored will also require $\mathbf{n}$, the unit vector normal to $\mathbb{M}$ at $\mathbf{x}_1$. For a few of the methods, it will be convenient to define $\mathcal{T}$ as the plane tangent to the surface at $\mathbf{x}_1$.

The finite difference weights that are generated will approximate $\mathcal{L}_{\mathbb{M}}$ at the single point $\mathbf{x}_1$. These processes may be repeated (possibly in parallel) to generate weights for each $\mathbf{x} \in X$, thus approximating $\mathcal{L}_{\mathbb{M}}$ over the entire surface.

3.2.1 Tangent Plane Method

In [21], Demanet describes a surprisingly simple finite difference approximation for surface operators. His method reduces the problem of finding finite difference

weights for a surface differential operator, to one of finding finite difference weights for points in $\mathbb{R}^2$. We couple his technique with 2-dimensional RBF-FD to create a new RBF finite difference technique for surface differential operators.

Consider the points $\mathbf{x}_n' = (I - \mathbf{n}\mathbf{n}^T)\mathbf{x}_n$ for $n = 1, 2, \ldots, k$ which are simply the projection[1] of the stencil onto $\mathcal{T}$ (note that $\mathbf{x}_1 = \mathbf{x}_1'$). Such a projection can be seen in Figure 3.1a. Let the weights $\{\omega_n\}_{n=1}^k$ be finite difference weights that approximate $\mathcal{L}_\mathcal{T}$ at $\mathbf{x}_1$. Then those same weights are a finite difference approximation of $\mathcal{L}_\mathbb{M}$ at $\mathbf{x}_1$.

To elaborate, let $u : \mathbb{M} \to \mathbb{R}$ and define $g : \mathcal{T} \to \mathbb{R}$ by $g(\mathbf{x}') = u(\mathbf{x})$ where $\mathbf{x}' = (I - \mathbf{n}\mathbf{n}^T)\mathbf{x}$. Then we find finite difference weights $\{\omega_n\}_{n=1}^k$ such that

$$\mathcal{L}_\mathcal{T} g(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_1} \approx \sum_{n=1}^{k} \omega_n g(\mathbf{x}_n')$$

and these same finite difference weights approximate $\mathcal{L}_\mathbb{M}$ via

$$\mathcal{L}_\mathbb{M} f(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_1} \approx \sum_{n=1}^{k} \omega_n f(\mathbf{x}_n).$$

[1] Technically, Demanet uses a slightly different projection than this.

Table 3.1: Table of symbols for Chapter 3.

$\mathbb{M}$	A smooth, orientable, closed manifold embedded in $\mathbb{R}^3$.
u	An arbitrary scalar function defined on $\mathbb{M}$.
$\mathcal{L}_\mathbb{M}$	A surface differential operator on $\mathbb{M}$.
∇, Δ	The gradient and Laplacian (respectively) in $\mathbb{R}^3$.
$\nabla_\mathbb{M}$, $\Delta_\mathbb{M}$	The surface gradient and surface Laplacian (respectively) on $\mathbb{M}$.
$\mathbf{x}_1$	The stencil center.
k	The number of points in the stencil.
$\{\mathbf{x}_n\}_{n=1}^k$	The stencil.
$\mathbf{u}$	A vector containing $\mathbf{u}_n = u(\mathbf{x}_n)$ for $n = 1, 2, \ldots, k$.
$\mathbf{n}$	The surface normal at $\mathbf{x}_1$.
$\mathcal{T}$	The plane tangent to $\mathbb{M}$ at $\mathbf{x}_1$.

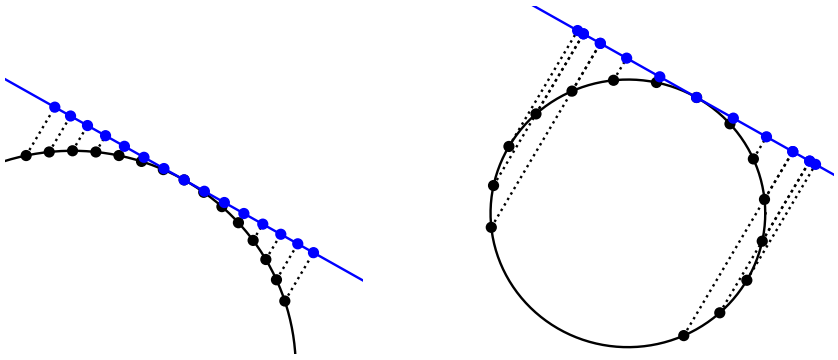
This is advantageous since it is much easier to construct finite difference weights for $\mathcal{L}_{\mathcal{T}}$. In the context of the Laplace-Beltrami operator, $\mathcal{L}_{\mathcal{T}} = \Delta_{\mathcal{T}}$ is simply the 2-dimensional Laplacian with respect to the intrinsic coordinates of $\mathcal{T}$.

To approximate $\mathcal{L}_{\mathcal{T}}$, we use Gram-Schmidt to find vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ such that $\mathbf{t}_1, \mathbf{t}_2, \mathbf{n}$ are pairwise orthonormal. Then $\mathbf{t}_1$ and $\mathbf{t}_2$ parameterize $\mathcal{T}$, and any point $\mathbf{x} \in \mathcal{T}$ can be represented as $\mathbf{x} = x\mathbf{t}_1 + y\mathbf{t}_2$ for a unique $(x, y) \in \mathbb{R}^2$. The problem is now reduced to finding finite difference weights for points in $\mathbb{R}^2$, for which, we choose to use 2-dimensional RBF-FD as described in Section 2.3.

Some efficiency can be gained by noting that one need not actually calculate $\{\mathbf{x}'_n\}_{n=1}^k$, but can instead compute the pairwise distances directly. We first translate the stencil so that $\mathbf{x}_1 = 0$. The matrix $R = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \mathbf{n} \end{bmatrix}$ is then an isometry such that R^T (when multiplying on the left) rotates points in the tangent plane to a plane that is parallel to the x-y plane. After this rotation, projection onto the tangent plane can be accomplished by simply dropping the z coordinate of the resulting vector. This is aesthetically similar to using a different metric in the formulation of the distance matrix. We define the "weighted distance" as $\text{dist}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{x}_1)^T W (\mathbf{y} - \mathbf{x}_1)}$ where the "weight matrix" W is given by $W = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \mathbf{0} \end{bmatrix}^T$.

There is a pitfall that must be avoided with this method. Defining g as the projection of f onto $\mathbb{M}$ is not necessarily well-defined. The projection operator is rank-deficient and may not be 1-to-1 on a domain containing the stencil. Figure 3.1b depicts such a case. If two points in the stencil get mapped to the same point (or nearly the same point) on $\mathcal{T}$ then the RBF interpolation matrix will be singular (or ill-conditioned). This can be overcome by increasing the total number of points on the surface, while keeping the stencil size constant. This will cause the points in X to more densely cover the surface, and will result in stencils that are roughly flat as

seen in Figure 3.1a.



(a) **(b)**

Figure 3.1: The projection of stencils (black) onto the tangent plane (blue). In (a) the stencil is nearly flat (co-planar), leading to a locally one-to-one projection. In (b) the stencil is not flat, the mapping is not one-to-one, and the resulting finite different weights will fail to approximate the operator.

Note that this method effectively reduces the dimension of the space by 1 which will require fewer polynomial terms to achieve a given order of accuracy as per Theorems 2.1 and 2.3.

3.2.2 Projected Gradient Method

The projection method was first introduced by Flyer and Wright in [12] using a global RBF approximation method on the sphere and extended later to RBF-FD in [13]. Fuselier and Wright generalized it to arbitrary surfaces using global RBFs [11] and this was later extended by Shankar et al. [15, 16] to RBF-FD.

The Projected Gradient method uses RBF-FD to approximate first order differential operators and then represents higher order operators as compositions of the approximations of the first order operators. In the case of the Laplace-Beltrami

operator we have $\Delta_{\mathbb{M}} = \nabla_{\mathbb{M}} \cdot \nabla_{\mathbb{M}}$, and so we first seek an approximation to $\nabla_{\mathbb{M}}$. We define the projection operator $\mathcal{P} : \mathbb{R}^3 \to \mathcal{T}$ as

$$\mathcal{P} = I - \mathbf{n}\mathbf{n}^T = \begin{bmatrix} \mathbf{p}^x & \mathbf{p}^y & \mathbf{p}^z \end{bmatrix}.$$

The surface gradient can be defined as an operator acting in the ambient space

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

where each $\mathcal{G}_x, \mathcal{G}_y, \mathcal{G}_z$ are linear operators in $\mathbb{R}^3$ that can be approximated via RBF-FD. For $\mathcal{G}_x$ we have

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} W_x^T \\ \Gamma_x^T \end{bmatrix} = \begin{bmatrix} \mathcal{G}_x A^T \\ \mathcal{G}_x P^T \end{bmatrix} \tag{3.3}$$

where A and P are from Equation (2.3), $(\mathcal{G}_x A^T)_{i,j} = \mathcal{G}_x \phi(\mathbf{x} - \mathbf{x}_i)|_{\mathbf{x}=\mathbf{x}_j}$, and $(\mathcal{G}_x P^T)_{i,j} = \mathcal{G}_x p_i(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_j}$. We discard Γ_x, and the matrix W_x contains the finite difference weights that approximate $\mathcal{G}_x$. Similar computations can be done to find W_y and W_z which approximate $\mathcal{G}_y$ and $\mathcal{G}_z$, respectively. Note that these matrices approximate their respective operators at each point in the stencil so that each row of W_x contains finite difference weights that approximate $\mathcal{G}_x$ at the corresponding point in the stencil. Therefore $W_x W_x \approx \mathcal{G}_x \mathcal{G}_x$ and we can approximate the surface Laplacian as

$$\Delta_{\mathbb{M}} \approx W = W_x W_x + W_y W_y + W_z W_z.$$

The matrix W will approximate $\Delta_{\mathbb{M}}$ at all points in the stencil, but the first row will give the weights that approximate $\Delta_{\mathbb{M}}$ at $\mathbf{x}_1$. Generally, the approximation for points farther away from the center will be worse. Unlike the other methods presented here, the Projected Gradient method will necessarily compute finite difference weights for

each point in a stencil, many of which will give particularly good approximations. Though we do not compare it here, the *overlapped* RBF-FD method [45] does exactly this and offers considerable speed-up by performing the computation on fewer stencils that overlap to cover the surface.

3.2.3 Hermite RBF-FD

Hermite RBF-FD for approximating surface operators was done by Piret and Dunnn in [20] as part of the Fast Orthogonal Gradients method (Fast OGr) — itself an improvement on the high-order and low-order RBF Orthogonal Gradients methods (RBF-OGr) introduced by Piret in [8].

Each of the RBF-OGr and Fast OGr methods have two distinct parts. First they use rough approximations of the surface normals to reconstruct the surface implicitly as the level set function of an RBF interpolant s. Since the surface is approximately the level set of s, we have that ∇s is approximately normal to the surface and can be used as a better approximation of the normal vectors. Second, they use these normal vectors to construct a function $f : \mathbb{R} \to \mathbb{R}$ that interpolates u at the nodes in such a way that ∇f is tangent to the surface (or equivalently orthogonal to ∇s hence the name Orthogonal Gradients). In the case of the original RBF-OGr methods, they used the improved approximation of the normal vectors to place points at an ε distance interior and exterior to the surface (tripling the number of points and adding ε as a hyper-parameter) and then used the closest-point mapping to extend f off of the surface. Fast OGr instead uses the improved approximation to the normal vectors and Hermite RBF-FD to create f that interpolates u at the nodes and directly enforces that ∇f is tangent to the surface.

Each of these parts is truly distinct in the sense that one can use the first part as a separate algorithm that only approximates surface normals, and if one already has good approximations to the surface normals, then one can use the second part as a distinct algorithm to construct the interpolant f with the desired property that ∇f is tangent to the surface. This thesis is not concerned with approximating surface normals, and thus we only consider the second part of Fast-OGr: Hermite RBF-FD.

As presented in [20], Fast OGr enforces that ∇s (where s is the interpolant) is tangent to the surface, and thus $\nabla s = \nabla_{\mathbb{M}} s \approx \nabla_{\mathbb{M}} u$. They then used this to approximate the three component differential operators $\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}$ with matrices D_x, D_y, D_z respectively, and (as in the Projected Gradient method) construct $D = D_x D_x + D_y D_y + D_z D_z \approx \Delta$ and thus $D\mathbf{u} \approx \Delta u \approx \Delta_{\mathbb{M}} u$.

Here, we instead use Hermite RBF-FD to approximate Δ directly. Recall the identity (2.11) from Section 2.4:

$$\Delta_{\mathbb{M}} u = \Delta u - \mathbf{n}^T H(u)\mathbf{n} - \kappa \frac{\partial u}{\partial \mathbf{n}}.$$

If we enforce that $\mathbf{n}^T H(u)\mathbf{n} = 0$ and $\frac{\partial u}{\partial \mathbf{n}} = 0$, we have that $\Delta u = \Delta_{\mathbb{M}} u$[2]. We will enforce these conditions on the stencil interpolant only at the stencil center $\mathbf{x}_1$, and thus define the functionals

$$\mathcal{L}_{\mathbf{x}}(\cdot) = \frac{\partial}{\partial \mathbf{n}}(\cdot)\bigg|_{\mathbf{x}=\mathbf{x}_1} \qquad \text{and} \qquad \mathcal{H}_{\mathbf{x}}(\cdot) = \mathbf{n}^T H(\cdot)\mathbf{n}\bigg|_{\mathbf{x}=\mathbf{x}_1}$$

for convenience.[3] In this new notation, we wish to find an RBF interpolant s that satisfies $\mathcal{L}_{\mathbf{x}} s = 0, \mathcal{H}_{\mathbf{x}} s = 0$, and still reproduces polynomials exactly (satisfying theorem 2.1). Following the method of *symmetric Hermite RBF-FD* [18, 19], we

[2]It is worth mentioning that in [8] this was this enforced approximately by the low-order and high-order methods respectively.

[3]Note that in Appendix A, we use this same notation to denote the same *operators* before evaluation at a point $\mathbf{x} = \mathbf{x}_1$.

use an interpolant of the form

$$s(\mathbf{x}) = \sum_{n=1}^{k} c_n \phi\left(\|\mathbf{x}-\mathbf{x}_n\|\right) + d\mathcal{L}_{\mathbf{y}}\phi\left(\|\mathbf{x}-\mathbf{y}\|\right) + e\mathcal{H}_{\mathbf{y}}\phi\left(\|\mathbf{x}-\mathbf{y}\|\right) + \sum_{m=1}^{L} \lambda_m p_m(\mathbf{x})$$

subject to the constraints

$$\begin{aligned} \mathcal{L}_{\mathbf{x}} s &= 0 \\ \mathcal{H}_{\mathbf{x}} s &= 0 \\ \sum_n c_n p_m(\mathbf{x}_n) + d\mathcal{L}_{\mathbf{x}} p_m(\mathbf{x}) + e\mathcal{H}_{\mathbf{x}} p_m(\mathbf{x}) &= 0, \quad \text{for} \quad m = 1, 2, \ldots L. \end{aligned}$$

Using the lemmas in Appendix A, this gives the interpolation matrix the form

$$\begin{bmatrix} A & \mathcal{L}\boldsymbol{\phi} & \mathcal{H}\boldsymbol{\phi} & P \\ \mathcal{L}\boldsymbol{\phi}^T & \mathcal{H}\phi_1 & \mathcal{L}\mathcal{H}\phi_1 & \mathcal{L}\mathbf{p}^T \\ \mathcal{H}\boldsymbol{\phi}^T & \mathcal{L}\mathcal{H}\phi_1 & \mathcal{H}\mathcal{H}\phi_1 & \mathcal{H}\mathbf{p}^T \\ P^T & \mathcal{L}\mathbf{p} & \mathcal{H}\mathbf{p} & 0 \end{bmatrix}, \tag{3.4}$$

where A is the standard RBF interpolation matrix, $\mathcal{L}\mathbf{p}_i = \mathcal{L}_{\mathbf{x}} p_i(\mathbf{x})$, $\mathcal{L}\boldsymbol{\phi}_i = \mathcal{L}_{\mathbf{x}}\phi(\|\mathbf{x}-\mathbf{x}_i\|)$, $\mathcal{H}\boldsymbol{\phi}_i = \mathcal{H}_{\mathbf{x}}\phi(\|\mathbf{x}-\mathbf{x}_i\|)$, and so on (as described in Appendix A, where the each of the operators is evaluated at $\mathbf{x} = \mathbf{x}_1$). Note that the center four blocks of this matrix are each 1×1 in size (that is to say they are scalars), the four corner blocks are matrices and the rest are vectors of the appropriate sizes. Thus, the two center columns are each column vectors, and the two center rows are each row vectors. The total size of the matrix is $(k+2+L) \times (k+2+L)$ where k is the stencil size and L is the number of polynomial terms.

The full system for finding finite difference weights is then given by

$$\begin{bmatrix} A & \mathcal{L}\boldsymbol{\phi} & \mathcal{H}\boldsymbol{\phi} & P \\ \mathcal{L}\boldsymbol{\phi}^T & \mathcal{H}\phi_1 & \mathcal{L}\mathcal{H}\phi_1 & \mathcal{L}\mathbf{p}^T \\ \mathcal{H}\boldsymbol{\phi}^T & \mathcal{L}\mathcal{H}\phi_1 & \mathcal{H}\mathcal{H}\phi_1 & \mathcal{H}\mathbf{p}^T \\ P^T & \mathcal{L}\mathbf{p} & \mathcal{H}\mathbf{p} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \gamma_1 \\ \gamma_2 \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \Delta\boldsymbol{\phi} \\ \Delta\mathcal{L}\boldsymbol{\phi}_1 \\ \Delta\mathcal{H}\boldsymbol{\phi}_1 \\ \Delta\mathbf{p} \end{bmatrix} \tag{3.5}$$

where $\boldsymbol{\omega}$ are the finite-difference weights.

Note that the Hermite RBF-FD method involves the computation of $\mathcal{H}\mathcal{H}$ and $\Delta\mathcal{H}$

which are fourth order surface differential operators. Consequentially, one cannot use PHS RBFs below ninth order as they are not sufficiently smooth (see the paragraph following Lemma A.7).

The addition of the $\mathcal{L}$ and $\mathcal{H}$ basis terms bring up the question of linear dependence. Indeed, with the addition of these terms the matrix given in (3.5) may be singular depending on the RBF used and the particular manifold. Two notable examples that induce singularity are 1) odd degree PHSs on the sphere, and 2) any choice of RBF on a plane (see Lemma A.8 and the paragraph that precedes it). This last example is particularly important as all manifolds will locally appear flat for small enough stencils. As with the singularity introduced by the polynomial terms, this can be ameliorated by use of the SVD approach detailed in Section 3.1.

CHAPTER 4

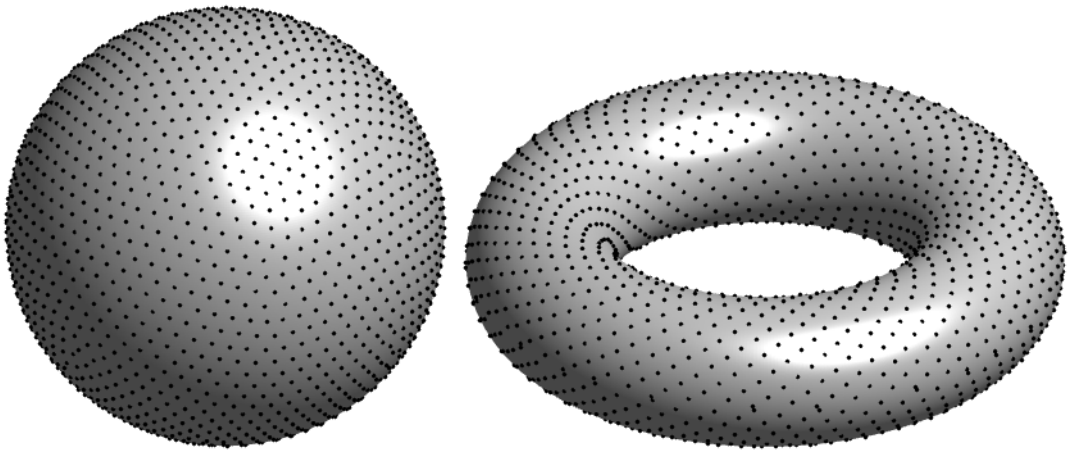
NUMERICAL RESULTS

In Section 4.1 we enumerate the parameters of each of the three methods presented in Chapter 3, review their restrictions, and suggest guidelines for parameter selections. Sections 4.2 and 4.3 use these guidelines to test accuracy and stability, respectively.

These tests are conducted over two manifolds: the unit sphere and a torus with major radius $r_{\text{major}} = 1$ and minor radius $r_{\text{minor}} = 1/3$. We discretize the sphere and the torus using the *minimum energy nodes* generated by the *spherepts* package [46] and the *phyllotaxis* or *spiral nodes*, respectively. Examples of these nodesets can be seen in Figure 4.1.

4.1 Parameter Selection

The three methods detailed in Chapter 3 each have the same three parameters that must be chosen: the stencil size k, the degree of PHS, and the degree of polynomial basis ℓ. Additionally the SVD method requires a tolerance parameter. We choose 10^{-12} unless otherwise specified and do not explore the effects of this parameter here. Note that the Tangent Plane method projects the points into a plane, where a basis of polynomials is linearly independent on that plane. Thus there is no polynomial dependence due to the surface itself, but polynomial dependence on the stencil is



(a) Minimum energy nodes on the sphere.

(b) Phyllotaxis nodes on the torus.

Figure 4.1: (a) $N = 2000$ minimum energy nodes on the sphere, and (b) $N = 2000$ phyllotaxis nodes on the torus. Note that the phyllotaxis nodes exhibit clustering around the outer "equator" of the torus. This is not the case when N is chosen to be Fibonacci number. Choosing such numbers will, in theory, give better stencils, however, it severely restricts the choices for N. We opted to choose N arbitrarily, so that we could test more configurations.

still possible. In practice, this is unlikely, and therefore the SVD method may be substituted for Gaussian elimination.

For the other three parameters, it is sensible that their selection should emanate from a desired order of convergence R. Polynomial reproduction guarantees a *theoretical* order of convergence due to Theorems 2.1 and 2.3, therefore we first select the degree of polynomial basis ℓ. As the Laplace-Beltrami operator is a second order differential operator, we should select $\ell = R + 1$.

We emphasize *theoretical* here, because the theorems suggest convergence as a single stencil is scaled in the ambient space. If stencils on manifolds are scaled, they will necessarily not lie on the manifold unless the manifold is locally linear. Additionally, stencils on manifolds will not usually tessellate, and thus we will be required to introduce new stencils that are distinct under scaling.

Next we choose the stencil size k, and to do so, we must first know the number of polynomials in our basis. Let $L = \binom{\ell+d}{\ell}$ be the number of polynomial terms in a basis of polynomials of degree ℓ in d dimensions. For the Projected Gradient and Hermite RBF-FD methods we let $d = 3$ and for the Tangent Plane method $d = 2$. In special circumstances this number can be reduced. For example, on the sphere, the $L = (\ell + 1)^2$ spherical harmonics of degree less than or equal to ℓ are an orthogonal basis of polynomials of that degree. Extra polynomial terms will be linearly dependent and the SVD method of Section 3.1 will yield the same finite difference weights.

If $k < L$, then the interpolation matrix in (2.6) will be singular due to the block structure of the matrix. In [47] and [40] they make two observations in the context of RBF-FD in $\mathbb{R}^2$. First, that increasing stencil size marginally increases accuracy, but does not change convergence rates. Second, that increasing the stencil size "improves" the eigenvalues of the differentiation matrix, in the sense that they more closely align with the eigenvalues of the differential operator. They hypothesize that this improvement may be due to one-sided stencils near the boundary of their domain.

In [47] they note that interpolation accuracy is moderately increased as the degree of PHS is increased. Of course with a higher degree of PHS, one requires augmentation with a higher degree of polynomial basis. It is therefore reasonable to choose the highest degree PHS available, namely $\phi(r) = r^{2\ell+1}$. Choosing higher degrees would go against PHS theory, however [47] concluded that choosing lower degree PHSs is stable. Note that the Hermite RBF-FD method fails for PHS degree below 9, and therefore we revert to the 9^{th} degree PHS whenever a lower degree is prescribed.

In conclusion, we recommend choosing parameters as follows. First choose a desired order of accuracy R. Then choose the degree of appended polynomials to be $\ell = R+1$. Let L be the number of polynomials in a basis of degree ℓ on the manifold

(or in $\mathbb{R}^2$ in the case of the Tangent Plane method), and choose $k \approx 2L$. Lastly, choose $\phi(r) = r^{2\ell+1}$.

4.2 Accuracy

We measure the accuracy of these methods in three tests each applied to the unit sphere and a torus with major radius $r_{\text{major}} = 1$ and minor radius $r_{\text{minor}} = 1/3$. Additionally, we test for polynomial reproduction by approximating the Laplace-Beltrami operator on the unit sphere, and applying the approximation to a spherical harmonic. The spherical harmonics are an orthogonal basis for polynomials restricted to the sphere, and thus we expect that any method that is exact for polynomials on the sphere, will be exact for the spherical harmonics.

For uniform grids, convergence rates for finite difference approximations are measured in h, the mesh width. Though there are ways to assign a value to h for unstructured points [27], we will take a different approach. For a uniform grid in 2 dimensions, we note that $h^{-1} \propto \sqrt{N}$ where N is the total number of points in the grid. Since surfaces are 2 dimensional, it is reasonable to use increasing $\sqrt{N}$ as a substitute for decreasing h for the purpose of measuring convergence rates.

4.2.1 Polynomial Reproduction

Our first test it to verify that our numerical results agree with Theorems 2.1 and 2.3. To test this, we choose $\mathbb{M}$ to be the unit sphere, and let

$$f(x, y, z) = 5x^4y - 10x^2y^3 + y^5,$$

a 5^{th} degree spherical harmonic. The spherical harmonics are eigenfunctions of the Laplace-Beltrami operator on the sphere, and thus $\Delta_{\mathbb{M}} f = -5(6)f$. We then

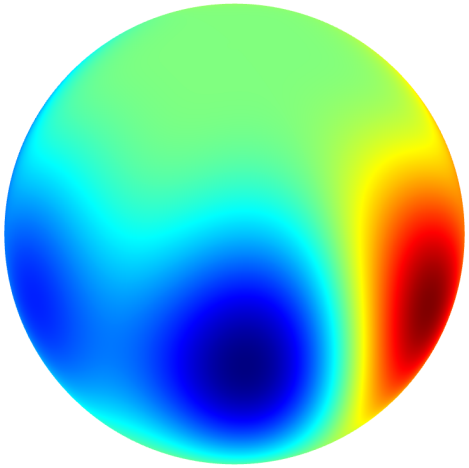
approximate $\Delta_{\mathbb{M}} f$ using each method and compare against the known analytical results. Figure 4.2 shows the function f and the results of the experiment. Six of the nine tests generally slope downward; as the number of points on the sphere increase, their accuracy improves. These methods are not reproducing the polynomial exactly, but rather are converging to the true solution. The remaining three methods are more accurate by several orders of magnitude, but the error increases as the total number of points is increased. These three methods are reproducing polynomials exactly but for round-off error. The larger the total number of points, the more error they accumulate.

The Hermite RBF-FD method reproduces the 5^{th} degree spherical harmonic when appending at least 5^{th} degree polynomials, as expected. The Projected Gradient method requires appending 6^{th} degree polynomials to be exact. This is due to the fact that the Projected Gradient method reproduces $\nabla_{\mathbb{M}}$ exactly rather than $\Delta_{\mathbb{M}}$. For a polynomial on the sphere, $\nabla_{\mathbb{M}}$ can be shown to be a polynomial of one degree higher. Thus, when the Projected Gradient method enforces reproduction of 6^{th} degree polynomials, the weights will exactly recover $\nabla_{\mathbb{M}}$ and $\Delta_{\mathbb{M}}$. Lastly, the Tangent Plane method fails to reproduce polynomials. As mentioned in Section 3.2.1, this is because polynomial reproduction is enforced on the tangent plane rather than on the manifold.

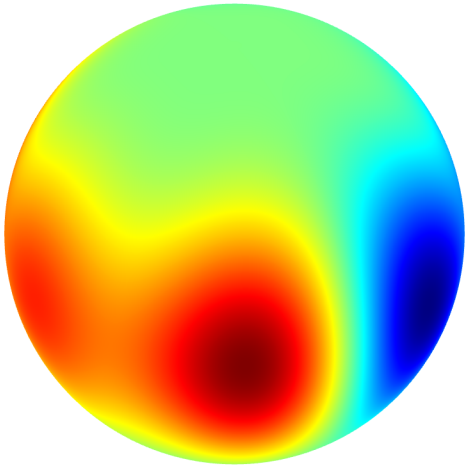
4.2.2 Differentiation

To test differentiation accuracy, we choose a test function on the sphere to be the sum of seven Gaussians. Specifically

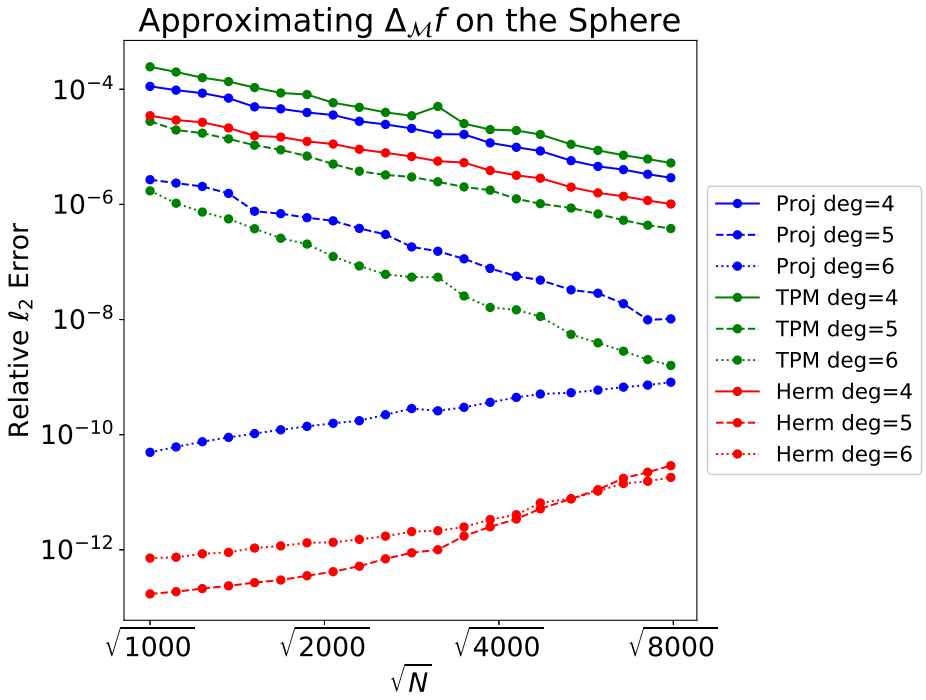
$$f(\mathbf{x}) = \sum_{i=1}^{7} e^{-\sigma_i r_i^2(\mathbf{x})} \tag{4.1}$$



(a) The spherical harmonic $f(x,y,z) = 5x^4y - 10x^2y^3 + y^5$. Note that $\Delta_{\mathbb{M}} f \propto f$.



(b) The same spherical harmonic from another angle.



(c) Approximation errors of $\Delta_{\mathbb{M}} f$.

Figure 4.2: (a,b) The 5^{th} degree spherical harmonic $f(x,y,z) = 5x^4y - 10x^2y^3 + y^5$ viewed from two angles. (c) Errors in approximating $\Delta_{\mathbb{M}} f$ using the three methods of Chapter 3. The Projected Gradient method appended with 6^{th} degree polynomials, and the Hermite RBF-FD method appended with 5^{th} or 6^{th} degree polynomials successfully reproduce the 5^{th} degree spherical harmonic.

with shapes

$$\begin{array}{llll} \sigma_1 = 2 & \sigma_2 = .5 & \sigma_3 = .3 & \sigma_4 = .1 \\ \sigma_4 = 5 & \sigma_5 = 2 & \sigma_6 = 1.5, & \end{array}$$

distance functions

$$r_i(\mathbf{x}) = \left\| \mathbf{x} - \tfrac{1}{\|\mathbf{c}_i\|}\mathbf{c}_i \right\|,$$

and centers

$$\begin{array}{llll} \mathbf{c}_1 = [0\ 0\ 1]^T & \mathbf{c}_2 = [1\ -1\ 1]^T & \mathbf{c}_3 = [2\ 0\ 1]^T & \mathbf{c}_4 = [-5\ 7\ 0]^T \\ \mathbf{c}_5 = [2\ -13\ 1]^T & \mathbf{c}_6 = [2\ 12\ -15]^T & \mathbf{c}_7 = [1\ 0\ -1]^T. & \end{array}$$

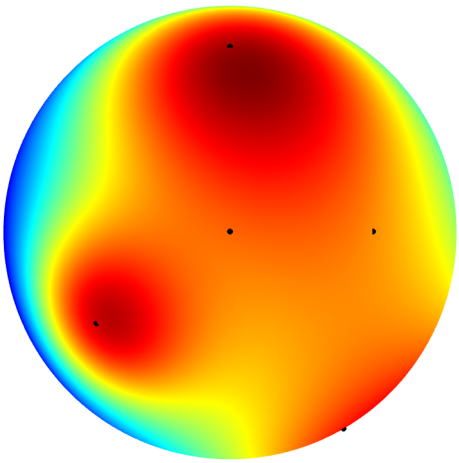
The exact solution of $\Delta_{\mathbb{M}} f$ is given by

$$\Delta_{\mathbb{M}} f(\mathbf{x}) = \sum_{i=1}^{7} -\sigma_i \left[4 + r_i^2(\mathbf{x}) \Big(-2 + \big(-4 r_i^2(\mathbf{x}) \big) \sigma_i \Big) \right] \exp\big(-\sigma_i r_i^2(\mathbf{x}) \big). \tag{4.2}$$

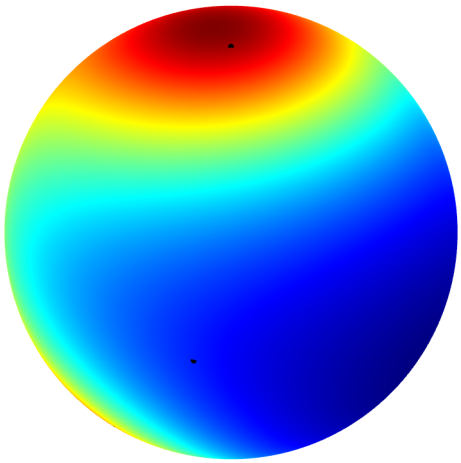
Both f and $\Delta_{\mathbb{M}} f$ can be seen in Figure 4.3.

Figure 4.4 shows the results of the approximation. As expected, each method attains convergence at a rate at least one order higher than the degree of polynomials appended, with one exception. The Tangent Plane Method when appending degree 5 polynomials, as measured by the ℓ^∞ norm measures the order of convergence to be roughly 3.78, slightly less than the predicted fourth degree convergence.

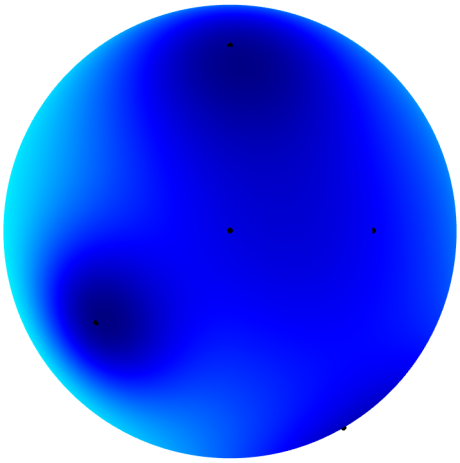
A similar test was performed on the torus. Figure 4.5 depicts a sum of seven Gaussians on the torus and the surface Laplacian of that function. Figure 4.6 shows the convergence results for the test on the torus. The convergence rates are as expected or better, however there are a few spikes in error for the ℓ^∞ norm in a few cases. As these spikes are not present for the ℓ^2 norm, we believe this indicates relatively poorer approximations on only a few stencils. This is likely due to the



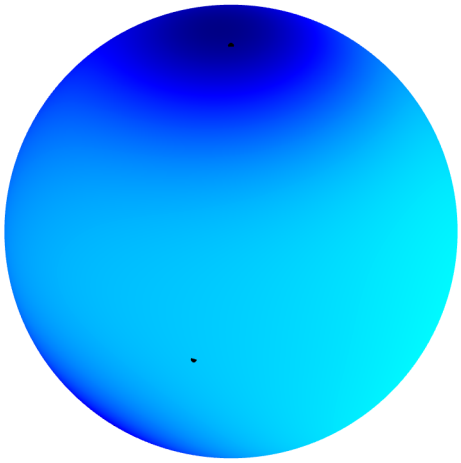
(a) The function f given in Equation (4.1) - a sum of seven Gaussians.



(b) The same function from another angle.

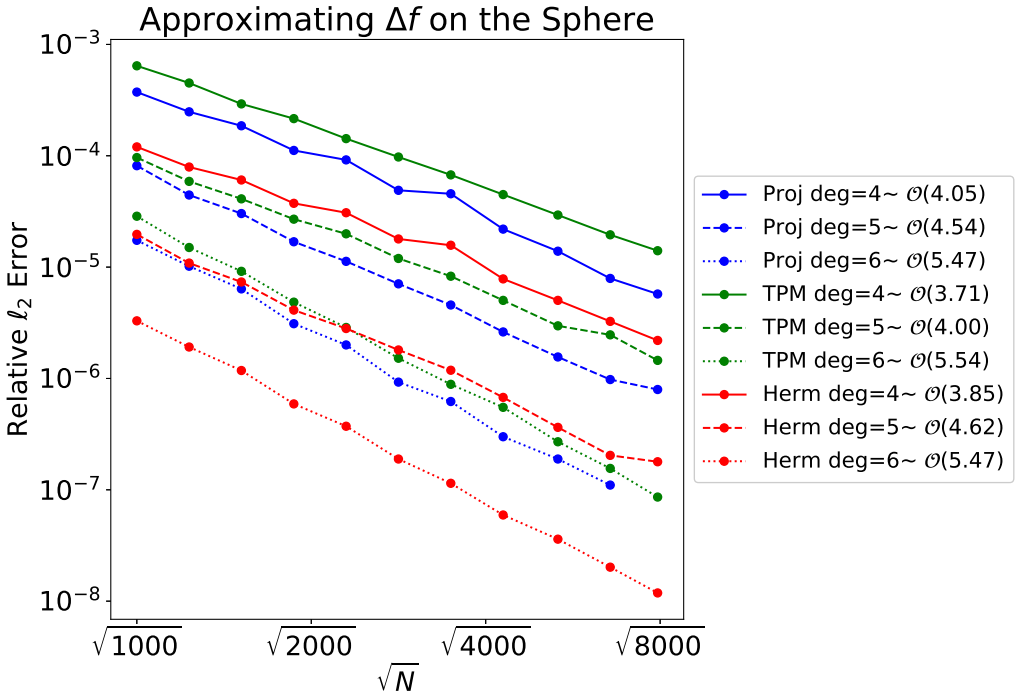


(c) A plot of $\Delta_{\mathbb{M}} f$.



(d) A plot of $\Delta_{\mathbb{M}} f$. from another angle.

Figure 4.3: (a)(b) The function f given in Equation (4.1) - a sum of seven Gaussians, with the centers of the Gaussians plotted in black. (c)(d) $\Delta_{\mathbb{M}} f$ viewed from the same angles as (a)(b) respectively.



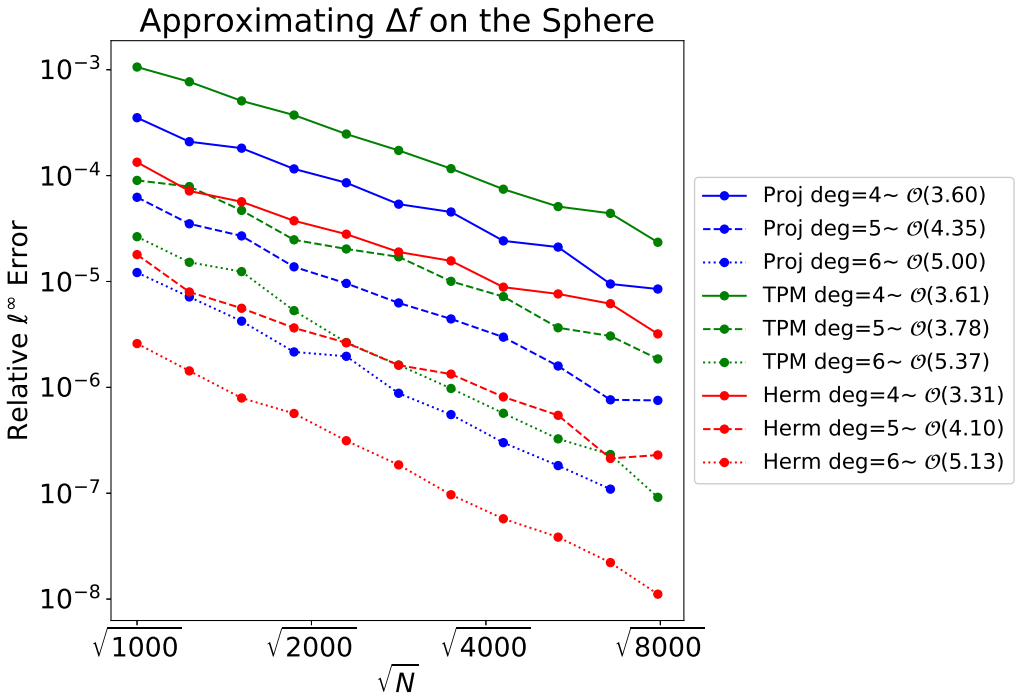
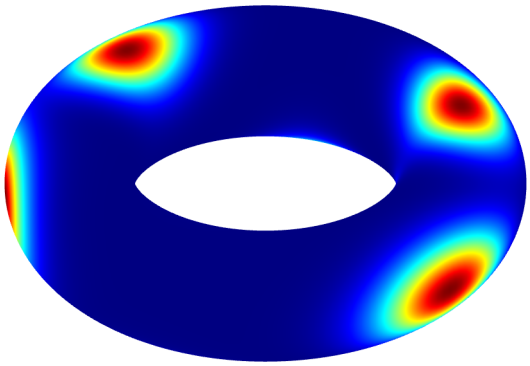
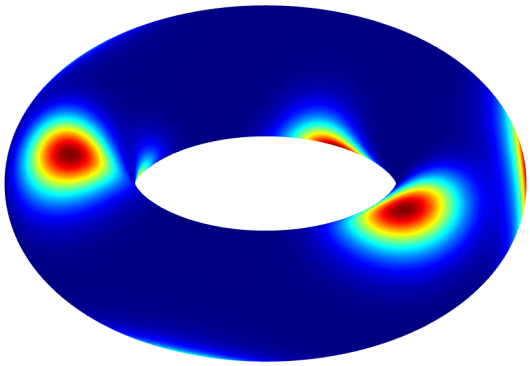


Figure 4.4: Error in ℓ^2 and ℓ^∞ norms, when approximating $\Delta_{\mathbb{M}} f$ on the sphere.

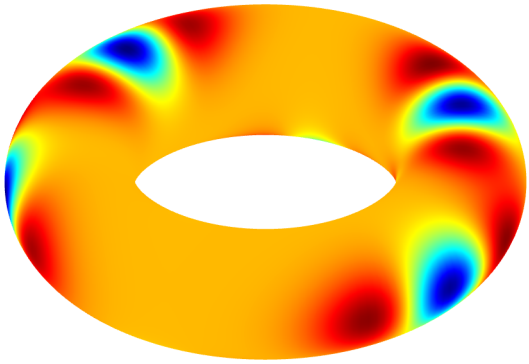
regularity, and occasional clustering of the phyllotaxial nodes.



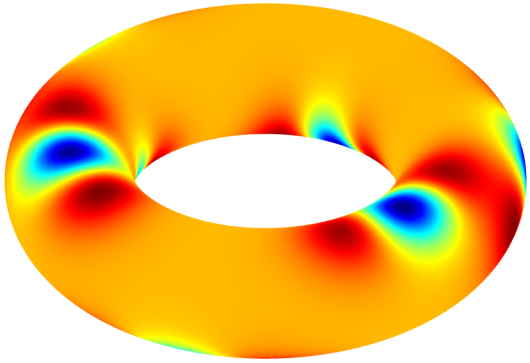
(a) A sum of seven Gaussians on the torus.



(b) The same function from another angle.



(c) The surface Laplacian of the sum of seven Gaussians



(d) The surface Laplacian of the sum of seven Gaussians from another angle.

Figure 4.5: (a)(b) A sum of seven Gaussians on the torus and (c)(d) the surface Laplacian of that sum.

4.2.3 Poisson Problem

Here we test accuracy by solving the Poisson problem $\Delta_{\mathbb{M}} u = g$, where g is chosen via the *method of manufactured solutions* to be $\Delta_{\mathbb{M}} f$ given by Equation (4.2). The true solution is then $u(\mathbf{x}) = f(\mathbf{x}) + C$ where f is given by (4.1), for any arbitrary constant C. The discrete version of the system $D\mathbf{u} = \mathbf{g}$ is also not uniquely solvable. To remedy this, we use a Lagrange multiplier.

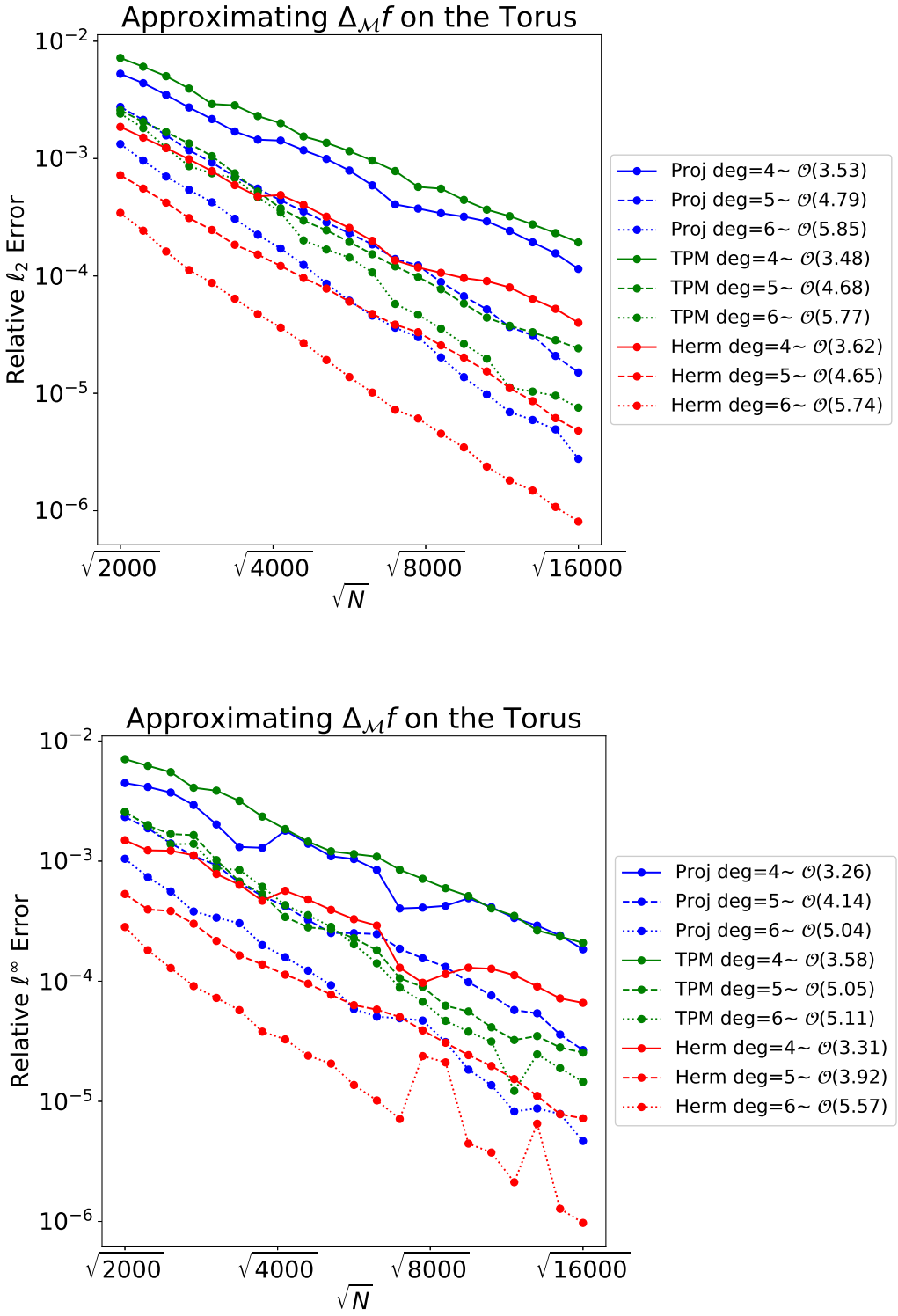


Figure 4.6: Error in ℓ^2 and ℓ^∞ norms, when approximating $\Delta_{\mathbb{M}} f$ on the torus.

To use Lagrange multipliers, we must pose this problem as a constrained minimization problem. Naturally, we want to minimize $\|D\mathbf{u} - \mathbf{g}\|_2$, but we must derive an appropriate constraint. In $u(\mathbf{x}) = f(\mathbf{x}) + C$, we would like to enforce that $C = 0$. From this, we have $\iint_{\mathbb{M}}(u - f)dA = \iint_{\mathbb{M}} CdA = 0$. Thus $\iint_{\mathbb{M}} udA = \iint_{\mathbb{M}} fdA$ is the constraint we will use. A convenient way of approximately enforcing this is

$$\sum_{n=1}^{N} u(\mathbf{x}_n) = b = \sum_{n=1}^{N} f(\mathbf{x}_n).$$

Then our problem can be posed as the system

$$\begin{bmatrix} D & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ b \end{bmatrix}$$

where $\lambda \geq 0$ is the Lagrange multiplier. We expect that $\lambda = 0$, as that corresponds to $D\mathbf{u} = \mathbf{g}$ being exactly satisfied. We used a direct sparse solver to solve this system, and in all test cases $\lambda = 0$ was verified up to machine precision.

Figures 4.7 and 4.8 show the results compared to the known solutions on the sphere and torus, respectively. The 5^{th} order methods on the sphere seem to converge at an order of roughly 3.5, otherwise the rates are above the expected minimum convergence rates.

4.2.4 Forced Diffusion Problem

Here we test accuracy by solving the forced diffusion problem

$$u_t = \Delta_{\mathbb{M}} u + g \tag{4.3}$$

where g is a forcing function dependent on both space and time. We again use the method of manufactured solutions by choosing a true solution u, and setting $g = u_t - \Delta_{\mathbb{M}} u$. We choose for the sphere and the torus, $u(\mathbf{x}, t) = e^{-t} f(\mathbf{x})$ where each

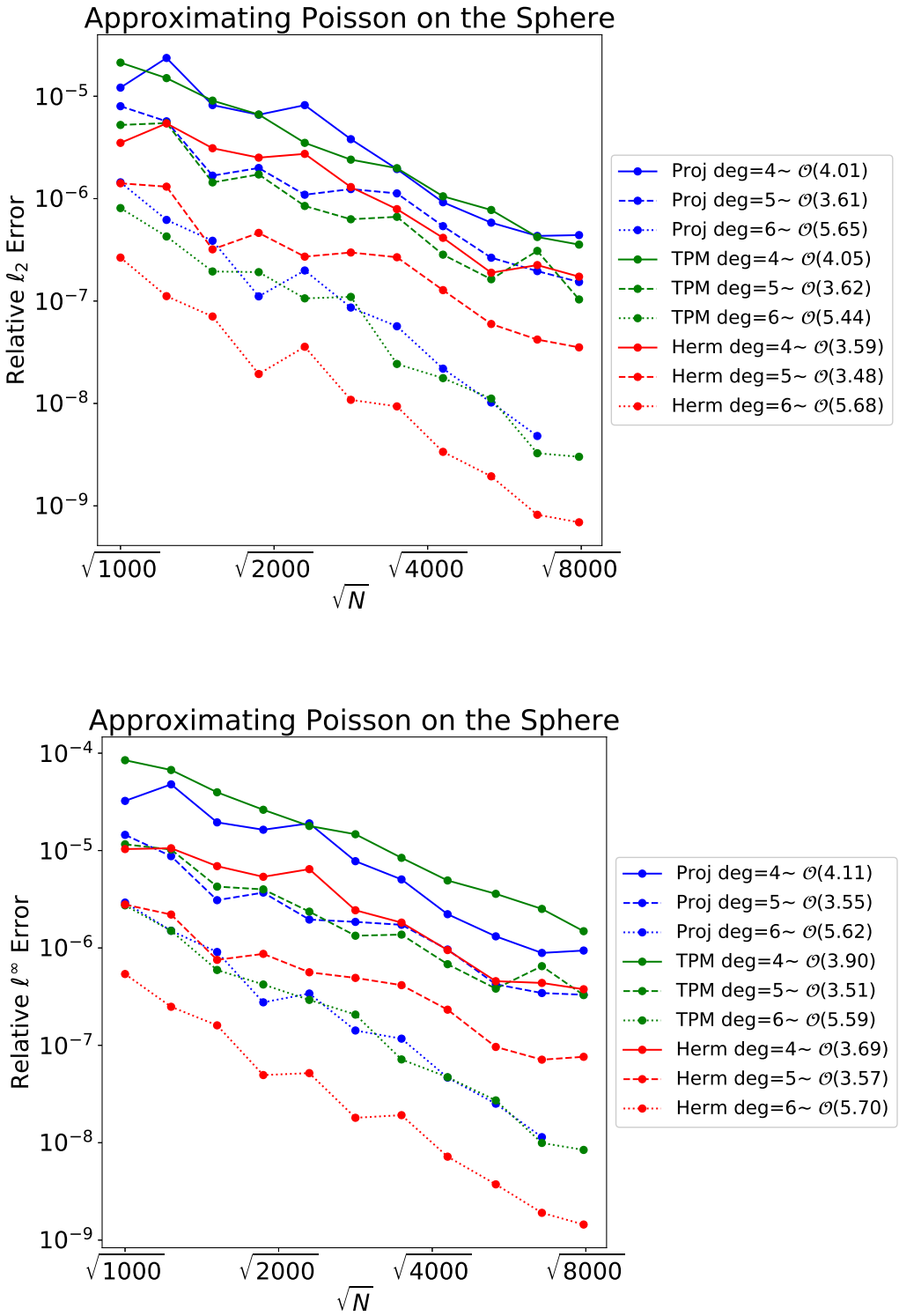


Figure 4.7: Error in ℓ^2 and ℓ^∞ norms, when solving the Poisson problem on the sphere.

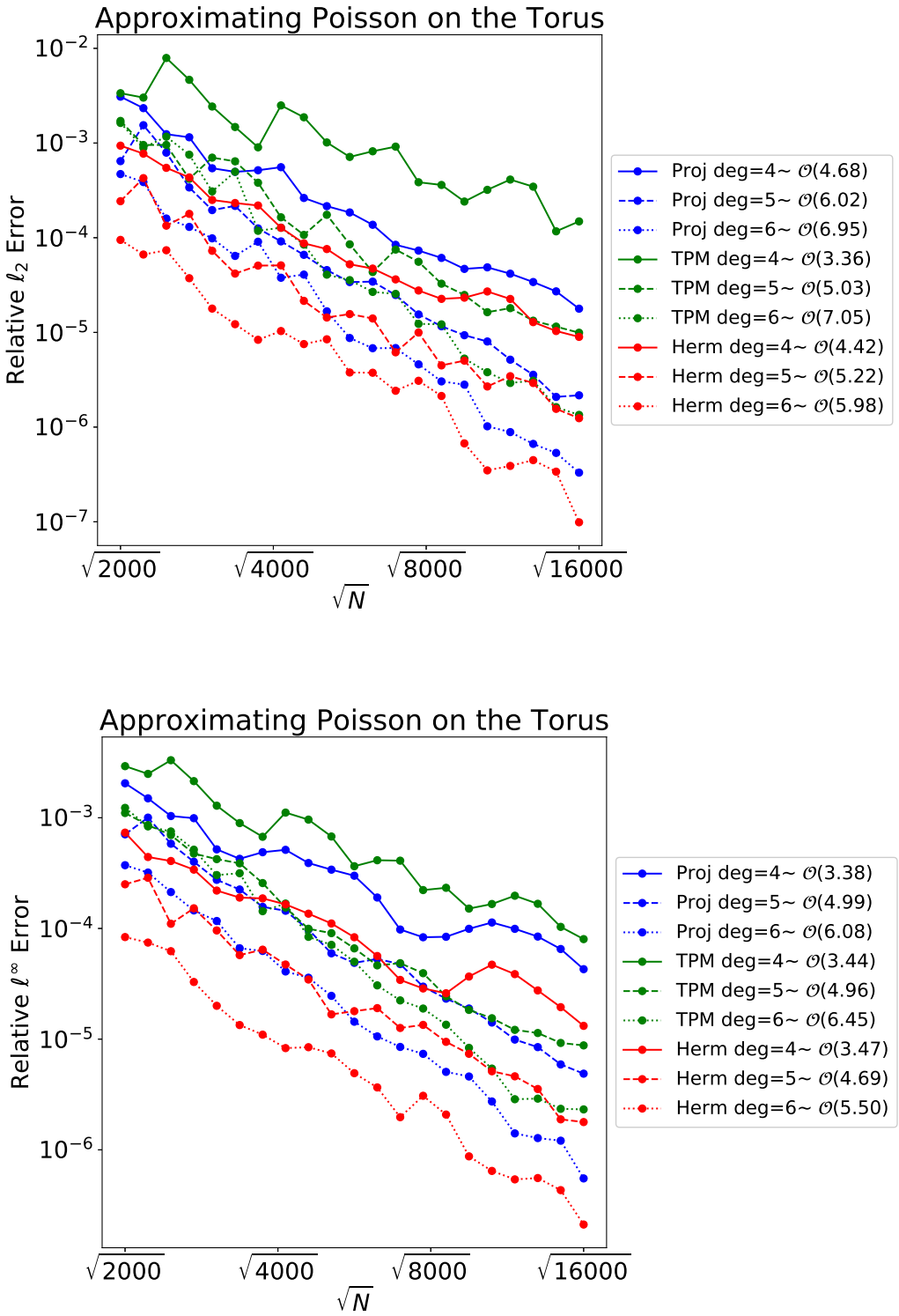


Figure 4.8: Error in ℓ^2 and ℓ^∞ norms, when solving the Poisson problem on the Torus.

f is the sum of seven Gaussians - the same functions chosen in the previous sections.

We will solve this using the method of lines approach, using each of our methods to approximate the spatial differential operator. Since our motivation is to test the accuracy and stability of the spatial approximation, we choose a high-order time stepping method and relatively small step sizes — just enough to showcase the differences in the spatial approximations. We choose $\Delta t = .005$ and take 200 steps using the third-order backward differentiation formula to a finial time of $t = 1$.

Figure 4.9 shows the results on both the sphere and the torus. All of the algorithms are stable, and they are all higher than fourth order. In particular, each of them is of higher order than the theoretical minimums guaranteed by Theorems 2.1 and 2.3. The Tangent Plane method seems to achieve the highest order of convergence, and the Hermite RBF-FD method has the highest total accuracy, with the Projected Gradient method being a middle ground for the two standards.

4.3 Stability

For time dependent problems, the stability of spatial finite difference approximations is measured by the eigenvalues of the differentiation matrix. These eigenvalues should align closely with the eigenvalues of the operator itself. The Laplace-Beltrami operator is a diffusive operator. For a given compact manifold without boundary, it has at least one eigenvalue of 0 (corresponding to a constant function), and the rest will be negative real numbers [48]. The specific values will depend on the surface. In particular, if the eigenvalues of our differentiation matrices have a positive real component, then any time-stepping scheme will be unstable.

For the sphere, the eigenfunctions are the spherical harmonics, and the eigenvalues

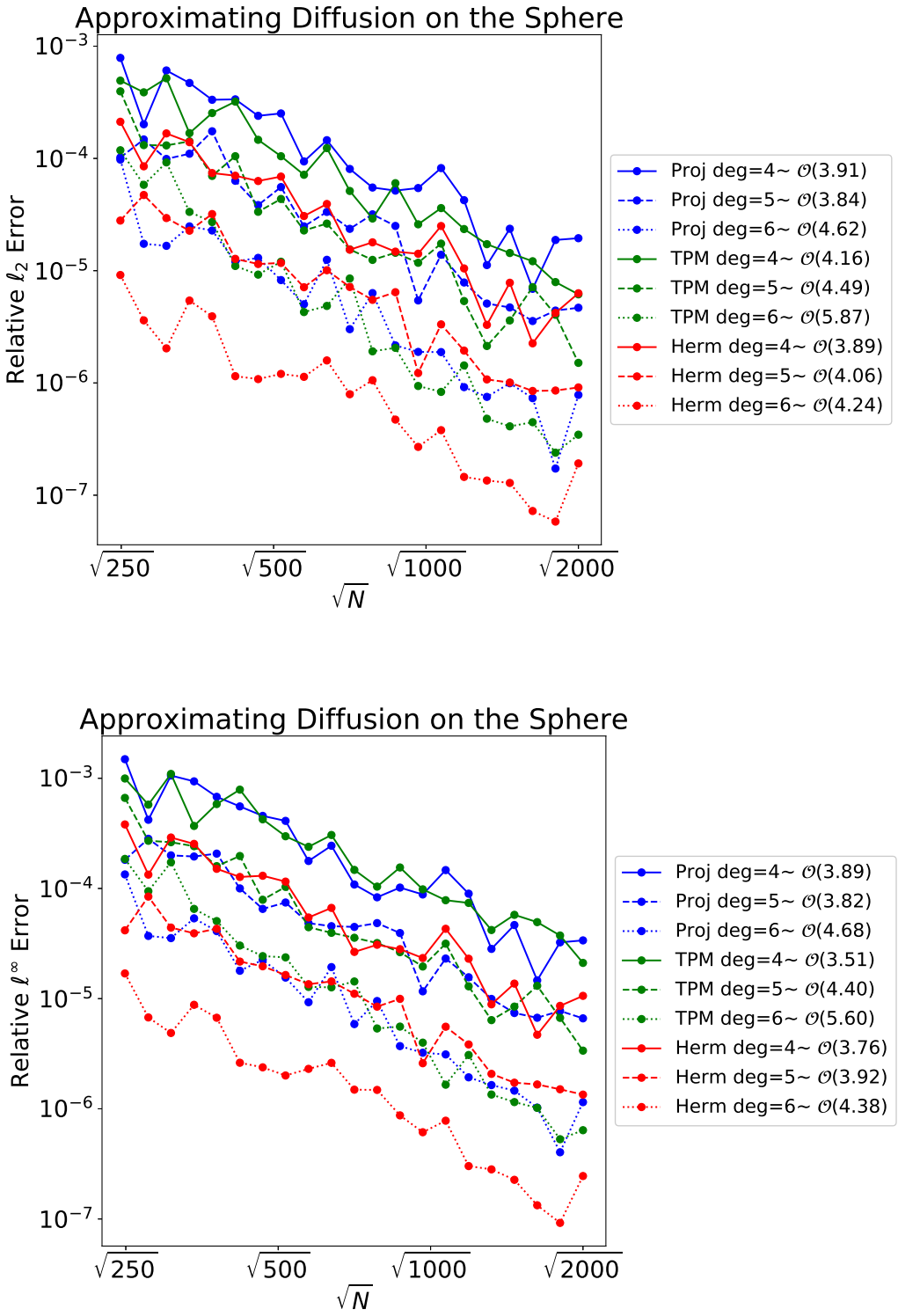


Figure 4.9: Error in ℓ^2 and ℓ^∞ norms, when solving the forced diffusion problem on the sphere.

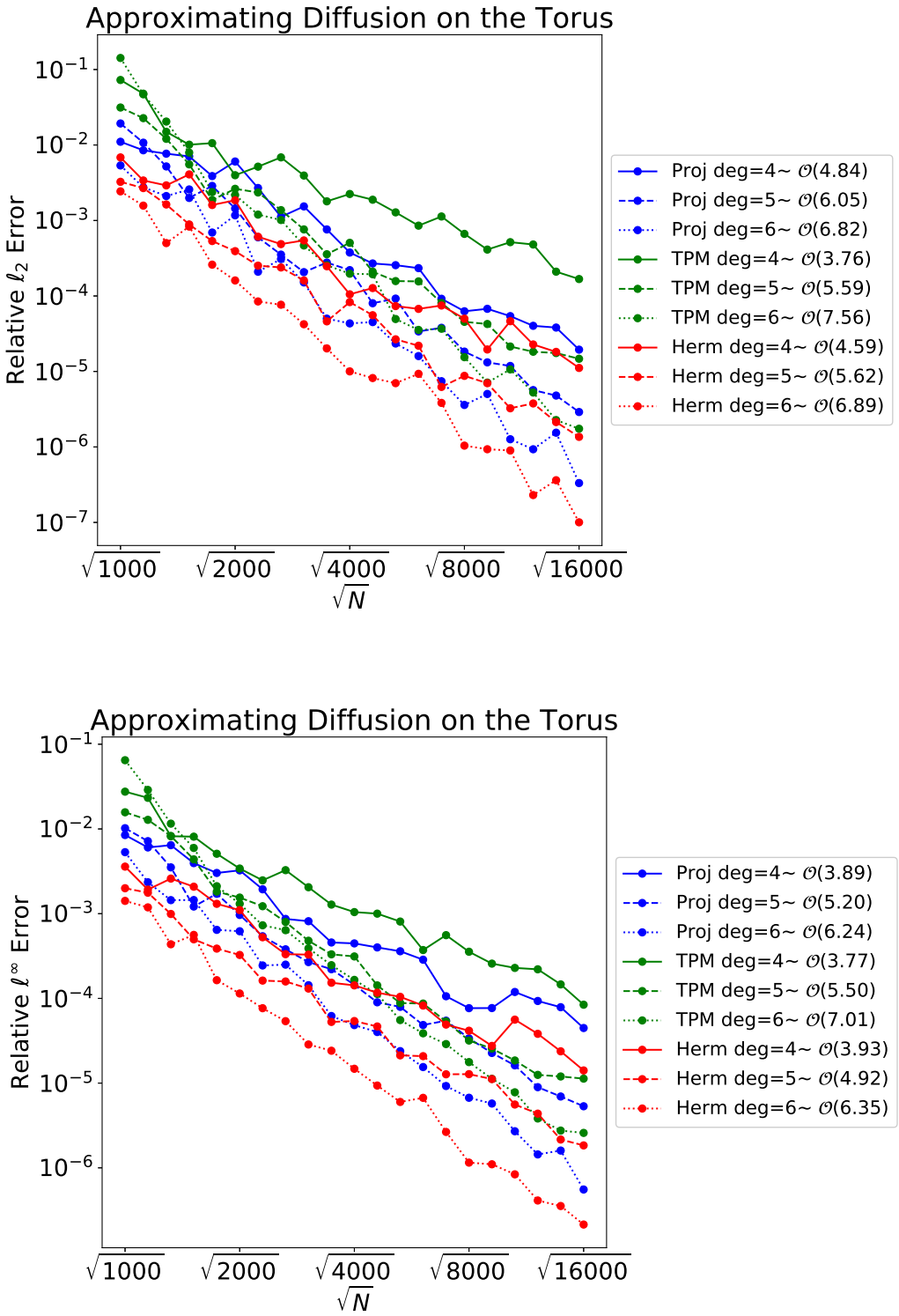


Figure 4.10: Error in ℓ^2 and ℓ^∞ norms, when solving the forced diffusion problem on the Torus.

are $-\ell(\ell+1)$ where ℓ is the degree of the spherical harmonic. Since our weights are exact for the spherical harmonics, we expect many of these to be exact, and hope that the remaining eigenvalues are close to eigenvalues of higher degree spherical harmonics.

Each subplot in Figures 4.11 and 4.12 shows the 100 eigenvalues with largest real component of the differentiation matrices. The matrices were generated by applying each method appended with 5^{th} degree polynomials to $N = 8000$ minimum energy nodes. Each subplot shows a range of stencil sizes k with the smallest stencils chosen to be the theoretical minimum: the dimension of a basis of up to 5^{th} degree polynomials. On the sphere, that would be $k = (5+1)^2 = 36$ spherical harmonics for the Projection and Hermite RBF-FD methods. On the torus, these are chosen to be a full basis of 3-variate polynomials: $k = \binom{5+3}{5} = 56$. For the Tangent Plane method, a full basis of 2-variate polynomials, $k = \binom{5+2}{5} = 21$, is chosen, for either surface.

Note that some of the subplots in Figures 4.11 and 4.12 are duplicated at a smaller scale in order to show the stencil sizes for which the eigenvalues of the matrices more closely align with the eigenvalues of the operator. For most choices of the stencil size, the eigenvalues of the matrices align closely with the eigenvalues of the differential operator, but there are a few exceptions. The Tangent Plane method seems to have spurious eigenvalues if the stencil size is selected at the minimum value, but these do not appear if the stencil size is at least 150% of the minimum value. Quite surprisingly, the Projected Gradient and Hermite RBF-FD methods have no eigenvalues with positive real part at the minimum stencil size, though the imaginary components become smaller (more closely approximating the eigenvalues of the differential operator) as the stencil size increases. To verify these results, we test against the forced diffusion problem from the previous section, this time varying the

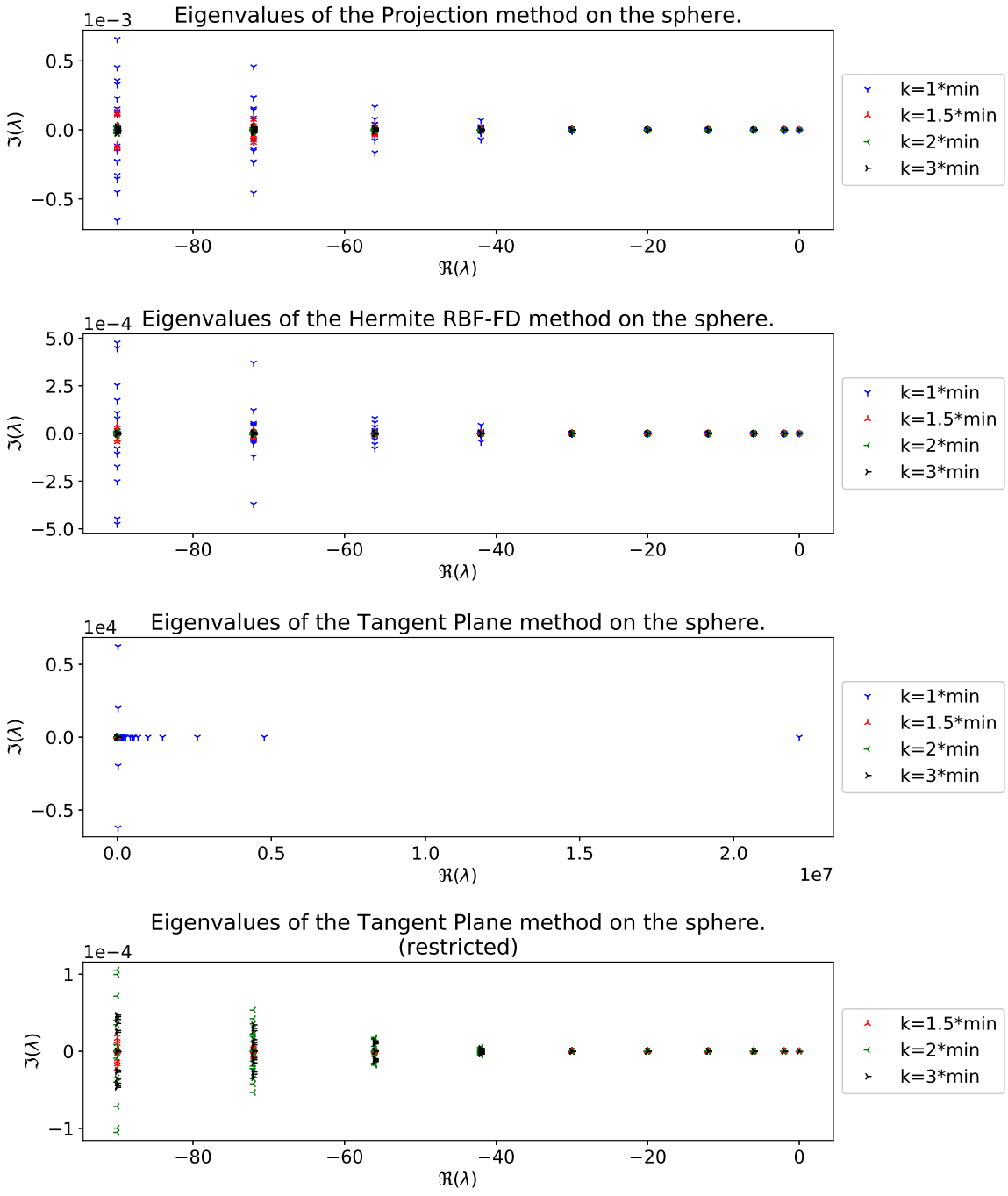


Figure 4.11: The 100 eigenvalues with the largest real parts for the differentiation matrices of each method when appending 5^{th} degree polynomials, and applied to the minimum energy nodes on the sphere. Several different stencil sizes k are compared. Here, the minimum stencil size k is given by the dimension of the polynomial basis. For the Projected Gradient and Hermite RBF-FD methods, that is the number of spherical harmonics. For the Tangent Plane method, that is a full basis for 2-variate polynomials.

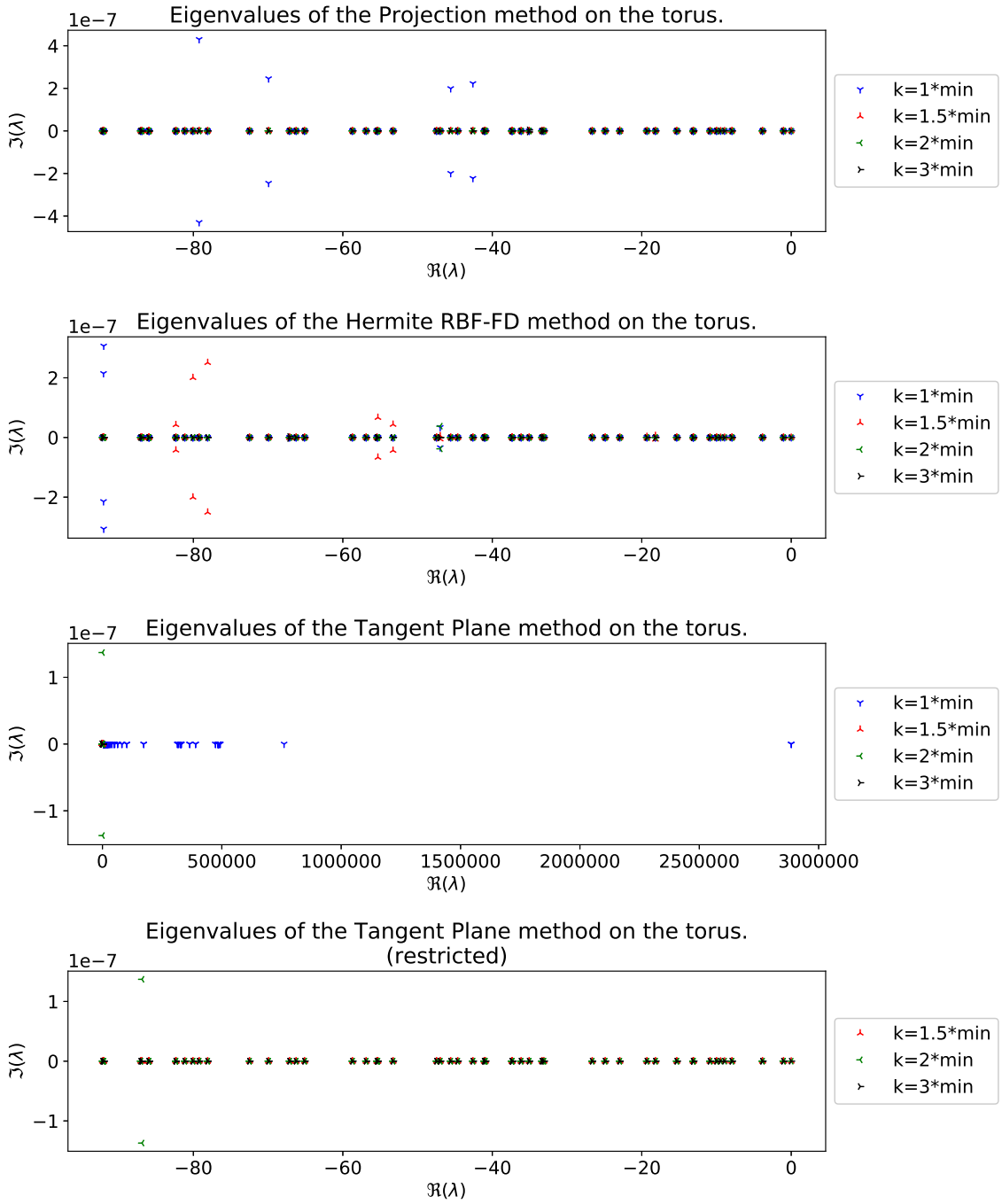


Figure 4.12: More eigenvalue plots as described in Figure 4.11, this time on the torus using the phyllotaxis nodes.

stencil size. The results can be seen in figure 4.13, and they agree with the assessment of the eigenvalues, except that there are values of N for which the minimum stencil sizes are not sufficient for the Projection and Hermite RBF-FD methods. In particular for our minimum energy nodeset with $N = 2070$, the minimum stencil size for the Projected Gradient and Hermite RBF-FD method yields matrices that are unstable for our time-stepping scheme. Additionally note, that though there is a legend entry for the Tangent Plane method with the minimum stencil size, it does not appear on the plot. None of the matrices generated using the Tangent Plane method and the minimum stencil size lead to stable time-stepping. Again, this agrees with our observation that the Tangent Plane method with the minimum stencil size generates matrices that have eigenvalues with positive real component.

4.4 Timings

As a final numerical comparison, we consider the time it takes to form a differentiation matrix versus the accuracy of a direct approximation of the operator. Figure 4.14 shows the time in seconds to form the differentiation matrices compared with the accuracy of the matrices in a direct computation of $\Delta_{\mathbb{M}}$. These stencils were the same stencils used in Figures4.4a and Figure 4.4b.

The timing results in Figure 4.14 demonstrate the efficiency of the Tangent Plane method, however they should be taken with a grain of salt. While the Tangent Plane method is very fast, we do not claim that the implementations of any of these methods are optimized. In particular, the Projected Gradient method can be made much faster by using overlapped stencils as in [45].

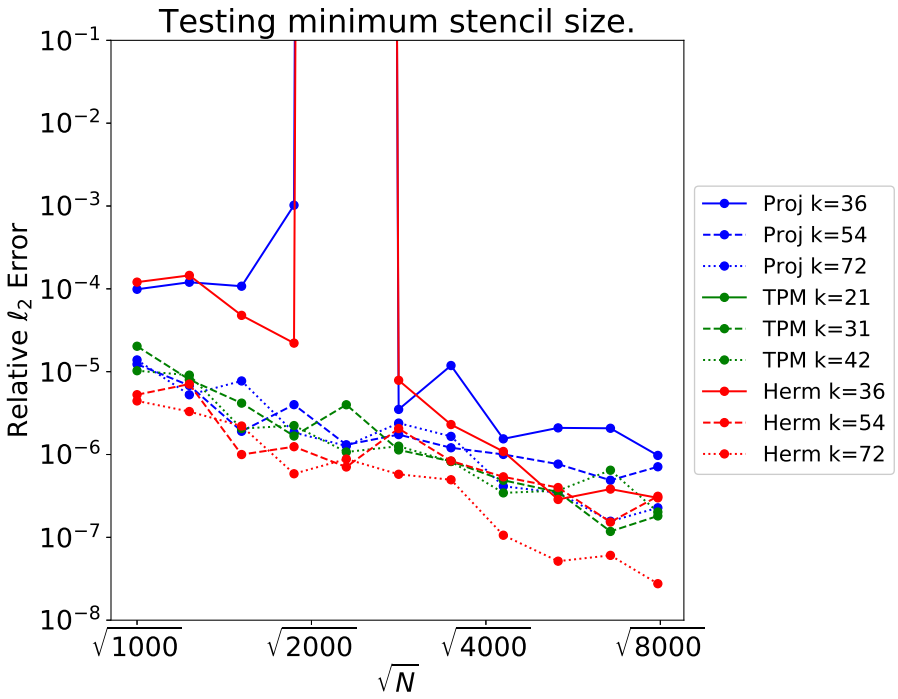


Figure 4.13: Error in ℓ^2 norm when solving the forced diffusion problem on the sphere for each method augmented with a 5^{th} degree polynomial basis. Several stencil sizes k are tested for each method that correspond to 1, 1.5, and 2 times the minimum. Note that for $N = 2070$, the Projected Gradients and Hermite RBF-FD methods lead to unstable time-stepping when the minimum stencil size is used. Also note that the Tangent Plane method using the minimum stencil size $k = 21$ appears on the legend but not on the plot. The Tangent Plane method with the minimum stencil size lead to unstable time-stepping for all values of N that were tested.

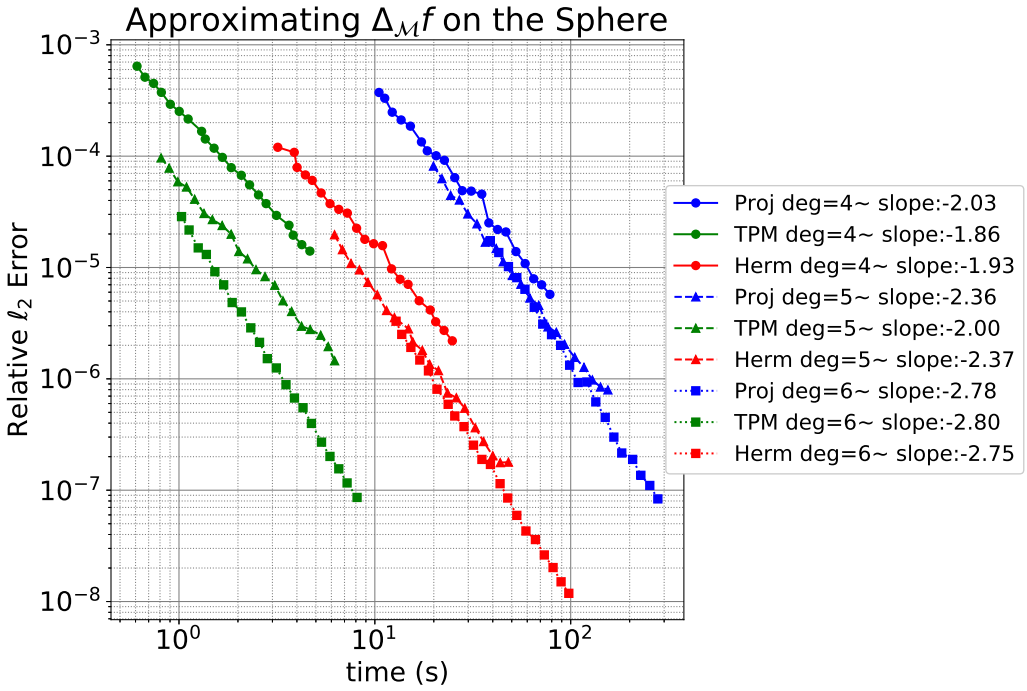


Figure 4.14: The time in seconds to form a differentiation matrix using each of the methods appended with polynomial basis for 4^{th}, 5^{th}, and 6^{th} degree polynomials, compared with the accuracy of computing $\Delta_{\mathbb{M}}$ on the sphere. The total number of points used ranged between 1000 and 8000, and were the same as in Figure 4.4a and Figure 4.4b.

CHAPTER 5

CONCLUSION

We have conducted the first direct comparison of the Projected Gradient method, the Hermite RBF-FD method, and the new Tangent Plane method for approximating the Laplace-Beltrami operator, and solving differential equations involving the Laplace-Beltrami operator. All methods proved accurate and stable in all three test cases, under appropriate parameter selection schemes. Given a desired order of accuracy R, parameters are chosen in the following ways:

1. Choose $\ell = R + 1$ to be the degree of polynomials to append.

2. Choose $\phi(r) = r^{2\ell+1}$. The minimum degree PHS for the Hermite RBF-FD method is r^9.

3. For the Projection and Hermite RBF-FD methods, let L be the dimension of a basis of 3-variate polynomials of degree ℓ on the surface. For the Tangent plane method, let L be the dimension of a basis of 2-variate polynomials of degree ℓ.

4. Choose $k \geq \frac{3}{2}L$. Larger stencils will have improved eigenvalues.

It is difficult to draw precise mathematical conclusions from a collection of convergence plots. Though, what amounts to a series of examples does not constitute a mathematical proof, it is noteworthy that they contain no counterexamples. That is

to say that with our prescribed parameter regime, each method successfully approximated the solution in each test with an observed convergence rate that was roughly on par with, or exceeded the theoretical convergence rate.

Nevertheless, we may make some observations. For a given degree of appended polynomials, it appears that the Hermite RBF-FD method is the most accurate, and the Projected Gradient method is usually the second most accurate. This agrees with the theory, since the Hermite method will reproduce polynomials for $\Delta_{\mathbb{M}}$, while the Projected Gradient method will reproduce polynomials for $\nabla_{\mathbb{M}}$, and the Tangent Plane does not reproduce polynomials in general. When it comes to the *order* of accuracy, the ranking seems to be reversed with the Tangent Plane as the highest, while the Projection and Hermite RBF-FD methods are roughly the same. Perhaps both the relatively sub-par accuracy and the above expected convergence rates of the Tangent Plane method are due to the stencils becoming increasingly flat under refinement. The Projection and Hermite RBF-FD methods would, in theory, have no benefit from this. The Tangent Plane method seems to be far faster than the other two methods for a given desired accuracy. This is expected given that our parameter selection regime mandates that the Projected Gradient and Hermite RBF-FD methods have larger stencil sizes and more terms in the polynomial basis than the Tangent Plane method for a desired order of accuracy.

Though we recommend choosing the stencil size to be greater than the minimum theoretical value, it is interesting that the minimum value appears to be sufficient most of the time for the Projection and Hermite RBF-FD methods. In [40], they theorized that stencil sizes may be chosen smaller when they do not contain boundary nodes. As none of our nodes are boundary nodes, this would seem to support their hypothesis.

We have succeeded in our goal to provide reliable parameter-selection guidelines, but there is still much to be explored. We have taken normal vectors as given, however they often must be approximated in applications where the surface is not known. Errors in the normals may have larger effects for some methods than others. Several parameters remained the same throughout all of our experiments. Most notable among them is our SVD tolerance and the choice of node set. Both of the sets of nodes that we used were relatively well-spaced. Computational complexity and speed, are two metrics that we hope to address in the future. In particular, incorporating the overlapping stencils technique of [45] to the Hermite RBF-FD method would make for a very fast, and highly accurate method.

Bibliography

[1] E. Kansa. "Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics—II solutions to parabolic, hyperbolic and elliptic partial differential equations". In: *Computers & Mathematics with Applications* 19.8 (1990), pp. 147–161.

[2] G. E. Fasshauer. "Solving Partial Differential Equations by Collocation with Radial Basis Functions". In: *In: Surface Fitting and Multiresolution Methods A. Le M'ehaut'e, C. Rabut and L.L. Schumaker (eds.), Vanderbilt.* University Press, 1997, pp. 131–138.

[3] C. Franke and R. Schaback. "Solving partial differential equations by collocation using radial basis functions". In: *Applied Mathematics and Computation* 93.1 (1998), pp. 73–82.

[4] G. B. Wright. "Radial Basis Function Interpolation: Numerical and Analytical Developments". PhD thesis. University of Colorado Boulder, 2003.

[5] D. A. Tolstykh A. I. and Shirobokov. "On using radial basis functions in a "finite difference mode" with applications to elasticity problems". In: *Computational Mechanics* 33.1 (Dec. 2003), pp. 68–79.

[6] C. Shu, H. Ding, and K. Yeo. "Local radial basis funcion-based differential quadrature method and its application to solve two-dimensional incompressible Navier-Stokes equations". In: *Computer Methods in Applied Mechanics and Engineering* 192 (Feb. 2003), pp. 941–954.

[7] T. Cecil, J. Qian, and S. Osher. "Numerical Methods for High Dimensional Hamilton-Jacobi Equations Using Radial Basis Functions". In: *J. Comput. Phys.* 196.1 (May 2004), pp. 327–347.

[8] C. Piret. "The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces". In: *Journal of Computational Physics* 231.14 (2012), pp. 4662–4675.

[9] V. Shankar, G. B. Wright, R. M. Kirby, and A. L. Fogelson. "A Radial Basis Function (RBF)-Finite Difference (FD) Method for Diffusion and Reaction—Diffusion Equations on Surfaces". In: *J. Sci. Comput.* 63.3 (June 2015), pp. 745–768.

[10] E. Lehto, V. Shankar, and G. B. Wright. "A Radial Basis Function (RBF) Compact Finite Difference (FD) Scheme for Reaction-Diffusion Equations on Surfaces". In: *SIAM Journal on Scientific Computing* 39 (Jan. 2017), A2129–A2151.

[11] E. J. Fuselier and G. B. Wright. "A High-Order Kernel Method for Diffusion and Reaction-Diffusion Equations on Surfaces". English. In: *Journal of Scientific Computing* 56 (2013), pp. 535–565.

[12] N. Flyer and G. B. Wright. "A radial basis function method for the shallow water equations on a sphere". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 465.2106 (2009), pp. 1949–1976.

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

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

[15] V. Shankar, G. B. Wright, R. M. Kirby, and A. L. Fogelson. "A Radial Basis Function (RBF)-Finite Difference (FD) Method for Diffusion and Reaction-Diffusion Equations on Surfaces". In: *J. Sci. Comput.* 63.3 (2014), pp. 745–768.

[16] V. Shankar, G. B. Wright, A. L. Fogelson, and R. M. Kirby. "A radial basis function (RBF) finite difference method for the simulation of reaction-diffusion equations on stationary platelets within the augmented forcing method". In: *International Journal for Numerical Methods in Fluids* 75.1 (2014), pp. 1–22.

[17] S. Ruuth and B. Merriman. "A simple embedding method for solving partial differential equations on surfaces". In: *Journal of Computational Physics* 227 (Jan. 2008), pp. 1943–1961.

[18] Z. Wu. "Hermite-Birkhoff interpolation of scattered data by radial basis functions". In: *Approximation Theory and its Applications* 8.2 (June 1992), pp. 1–10.

[19] F. J. Narcowich and J. D. Ward. "Generalized Hermite interpolation via matrix-valued conditionally positive definite functions". In: *Mathematics of Computation* 63.208 (Oct. 1994), pp. 661–687.

[20] C. Piret and J. Dunn. "Fast RBF OGr for solving PDEs on arbitrary surfaces". In: *AIP Conference Proceedings* 1776.1 (2016), p. 070005.

[21] L. Demanet. "Painless, highly accurate discretizations of the Laplacian on a smooth manifold". In: *Technical report, Stanford University* (2006).

[22] B. Fornberg. "Generation of finite difference formulas on arbitrarily spaced grids". In: *Mathematics of Computation* 51.184 (1988), pp. 699–706.

[23] R. J. Leveque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.

[24] J. C. Mairhuber. "On Haar's theorem concerning Chebychev approximation problems having unique solutions". In: *Proceedings of the American Mathematical Society* 7.4 (1956), pp. 609–615.

[25] P. C. Curtis. "n-parameter families and best approximation." In: *Pacific J. Math.* 9.4 (1959), pp. 1013–1027.

[26] R. L. Hardy. "Multiquadric equations of topography and other irregular surfaces". In: *Journal of Geophysical Research (1896-1977)* 76.8 (1971), pp. 1905–1915.

[27] G. F. Fasshauer. *Meshfree Approximation Methods with MATLAB*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2007.

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

[29] W. Cheney and W. Light. *A Course in Approximation Theory*. Brooks/Cole Publishing Company, 2000.

[30] R. Franke. *A Critical Comparison of Some Methods for Interpolation of Scattered Data*. 1979.

[31] J. H. Halton. "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals". In: *Numerische Mathematik* 2.1 (Dec. 1960), pp. 84–90.

[32] S. Rippa. "An algorithm for selecting a good value for the parameter c in radial basis function interpolation". In: *Adv. Comput. Math.* 11 (1999), pp. 193–210.

[33] G. E. Fasshauer and J. G. Zhang. "On choosing "optimal" shape parameters for RBF approximation". In: *Numerical Algorithms* 45 (2007), pp. 345–368.

[34] B. Fornberg and G. B. Wright. "Stable computation of multiquadric interpolants for all values of the shape parameter". In: *Comput. Math. Appl.* 48 (2004), pp. 853–867.

[35] B. Fornberg and C. Piret. "A stable algorithm for flat radial basis functions on a sphere". In: *SIAM J. Sci. Comput.* 30 (2007), pp. 60–80.

[36] G. B. Wright and B. Fornberg. "Stable computation with flat radial basis functions using vector-valued rational approximations". In: *J. Comput. Phys.* 331 (2017), pp. 137–156.

[37] B. Fornberg, E. Larsson, and N. Flyer. "Stable computations with Gaussian radial basis functions". In: *SIAM J. Sci. Comput.* 33 (2011), pp. 869–892.

[38] E. Larsson and B. Fornberg. "Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions". In: *Computers & Mathematics with Applications* 49.1 (2005), pp. 103–130.

[39] B. Fornberg, T. Driscoll, G. B. Wright, and R. Charles. "Observations on the behavior of radial basis function approximations near boundaries". In: *Computers & Mathematics with Applications* 43.3 (2002), pp. 473–490.

[40] V. Bayona, N. Flyer, B. Fornberg, and G. A. Barnett. "On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs". In: *Journal of Computational Physics* 332 (2017), pp. 257–273.

[41] O. Davydov and R. Schaback. “Error Bounds for Kernel-based Numerical Differentiation”. In: *Numer. Math.* 132.2 (Feb. 2016), pp. 243–269.

[42] O. Davydov and R. Schaback. “Optimal stencils in Sobolev spaces”. In: *IMA Journal of Numerical Analysis* 39.1 (Dec. 2017), pp. 398–422.

[43] J. Xu and H. Zhao. “An Eulerian Formulation for Solving Partial Differential Equations Along a Moving Interface”. In: *Journal of Scientific Computing* 19.1 (Dec. 2003), pp. 573–594.

[44] D. Malmuth. “Meshfree semi-Lagrangian schemes for advection on surfaces: polyharmonic splines augmented with polynomials”. (Work in Progress). MS Thesis. Boise State University, 2019.

[45] V. Shankar. “The overlapped radial basis function-finite difference (RBF-FD) method: A generalization of RBF-FD”. In: *Journal of Computational Physics* 342 (2017), pp. 211–228.

[46] G. B. Wright. *SpherePts*. `https://github.com/gradywright/spherepts`. 2018.

[47] N. Flyer, B. Fornberg, V. Bayona, and G. A. Barnett. “On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy”. In: *Journal of Computational Physics* 321 (2016), pp. 21–38.

[48] M. Obata. “Certain conditions for a Riemannian manifold to be isometric with a sphere”. In: *J. Math Soc. Japan* 14.3 (Mar. 1962), pp. 333–340.

Appendices

APPENDIX A

HERMITE RBF-FD LEMMAS

Section 3.2.3 makes extensive use of the operators

$$\mathcal{L}_{\mathbf{x}}(\cdot) = \mathbf{n} \cdot \nabla_{\mathbf{x}}(\cdot) \qquad \text{and} \qquad \mathcal{H}_{\mathbf{x}}(\cdot) = \mathbf{n}^T H_{\mathbf{x}}(\cdot)\mathbf{n}.$$

We dedicate this appendix to proving several lemmas regarding these operators. Note that this notation differs slightly from that of Section 3.2.3. Here we use this notation to refer the *operators*, where in Section 3.2.3 this notation is used to refer to the *functionals* that are given by evaluating the operators at a stencil center. Also note that we treat the vector $\mathbf{n}$ as a constant direction and thus it is not differentiated with successive application of the operators.

Before stating and proving the lemmas, let us first establish some common notations. For a given RBF ϕ, let $\phi_0(r) = \phi(r)$ and $\phi_{n+1}(r) = \frac{1}{r}\frac{d}{dr}\phi_n$. Let $\mathbf{x} = [x\ y\ z]^T$ and $\mathbf{y} = [a\ b\ c]^T$ be vectors in $\mathbb{R}^3$, and let $r = \|\mathbf{x} - \mathbf{y}\|$. Lastly, let $p = \mathbf{n} \cdot (\mathbf{x} - \mathbf{y})$ and $\mathbf{n} = [n_x\ n_y\ n_z]^T$ for notational convenience.

Lemma A.1.

$$\mathcal{L}_{\mathbf{x}}\phi_n(r) = p\phi_{n+1}(r) = -\mathcal{L}_{\mathbf{y}}$$

Proof: See that

$$\begin{aligned}\frac{dr}{dx} &= \frac{d}{dx}\big((x-a)^2+(y-b)^2+(z-c)^2\big)^{1/2}\\ &= 2(x-a)\frac{1}{2}\big((x-a)^2+(y-b)^2+(z-c)^2\big)^{-1/2}\\ &= (x-a)\frac{1}{r}\\ \frac{d}{dx}\phi_n(r) &= \phi_n'(r)\frac{dr}{dx} = (x-a)\phi_{n+1}(r),\end{aligned}$$

and similarly

$$\begin{aligned}\nabla_{\mathbf{x}}\phi_n(r) &= (\mathbf{x}-\mathbf{y})\phi_{n+1}(r)\\ \mathcal{L}_{\mathbf{x}}\phi_n(r) &= \mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\phi_{n+1}(r)\\ &= p\phi_{n+1}(r).\end{aligned}$$

Also since $\frac{d}{da}\phi(r) = -(x-a)\phi_1(r) = -\frac{d}{dx}\phi(r)$ it follows that $\mathcal{L}_{\mathbf{y}} = -\mathcal{L}_{\mathbf{x}}$. □

Lemma A.2. For $n \in \mathbb{N}$,

$$\mathcal{L}_{\mathbf{x}}p^{n+1} = (n+1)p^n.$$

Proof: Consider

$$\begin{aligned}\frac{\partial}{\partial x}\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^{n+1} &= (n+1)\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^n n_x\\ \frac{\partial}{\partial y}\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^{n+1} &= (n+1)\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^n n_y\\ \frac{\partial}{\partial z}\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^{n+1} &= (n+1)\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^n n_z\\ \nabla_{\mathbf{x}}\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^{n+1} &= (n+1)\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^n \mathbf{n}\\ \mathcal{L}_{\mathbf{x}}\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^{n+1} &= \mathbf{n}\cdot\nabla_{\mathbf{x}}\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^{n+1}\\ &= (n+1)\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^n\mathbf{n}\cdot\mathbf{n}\\ &= (n+1)\big(\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\big)^n.\end{aligned}$$

□

Lemma A.3.

$$\mathcal{H}_{\mathbf{x}}\phi(r) = \mathcal{L}_{\mathbf{x}}\mathcal{L}_{\mathbf{x}}\phi(r) = \phi_1(r) + p^2\phi_2(r) = \mathcal{H}_{\mathbf{y}}\phi$$

Proof: From Lemma A.1 we have that $\mathcal{L}_{\mathbf{x}}\phi(r) = p\phi_1(r)$. Then

$$\begin{aligned}\mathcal{L}_{\mathbf{x}}\mathcal{L}_{\mathbf{x}}\phi(r) &= \mathcal{L}_{\mathbf{x}}\big(p\phi_1(r)\big) \\ &= \phi_1(r)\mathcal{L}_{\mathbf{x}}p + p\mathcal{L}_{\mathbf{x}}\phi_1(r) \\ &= \phi_1(r) + p^2\phi_2(r).\end{aligned}$$

Next consider

$$\begin{aligned}\frac{\partial}{\partial x}\phi(r) &= (x-a)\phi_1(r) \\ \frac{\partial^2}{\partial x^2}\phi(r) &= \phi_1(r) + (x-a)^2\phi_2(r) \\ \frac{\partial^2}{\partial xy}\phi(r) &= (x-a)(y-b)\phi_2(r).\end{aligned}$$

Then

$$\begin{aligned}\mathcal{H}_{\mathbf{x}}\phi(r) &= \mathbf{n}^T \begin{bmatrix} \frac{\partial^2\phi}{\partial x^2} & \frac{\partial^2\phi}{\partial xy} & \frac{\partial^2\phi}{\partial xz} \\ \frac{\partial^2\phi}{\partial xy} & \frac{\partial^2\phi}{\partial y^2} & \frac{\partial^2\phi}{\partial yz} \\ \frac{\partial^2\phi}{\partial xz} & \frac{\partial^2\phi}{\partial yy} & \frac{\partial^2\phi}{\partial z^2} \end{bmatrix} \mathbf{n} \\ &= n_x^2\frac{\partial^2\phi}{\partial x^2} + n_y^2\frac{\partial^2\phi}{\partial y^2} + n_z^2\frac{\partial^2\phi}{\partial z^2} + 2n_xn_y\frac{\partial^2\phi}{\partial xy} + 2n_xn_z\frac{\partial^2\phi}{\partial xz} + 2n_yn_z\frac{\partial^2\phi}{\partial yz} \\ &= (n_x^2 + n_y^2 + n_z^2)\phi_1(r) + \big(n_x(x-a) + n_y(y-b) + n_z(z-c)\big)^2\phi_2(r) \\ &= \phi_1(r) + p^2\phi_2(r).\end{aligned}$$

Lastly we have that $\mathcal{H}_{\mathbf{x}}\phi = \mathcal{L}_{\mathbf{x}}\mathcal{L}_{\mathbf{x}}\phi = (-\mathcal{L}_{\mathbf{y}})(-\mathcal{L}_{\mathbf{y}})\phi = \mathcal{H}_{\mathbf{y}}\phi$. □

Lemma A.4.

$$\mathcal{L}_{\mathbf{x}}\mathcal{H}_{\mathbf{x}}\phi(r) = 3p\phi_2(r) + p^3\phi_3(r)$$

Proof: From Lemma A.3 we have

$$\begin{aligned}\mathcal{L}_{\mathbf{x}}\mathcal{H}_{\mathbf{x}}\phi(r) &= \mathcal{L}_{\mathbf{x}}\big(\phi_1(r) + p^2\phi_2(r)\big)\\ &= \mathcal{L}_{\mathbf{x}}\phi_1(r) + \phi_2(r)\mathcal{L}_{\mathbf{x}}p^2 + p^2\mathcal{L}_{\mathbf{x}}\phi_2(r)\\ &= p\phi_2(r) + 2p\phi_2(r) + p^3\phi_3(r)\\ &= 3p\phi_2(r) + p^3\phi_3(r).\end{aligned}$$

□

Lemma A.5.

$$\mathcal{H}_{\mathbf{x}}\mathcal{H}_{\mathbf{x}}\phi(r) = 3\phi_2(r) + 6p^2\phi_3(r) + p^4\phi_4(r)$$

Proof: From Lemmas A.3 and A.4 we have

$$\begin{aligned}\mathcal{H}_{\mathbf{x}}\mathcal{H}_{\mathbf{x}}\phi(r) &= \mathcal{L}_{\mathbf{x}}\mathcal{L}_{\mathbf{x}}\mathcal{H}_{\mathbf{x}}\phi(r)\\ &= \mathcal{L}_{\mathbf{x}}\big(3p\phi_2(r) + p^3\phi_3(r)\big)\\ &= 3\phi_2(r)\mathcal{L}_{\mathbf{x}}p + 3p\mathcal{L}_{\mathbf{x}}\phi_2(r) + \phi_3(r)\mathcal{L}_{\mathbf{x}}p^3 + p^3\mathcal{L}_{\mathbf{x}}\phi_3(r)\\ &= 3\phi_2(r) + 3p^2\phi_3(r) + 3p^2\phi_3(r) + p^4\phi_4(r)\\ &= 3\phi_2(r) + 6p^2\phi_3(r) + p^4\phi_4(r).\end{aligned}$$

□

Lemma A.6.

$$\Delta\mathcal{L}_{\mathbf{x}}\phi(r) = 5p\phi_2(r) + r^2p\phi_3(r)$$

Proof: From Lemma A.1 it follows that

$$\begin{aligned}\frac{d}{dx}\mathcal{L}_{\mathbf{x}}\phi(r) &= \frac{d}{dx}p\phi_1(r)\\ &= n_x\phi_1(r) + p(x-a)\phi_2(r)\\ \frac{d^2}{dx^2}\mathcal{L}_{\mathbf{x}}\phi(r) &= n_x(x-a)\phi_2(r) + n_x(x-a)\phi_2(r) + p\phi_2(r) + p(x-a)^2\phi_3(r)\\ &= 2n_x(x-a)\phi_2(r) + p\phi_2(r) + p(x-a)^2\phi_3(r)\\ \Delta\mathcal{L}_{\mathbf{x}}\phi(r) &= 5p\phi_2(r) + r^2p\phi_3(r).\end{aligned}$$

□

Lemma A.7.

$$\Delta\mathcal{H}_{\mathbf{x}}\phi(r) = 5\phi_2(r) + \big(r^2 + 7p^2\big)\phi_3(r) + p^2r^2\phi_4(r)$$

Proof: From Lemma A.3 it follows that

$$\begin{aligned}
\frac{d}{dx}\mathcal{H}_{\mathbf{x}}\phi(r) &= \frac{d}{dx}\big(\phi_1(r) + p^2\phi_2(r)\big) \\
&= (x-a)\phi_2(r) + 2pn_x\phi_2(r) + p^2(x-a)\phi_3(r) \\
\frac{d^2}{dx^2}\mathcal{H}_{\mathbf{x}}\phi(r) &= \phi_2(r) + (x-a)^2\phi_3(r) + 2n_x^2\phi_2(r) + 2pn_x(x-a)\phi_3(r) + \\
&\qquad 2pn_x(x-a)\phi_3(r) + p^2\phi_3(r) + p^2(x-a)^2\phi_4(r) \\
&= \phi_2(r) + (x-a)^2\phi_3(r) + 2n_x^2\phi_2(r) + 4pn_x(x-a)\phi_3(r) \\
&\qquad + p^2\phi_3(r) + p^2(x-a)^2\phi_4(r) \\
\Delta\mathcal{H}_{\mathbf{x}}\phi(r) &= 3\phi_2(r) + r^2\phi_3(r) + 2\phi_2(r) + 4p^2\phi_3(r) + 3p^2\phi_3(r) + p^2r^2\phi_4(r) \\
&= 5\phi_2(r) + \big(r^2 + 7p^2\big)\phi_3(r) + p^2r^2\phi_4(r).
\end{aligned}$$

□

Note that both of the expressions for $\Delta\mathcal{H}_{\mathbf{x}}\phi(r)$ and $\mathcal{H}_{\mathbf{x}}\mathcal{H}_{\mathbf{x}}\phi(r)$ involve $\phi_4(r)$. This function is not smooth for PHS of degree 8 or less. To see this, consider that for $\phi(r) = r^\ell$ where ℓ is odd, we have that

$$\phi_1(r) = \frac{1}{r}\frac{d}{dr}\phi = \frac{1}{r}\ell r^{\ell-1} = \ell r^{\ell-2}$$

and for $\phi(r) = r^\ell \log(r)$ where ℓ is even we have that

$$\phi_1(r) = \frac{1}{r}\frac{d}{dr}\phi = \frac{1}{r}\big(\ell r^{\ell-1}\log(r) + r^\ell\frac{1}{r}\big) = \big(\ell\log(r) + 1\big)r^{\ell-2}.$$

Each application of $\frac{1}{r}\frac{d}{dr}$ drops the degree by 2, thus, $\phi(r) = r^9$ is the lowest degree of PHS that can be used.

As noted in Section 3.2.3, the terms $\phi, \mathcal{H}_{\mathbf{x}}\phi, \mathcal{L}_{\mathbf{x}}\phi$ may not be linearly independent. Here we present two noteworthy examples: 1) on the plane, any choice of ϕ will cause $\mathcal{L}_{\mathbf{x}}\phi = 0$, and 2) on the sphere, choosing ϕ to be an odd PHS will cause $\mathcal{L}_{\mathbf{x}}\phi$ to be a

linear multiple of ϕ. The first of these is easy to see, as $\mathbf{n}$ is orthogonal to the plane, and thus $\mathbf{x}-\mathbf{y}$ is orthogonal. Thus $p = \mathbf{n}\cdot(\mathbf{x}-\mathbf{y}) = 0$ and $\mathcal{L}_{\mathbf{x}}\phi = p\phi_1(r) = 0$. This is a very important example, as stencils over smooth surfaces will appear increasingly flat under refinement. The second example is also important, as the sphere is a common manifold in applications. It is however, less obvious to see why it is the case.

Lemma A.8. If $\phi(r) = r^k$ where k is odd, then

$$\phi \propto \mathcal{L}_{\mathbf{x}}\phi$$

on any sphere.

Proof: Let $\mathbb{M}$ be a sphere of radius R and let $\mathbf{x}, \mathbf{y} \in \mathbb{M}$. Let $\phi(r) = r^k$ where k is odd. Then from Lemma A.1 we have that $\mathcal{L}_{\mathbf{x}} = pkr^{k-2}$ We will show that $p = -\frac{r^2}{2R}$ and thus $\mathcal{L}_{\mathbf{x}} = -\frac{k}{2R}\phi(r)$.

Define $L = \sqrt{r^2 - p^2}$. Note that $r^2 \geq p^2$ since $\|\mathbf{n}\| = 1$, so $p = \|\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})\| \leq r$. Also note that p must be negative, since $\mathbf{n}$ is oriented outward and $\mathbf{y}$ is on another angle of the tangent plane.

On the sphere, $\mathbf{n}$ is parallel to the line passing through the center of the sphere and $\mathbf{x}$. This gives us the configuration seen in Figure A.1. Then we have that

$$\begin{aligned} R^2 &= (R+p)^2 + L^2 \\ &= R^2 + 2Rp + p^2 + L^2 \\ 0 &= 2Rp + p^2 + L^2 \\ &= 2Rp + p^2 + r^2 - p^2 \\ &= 2Rp + r^2 \\ p &= -\frac{r^2}{2R}. \end{aligned}$$

□

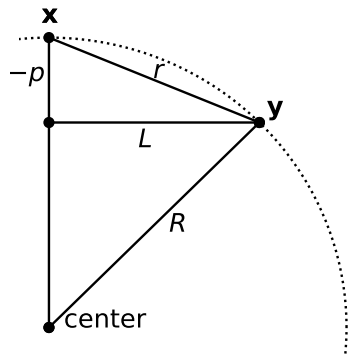


Figure A.1: A visualization of the argument used to prove Lemma A.8.