Partition of Unity Methods for Solving Partial Differential Equations on Surfaces

by

Genesis J. Islas

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2021 by the
Graduate Supervisory Committee:

Rodrigo Platte, Chair
Douglas Cochran
Malena Español
Ming-Hung Kao
Rosemary Renaut

ARIZONA STATE UNIVERSITY

December 2021

ABSTRACT

Solving partial differential equations on surfaces has many applications including modeling chemical diffusion, pattern formation, geophysics and texture mapping. This dissertation presents two techniques for solving time dependent partial differential equations on various surfaces using the partition of unity method. A novel spectral cubed sphere method that utilizes the windowed Fourier technique is presented and used for both approximating functions on spherical domains and solving partial differential equations. The spectral cubed sphere method is applied to solve the transport equation as well as the diffusion equation on the unit sphere. The second approach is a partition of unity method with local radial basis function approximations. This technique is also used to explore the effect of the node distribution as it is well known that node choice plays an important role in the accuracy and stability of an approximation. A greedy algorithm is implemented to generate good interpolation nodes using the column pivoting QR factorization. The partition of unity radial basis function method is applied to solve the diffusion equation on the sphere as well as a system of reaction-diffusion equations on multiple surfaces including the surface of a red blood cell, a torus, and the Stanford bunny. Accuracy and stability of both methods are investigated.

DEDICATION

Les dedico esta disertación a mis padres y a mi familia que con su amor y apoyo lo hicieron posible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

In this dissertation, we consider numerical solutions to partial differential equations (PDEs) defined on surfaces using two variations of the partition of unity method and pay special attention to the choice of node distribution. Solving PDEs on surfaces has applications in numerous fields including geophysics, biology, chemistry, image processing, fluid dynamics, and computer graphics [16, 8, 44, 31, 32, 45]. In chemistry, reaction-diffusion equations which take the form of semi-linear parabolic PDEs are used to model interactions between chemicals on the surface of red blood cells [44]. Computer graphics are another common application of PDEs on various surfaces [45]. Numerical solutions of PDEs are often used to create details for objects by synthesizing textures or patterns on surfaces such as stone walls, grass, etc. This dissertation pays special attention to PDEs on spherical geometries since they have important applications in geophysics. PDEs defined on the sphere are used for weather forecasting and climate modeling [10, 9, 26, 32].

There are many approaches to solving PDEs on arbitrary surfaces and a common technique is to numerically solve the PDE on a plane and then project the solution as a parametric texture onto the surface [48]. However, the parametrizations are often difficult to construct since complicated surfaces cannot be described by a single parametric function. Another common approach, presented in [45], solves the PDEs directly on the triangulated surface by discretizing the equations on a general polygonal grid with finite differences. There has also been much work done using finite element techniques, as in [13] and [14], where the Laplace-Beltrami operator is represented in terms of the tangential gradient by projecting the space function gradient onto the surface using the normal vectors. In [21], radial basis function (RBF) methods are explored using the projection of the space function gradient onto the plane tangent to the surface to compute the surface gradient similarly to the finite element methods. In [34], an RBF representation is also used but the surface gradient is computed using the orthogonal gradients method.

In Chapter 2 of this dissertation, we concentrate on solving PDEs specifically on the unit sphere. Spherical harmonics [2], radial basis functions (RBFs) [16], cubed sphere methods [37], and the double Fourier sphere [42] have all been used to compute approximations on the sphere to machine precision. In the following chapter, we introduce a novel technique that uses a partition of unity method combined with Fourier approximations and takes advantage of the convergence results obtained by spectral methods. Spectral methods consist of representing a function as a truncated series of known basis functions such as the Fourier series expansion. These methods are desirable due to their exponential convergence known as spectral accuracy. If the function is suitably smooth, convergence rates can be achieved at a rate of $O(c^N)$ for some $0 < c < 1$ [43]. There are several existing techniques which implement spectral methods on the sphere. A common approach is the use of spherical harmonics [11]. Spherical harmonic expansions are a global technique analogous to trigonometric expansions for periodic functions and are a prevalent tool for solving PDEs on the sphere [2]. Another technique for solving PDEs on the sphere is the double Fourier sphere method. This approach uses a doubled up longitude latitude grid by transforming a function defined on the sphere to one defined on a rectangular grid while still preserving the periodicity of the function in both the longitude and latitude directions. The rectangular grid provided by the double Fourier sphere method allows one to also implement spectral methods on the sphere. "Spherefun" developed in [42] implements the double Fourier sphere method combined with low rank approximations and utilizes it to solve the Poisson equation. Chapter 2 of this dissertation, uses a variation of the cubed sphere method which also allows us to implement spectral methods on the sphere by using windowed Fourier approximations. This approach takes advantage of the fast Fourier transform and uses a node distribution free of coordinate singularities.

Chapters 3 and 4 are dedicated to solving PDEs on arbitrary surfaces using a partition of unity method combined with local radial basis function approximations. RBF approximations is an appealing meshfree technique since it can provide high order convergence for smooth solutions in complex geometries. The idea behind the radial basis function partition of unity (RBF-PU) method is to solve many local problems using RBF approximations and then combine the local solutions using a partition of unity method. The efficiency of the RBF-PU method comes from solving many small problems as opposed to one large scale problem. The partition of unity method with local RBF approximations was first explored in [47] and [27] and then covered in detail in the book [15]. More recently, it has been applied to solve PDEs in various scenarios including solving convection-diffusion equations arising in financial applications in [38] and approximating ice sheet models used to model the velocity fields of ice flow in glaciers in [1]. Most recently, in [12], the RBF-PU method is used to approximate curl-free and divergence free vector fields. This dissertation focuses on using the RBF-PU method to solve PDEs on various surfaces including the unit sphere, a red blood cell, a torus, and a bunny.

Chapter 3 also investigates the role of the node point distribution in the accuracy and stability of the RBF-PU method. It is well known that “good” nodes are needed when creating an accurate and stable approximation. In the case of parameterized surfaces, it is common to use the parameterization to generate a grid for the nodes. However, this mapping distorts the node distribution and introduces issues such as artificial singularities. This can most obviously be seen when considering the longitude latitude grid on the sphere where the coordinate system introduces artificial singularities and oversampling at the poles. In this work, we utilize a greedy algorithm for finding near optimal nodes for the RBF-PU method. In Chapter 4, the radial basis function partition of unity method is adapted for larger problems. The RBF-PU method is ideal for larger problems or intricate surfaces since they both require a large number of points for accuracy and highlight the benefits of the domain decomposition implemented by this technique. The greedy algorithm used for the point node selection is improved by introducing a preselection process and reducing the size of the node selection problem by solving an analogous problem using a randomized projection.

1.1 Partition of Unity

In this dissertation, we explore two variations of the partition of unity method to numerically solve PDEs on the various surfaces. The application of the partition of unity method to solve PDEs was first presented in [30] in the 1990s. Generally, the idea for this technique is to decompose the domain into subdomains and create local approximations as shown in [15]. The global approximation is then computed by taking a weighted average of the local approximations. Formally, given an open and bounded domain $\Omega \subset \mathbb{R}^s$, we define M subdomains such that for each $j \in 1, \ldots, M$, $\Omega_j \subseteq \Omega$ and $\Omega \subseteq \bigcup_{j=1}^{M} \Omega_j$. Note that each subdomain will have some overlap with neighboring subdomains. Then a family of compactly supported, nonnegative, continuous functions, $\{w_j\}$, are chosen such that each w_j is zero outside the closure of the corresponding subdomain Ω_j and for every $x \in \Omega$ we have

$$\sum_{j=1}^{M} w_j(x) = 1.$$

Hence, the name partition of unity. Then on each subdomain, an approximation is constructed, say u_j, and the global approximation is given by

$$\tilde{u}(x) = \sum_{j=1}^{M} u_j(x) w_j(x)$$

for $x \in \Omega$. It is important to note that if each local approximation is an interpolant then the global approximation is also an interpolant. In the following chapters, two variations of the partition of unity method are implemented by using the windowed Fourier method and radial basis functions for the local approximations to solve PDEs on surfaces. Throughout this dissertation we assume the surfaces are connected.

1.2 Summary of Contributions

The main contributions of this dissertation are summarized below. Additionally, links to the codes for select experiments can be found in the appendix.

Spectral Cubed Sphere Method:

- A novel variation of the cubed sphere method for function approximation on the sphere is proposed.
- The gradient operators are defined on the sphere.
- Object oriented code for the spectral cubed sphere method is developed. The accuracy is tested using various functions.
- The method is applied to solve the diffusion and transport equations defined on the sphere.
- A global filter and a local spectral filter are implemented for stability in the time stepping scheme.
- Parameter sensitivity is investigated.

Radial Basis Function Partition of Unity Method:

As noted above, the RBF partition of unity method is not a novel technique and has been used to solve PDEs in the past such as those arising in financial applications in [38]. On the other hand, various other approaches which use RBF representations have been applied to solve PDEs on surfaces such as the finite difference method implemented in [40] and the radial basis function orthogonal gradients method presented in [34]. However, to our knowledge, the RBF partition of unity method has not been implemented to solve PDEs defined on surfaces as is done in this dissertation. In particular, our contributions are summarized as follows:

- A greedy algorithm is used to select the interpolation nodes on the surface for the partition of unity method with local RBF approximations.
- The RBF partition of unity method is implemented to solve PDEs on smooth surfaces. The surface normal vectors are used to define the differential operators on the surface used to solve the diffusion equation on the sphere and a set of reaction-diffusion equations on various surfaces.
- A time comparison of the node selection process is presented.
- Accuracy and stability of the method are investigated.

Chapter 2

SPECTRAL CUBED SPHERE METHOD

In this chapter, we explore a novel method that implements the partition of unity method described in Section 1.1 and utilizes the windowed Fourier technique for the local approximations. It is well known that spectral methods are computationally efficient with strong convergence properties under the right conditions. However, Fourier approximations require smooth functions with periodic boundary conditions for accuracy and fast convergence. The error observed in a Fourier approximation near a jump discontinuity of a discontinuous signal occurs in the form of a ringing artifact known as the Gibbs phenomenon. For nonperiodic functions, the end of the intervals are considered a jump discontinuity and oscillations are observed in the approximation. In order to avoid this, the windowed Fourier method is used to create Fourier approximations for nonperiodic functions.

2.1 Windowed Fourier Method

The windowed Fourier method is an attractive technique used to avoid the Gibbs phenomenon encountered in Fourier approximations of nonperiodic functions and is investigated in detail in [35] and [36]. The general idea is to simulate periodic boundary conditions by multiplying the signal by a window function and computing a Fourier approximation on the product to take advantage of the spectral properties inherited from the periodic boundary conditions. For simplicity, the windowed Fourier method is described here in one dimension. We begin by establishing some properties of the discrete Fourier transform.

Given a set of equally spaced points, $\{x_j\}_{j=0}^{N} \subset [-\pi, \pi]$, the standard Fourier approximation is defined as

$$F_N[u](x) = \sum_{n=-N/2}^{N/2} \hat{u}_n \exp(inx).$$

Here, the coefficients $\hat{u}_n$ are computed by

$$\hat{u} = \mathcal{F}\mathbf{u}, \text{ with } \hat{u} = \begin{bmatrix} \hat{u}_0 & \dots & \hat{u}_{N/2} & \hat{u}_{-N/2} & \dots & \hat{u}_{-1} \end{bmatrix}^T \tag{2.1}$$

where $\mathbf{u}$ is the function evaluated at the points $\{x_j\}_{j=0}^{N}$ and $\mathcal{F}$ is the Fourier transform matrix,

$$\mathcal{F}_{k,j} = \frac{e^{-i(k-1)x_{(j-1)}}}{N}, \qquad 1 \leq j \leq N+1,\ 1 \leq k \leq N+1.$$

The fast Fourier transform (FFT) can be used to efficiently compute the coefficients in (2.1) by reducing the complexity from $O(N^2)$ to $O(NlogN)$. The following theorems establish the convergence properties of the truncated Fourier series [24].

Theorem 2.1.1 *Suppose $f \in C^{(q-1)}[-\pi, \pi]$, f and its $q-1$ derivatives are 2π-periodic, and $f^{(q)} \in L^2[-\pi, \pi]$ with F_N defined as above, then*

$$||f - F_N[f]||_2 \leq cN^{-q},$$

where c is a positive constant which depends on q but not on N.

If the function is analytic then the truncated Fourier series converges even faster and exponential convergence is observed.

Theorem 2.1.2 *Suppose f is analytic and 2π-periodic, then*

$$||f - F_N(f)||_2 \leq Ke^{-\alpha N}$$

where $K, \alpha > 0$.

We now use these properties to formally define the windowed Fourier approximation of a function. Given a continuous and not necessarily periodic function u on a bounded domain $[-\pi, \pi]$, we consider the product uw where w is a smooth window function with values and derivatives near zero on the boundaries. Since the function uw is a continuous periodic function then it can be approximated using a truncated Fourier series denoted by $F_N[uw](x)$. The approximation for u is then constructed by dividing by the known function w where w is defined such that $w(x)$ is not zero for $x \in [-\pi, \pi]$, so

$$u(x) \approx \frac{F_N[uw](x)}{w(x)}.$$

We note then that the accuracy of the approximation of u depends on the accuracy of the Fourier approximation of uw.

We now illustrate this method with an example. We consider the non-periodic function $u(x) = \exp(x/\pi) + \exp(-50x^2)$ shown in Figure 2.1. The window function used is a super-Gaussian function defined as $w(x) = \exp(-\alpha(x^{2\lambda})))$ for $x \in [-\pi, \pi]$ with $\lambda = 6$ and $\alpha = 52 \ln 2$. The windowed Fourier approximation for u using 200 points is also shown in Figure 2.1. It should be noted that since the Fourier approximation is divided by the window function which is close to zero near the boundaries, the approximation of u becomes inaccurate in those areas. However, we recall that the windowed Fourier approximation will be utilized in conjunction with the partition of unity method which inherently contains overlapping windows. The windows are chosen large enough so that the inaccuracies near the edges will fall in the overlap regions and the averaging of the multiple approximations remedies this issue. In contrast, the Fourier approximation of just the function u using 200 points can be seen in Figure 2.1. We note that the oscillations due to the Gibbs phenomenon are observed throughout the approximation.

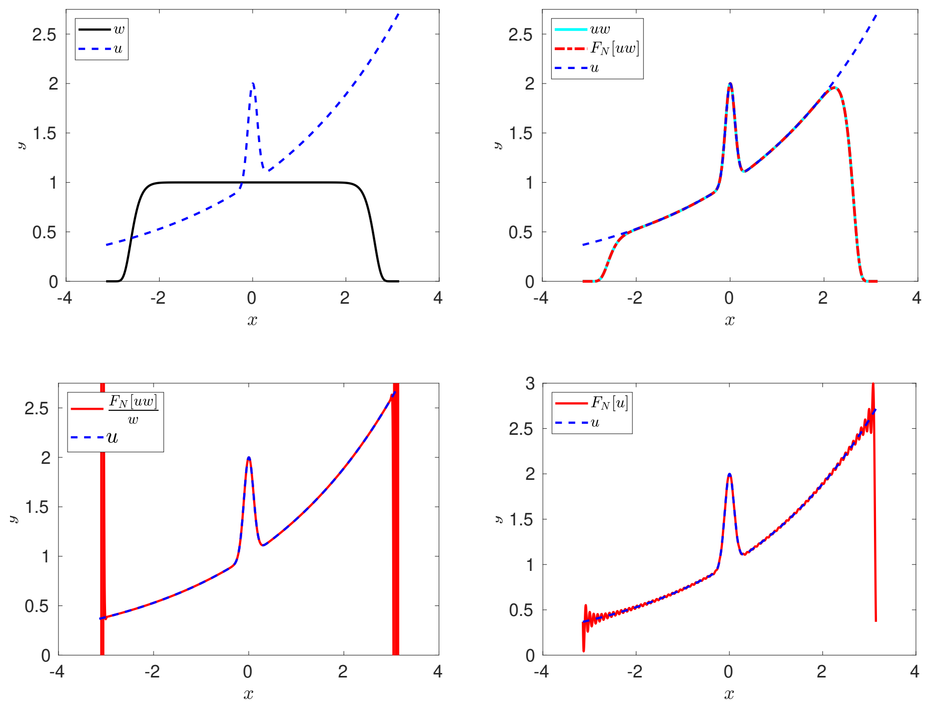

Figure 2.1: Top Left: The Function $u(x) = \exp(x/\pi) + \exp(-50x^2)$ and the Window Function $w(x) = \exp(-\alpha(x/\pi)2\lambda)$. Top Right: The Function uw, the Fourier Approximation of uw, and the Function u. Bottom Left: The Windowed Fourier Approximation of u and the Function u. Bottom Right: The Fourier Approximation of u and the Function u.

2.2 Spectral Cubed Sphere Method

In this section, we introduce the spectral cubed sphere method which combines the partition of unity method with windowed Fourier approximations. This approach was inspired by the cubed sphere method first introduced in [37] which presented a new gridding technique for solving PDEs on the unit sphere using finite differences. The cubed sphere method partitions the unit sphere into six identical regions by projecting the sides of a cube onto an inscribed sphere as shown in Figure 2.2. This technique considers the sphere as six coupled regions and derives the node point distribution by projecting a grid of uniformly spaced points on each face of the cube onto the sphere. We consider a similar gridding technique for our spectral cubed

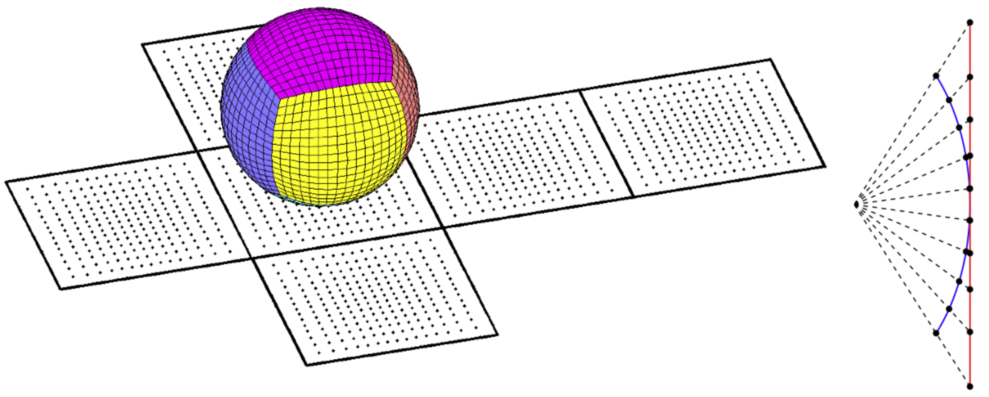

Figure 2.2: The Unit Sphere Partitioned into Six Similar Regions and the Corresponding Decomposed Cube. The Equally Spaced Nodes on the Face of the Cube are also Shown with the Corresponding Grid Formed on the Sphere. The Image on the Right is a Set of Nodes from a Cross Section of the Cube and the Corresponding Nodes Projected on the Sphere.

sphere method. The main difference with our approach is that we utilize overlapping regions instead of an exact partition and create the global approximation using the partition of unity method. Thus our method uses a node point distribution generated by six uniform square grids on planes tangent to the sphere mapped onto the surface. The motivation for this technique is to have a uniform square grid on each subdomain to compute Fourier approximations for the local approximations in the partition of unity method. Figure 2.3 demonstrates an example of one of these subdomains on the sphere. In addition, Figure 2.3 also shows all six regions in order to demonstrate the overlap between the subdomains and the node distributions.

We now define the map necessary to project points onto the sphere from the tangent planes and vice versa. A simple map that takes any point on the surface of the unit cube onto the unit sphere is obtained by dividing it by the magnitude of the vector from the origin to the point. Hence, we define the map $R : \mathbb{R}^3/\{0\} \to \mathbb{R}^3$ as $R(x, y, z) \to (\frac{x}{d}, \frac{y}{d}, \frac{z}{d})$, where $d = \sqrt{x^2 + y^2 + z^2}$. The mapping from the sphere to the cube is constructed by inverting the previously defined map, R. Let (x_s, y_s, z_s) be an

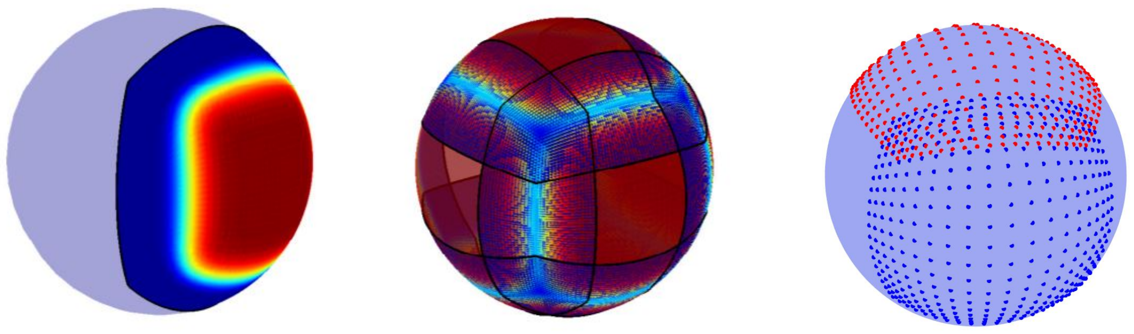

Figure 2.3: Left: An Example of the Window Function Defined on One of the Subdomains. Center: The Overlap of the Window Functions Defined on the Six Subdomains. Right: Node Distribution for Two Overlapping Subdomains.

arbitrary point on the sphere and (x_c, y_c, z_c) be the corresponding point on the cube. Then $R(x_c, y_c, z_c) = (x_s, y_s, z_s)$ and the following set of equations can be derived from the definition of the map R:

$$x_s = \frac{x_c}{d}, \qquad y_s = \frac{y_c}{d}, \qquad z_s = \frac{z_c}{d}. \tag{2.2}$$

The inverse mapping is defined separately for each region of the sphere. The first map will be defined for the region on the sphere corresponding to the face of the cube where $x_c = 1$ for all points. Using the fact that $x_c = 1$, we can solve for the coordinates on the cube using (2.2). Taking the first equation and solving for d we get,

$$d = \frac{1}{x_s}.$$

Now d can be replaced in (2.2) to get,

$$x_c = 1, \qquad y_c = \frac{y_s}{x_s}, \qquad z_c = \frac{z_s}{x_s}.$$

Hence, the inverse mapping for the first region is defined by $R_1^{-1} : \mathbb{R}^3 \to \mathbb{R}^3$ as $R_1^{-1}(x, y, z) \to (1, \frac{y}{x}, \frac{z}{x})$. We define the maps $R_2^{-1}, ..., R_6^{-1}$ in a similar fashion so that the constant coordinate on the cube can be used to solve for the mapping. Hence, each map takes the appropriate region of the sphere and projects it onto the face of the cube and they are defined as:

$$\begin{aligned} R_1^{-1}(x,y,z) &\rightarrow \left(1, \frac{y}{x}, \frac{z}{x}\right), & R_2^{-1}(x,y,z) &\rightarrow \left(\frac{x}{y}, 1, \frac{z}{y}\right), \\ R_3^{-1}(x,y,z) &\rightarrow \left(-1, \frac{-y}{x}, \frac{-z}{x}\right), & R_4^{-1}(x,y,z) &\rightarrow \left(\frac{-x}{y}, -1, \frac{-z}{y}\right), \\ R_5^{-1}(x,y,z) &\rightarrow \left(\frac{x}{z}, \frac{y}{z}, 1\right), & R_6^{-1}(x,y,z) &\rightarrow \left(\frac{-x}{z}, \frac{-y}{z}, -1\right). \end{aligned}$$

We now define the windowed Fourier method in two dimensions in order to implement the partition of unity method described in Section 1.1. Since the sphere is decomposed into six different subdomains, a function defined on the sphere is now restricted to each of these regions and is no longer periodic. Thus the windowed Fourier method described in Section 2.1 is an attractive technique which allows us to create Fourier approximations while preserving the spectral properties that come with the periodic boundary conditions. Similarly to the one dimensional case and as in [36], we use a super-Gaussian as the window function, however, there are a variety of window functions which can be used as explored in [7]. We then define the window function as

$$w(t,p) = \exp(-\alpha(t^{2\lambda} + p^{2\lambda})),$$

for $(t,p) \in [-d,d] \times [-d,d]$ where λ is a positive integer. Here, α is chosen such that the window function near the boundaries is approximately zero relative to 1. In our implementation, we use double precision (IEEE 754 standard) and choose α such that $w(\pm d, \cdot) \approx w(\cdot, \pm d) \approx 2^{-52}$. The parameter d determines the size of the window functions and the width of the overlapping regions and we note that three regions overlap in some areas of the sphere as seen in Figure 2.3. Although, larger windows imply having larger overlap regions which give a more accurate global approximation, this is also more computationally expensive. Also, note that a larger choice of λ for the window function results in a wider window implying more support for the Fourier approximation. However, a larger value of λ also produces stiffer gradients near the boundaries which means creating an accurate approximation is harder. This trade-off in the choice of d and λ is further explored in Section 2.3. Given the window function, we now consider its product with a function u and note that the result is a periodic function. The function uw is then approximated using the truncated double Fourier series defined by

$$F_N[u](x,y) = \sum_{m=-N/2}^{N/2} \sum_{n=-N/2}^{N/2} \hat{u}_{m,n} \exp\left(\frac{i\pi nx}{d} + \frac{i\pi my}{d}\right), \tag{2.3}$$

with $(x,y) \in [-d,d] \times [-d,d]$. The two dimensional fast Fourier transform can be used to efficiently compute the coefficients. Then the Fourier approximation of the product uw is given by $F_N[uw](t,p)$ and the local approximation for u is given by

$$u(t,p) \approx \frac{F_N[uw](t,p)}{w(t,p)},$$

as was done in the one dimensional case in Section 2.1. However, this is the approximation for only one region and the partition of unity method is used to compute the global approximation defined as

$$u(t,p) = \sum_{j\in I} \frac{F_N[uw_j](t,p)}{\sum_{j\in I} w_j(t,p)}.$$

Here, w_j for $j \in I = \{1,\dots,6\}$ are the super-Gaussian window functions defined to be nonzero on the corresponding regions of the sphere. We recall that the distribution of node points on the sphere is constructed by projecting six uniform grids from tangent planes. Therefore, the nodes projected from two neighboring subdomains will not perfectly align on the sphere as seen in Figure 2.3. Thus, when creating the weighted average, the points which lie in the overlap regions must be evaluated in the Fourier approximations of the neighboring subdomains. While the FFT is an efficient technique to compute the coefficients of the Fourier approximations, the non-uniform fast Fourier transform (NUFFT) is an attractive technique that can be used to efficiently evaluate points in said Fourier approximations. In our method, we compute the evaluations needed to construct the weighted average by applying a NUFFT package developed in [4].

2.3 Parameter Choice and Convergence

In this section, we first consider the parameters in the definition of the window function. We restate the super-Gaussian window function used in this chapter for convenience, $w(x, y) = \exp(-\alpha \left((x/d)^{2\lambda} + (y/d)^{2\lambda}\right))$. Again, α is chosen such that w on the boundary is machine zero or $w(\pm d, \cdot) \approx 0 \approx w(\cdot, \pm d)$. We recall that the approximation for uw is defined on a square grid where the parameter d dictates the size of that square. Thus the magnitude of d determines the size of the overlap regions and is chosen so that $w(\cdot, \pm 1) = w(\pm 1, \cdot) = 0.5$. The motivation for this choice being that when two windows intersect, the height of each window function is a half and both windows have equal weight.

The parameter λ dictates the rate at which the window function reduces to zero on the boundary. A larger λ produces a wider window meaning that the region where the window is equal to one is larger and produces more support for the Fourier approximation. However, choosing a larger value for λ also produces a stiffer gradient near the boundaries which means creating an accurate approximation is harder and requires more points. The effect of λ is investigated in detail in [36] where it is concluded that at least 24λ points are need to approximate a smooth function in one dimension with accuracy of 10^{-12}. We demonstrate how varying λ affects accuracy in the following example. We consider the function $u(x) = \sin(50x)$. A windowed

Fourier approximation, $\tilde{u}$, is constructed using a super Gaussian window function with multiple values of λ and N. The error, $||u - \tilde{u}||_2$, over 500 points is shown in Figure 2.4 where it can be seen that the optimal λ is larger than the lower bound defined by $N/24$ for each N.

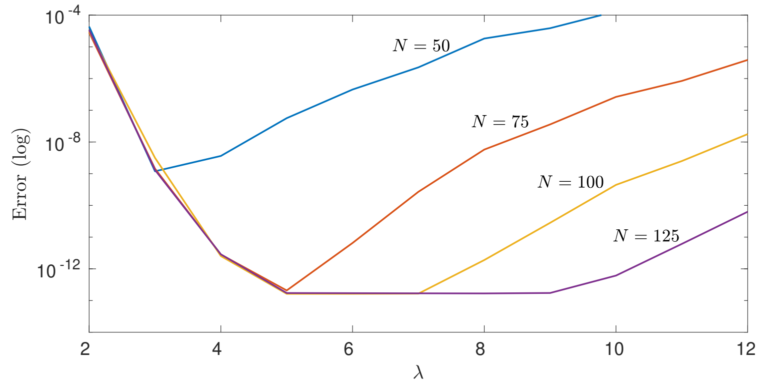

Figure 2.4: Error for the Windowed Fourier Approximation for the Function $u(x) = \sin(50x)$ for $\lambda = 2, 4, \ldots, 12$ and $N = 50, 75, 100, 125$.

Using the parameters chosen above, we consider the convergence of the spectral cubed sphere method using the following four functions: $u_1(x, y, z) = \sin(100xyz)$, $u_2(x, y, z) = 1/(1 + 100(x^2 + y^2))$, $u_3(x, y, z) = \cos(xz - \sin(y))$, and $u_4(x, y, z) = \sqrt{(1.05 + xyz)}$. Figure 2.5 and Figure 2.6 show each of the functions and Figure 2.7 shows the error of the approximation, $||u - \tilde{u}||_\infty$, evaluated over 500 points. We note the geometric convergence in all cases.

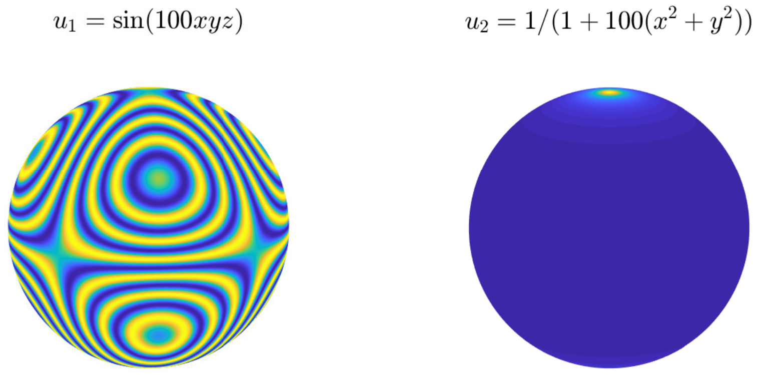

Figure 2.5: The Functions u_1 and u_2 on the Sphere.

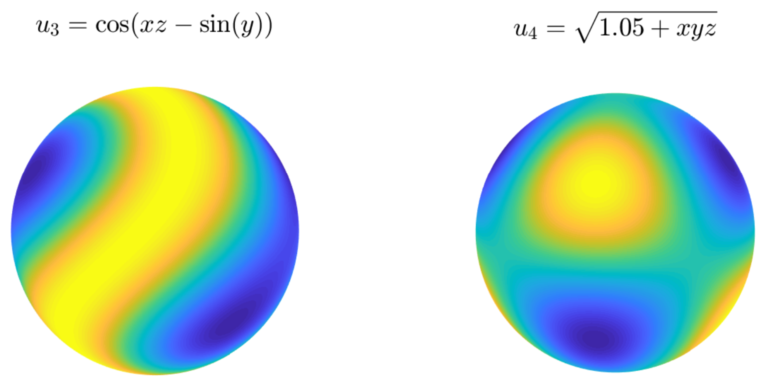

Figure 2.6: The Functions u_3 and u_4 on the Sphere.

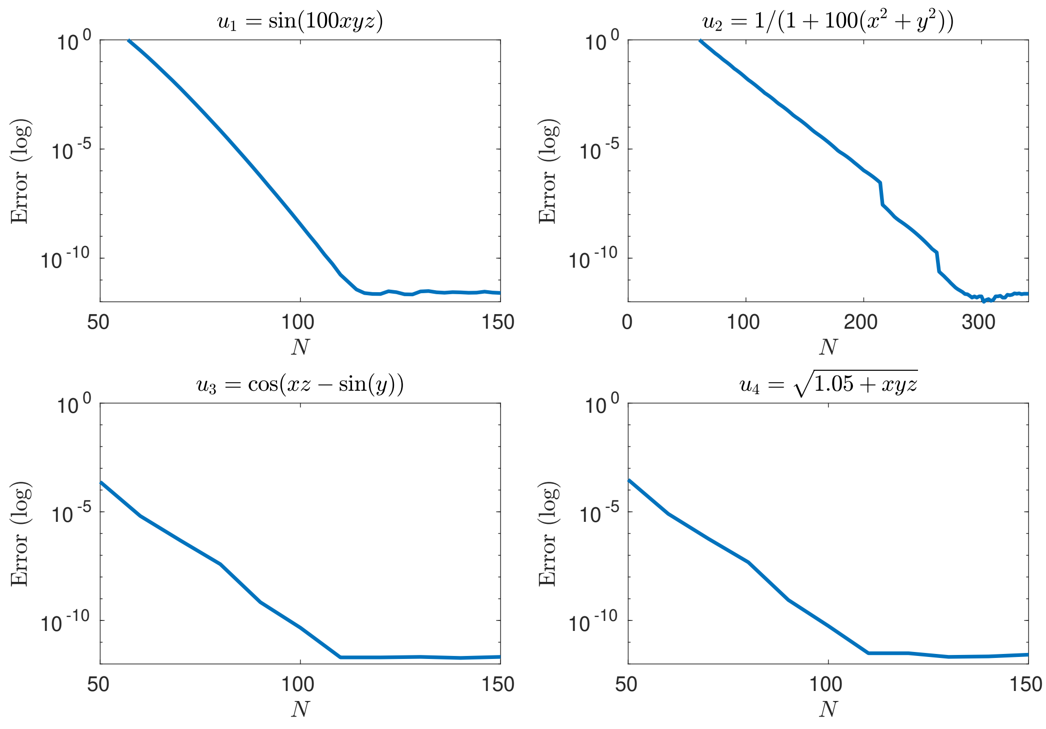

Figure 2.7: Error for the Spectral Cubed Sphere Method Approximation of u_1, u_2, u_3, and u_4 for Various Values of N.

2.4 Gradient Operators

Since the main motivation for this method is solving PDEs, in this section we present the surface differentiation operators. The surface gradient is defined by computing the gradient on the equally spaced grid on the faces of the cube and then using the mapping defined in Section 2.2 to project it onto the sphere. We recall that the approximation for the function uw is given by the truncated Fourier series, $F_N[uw](t,p)$. Thus it is simple to compute the partial derivative with respect to t and is defined by

$$(F_N[uw](t,p))_t = \sum_{m=-N/2}^{N/2} \sum_{n=-N/2}^{N/2} in\, \widehat{uw}_{m,n} \exp\left(\frac{i\pi nx}{d} + \frac{i\pi my}{d}\right),$$

for $(t,p) \in [-d,d]\times[-d,d]$. The partial with respect to p is defined similarly. However, a change of variables is necessary to obtain the partial derivatives of uw with respect to the Cartesian coordinates x, y, and z on the sphere. For simplicity in notation, we define $f = uw$ and use the chain rule to find a relationship between the partials,

$$f_t = f_x x_t + f_y y_t + f_z z_t \tag{2.4}$$

$$f_p = f_x x_p + f_y y_p + f_z z_p. \tag{2.5}$$

We recall the mapping from the cube to the sphere, R, defined in Section 2.2 and reduce the domain to two variables since one of the variables is constant and equal to 1 on each face. By properties of the surface gradient, ∇f, it is tangent to the surface and since R_t and R_p are linearly independent vectors parallel to the surface we can relate them as

$$\nabla f = \alpha R_t + \beta R_p. \tag{2.6}$$

Then, by combining (2.4), (2.5), and (2.6) we have that

$$\begin{pmatrix} f_t \\ f_p \end{pmatrix} = \begin{pmatrix} x_t & y_t & z_t \\ x_p & y_p & z_p \end{pmatrix} \nabla f = \begin{pmatrix} R_t^T \\ R_p^T \end{pmatrix} (\alpha R_t + \beta R_p) .$$

Letting $E = R_t^T R_t$, $F = R_t^T R_p$, and $G = R_p^T R_p$ we get

$$\begin{pmatrix} f_t \\ f_p \end{pmatrix} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} .$$

Thus, we define the surface gradient as $\nabla f = \alpha R_t + \beta R_p$ with

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{EG - F^2} \begin{pmatrix} G & -F \\ -F & E \end{pmatrix} \begin{pmatrix} f_t \\ f_p \end{pmatrix} .$$

Note that we can rewrite $EG-F^2 = |R_t \times R_p|^2 > 0$ by Lagrange's identity from Linear Algebra and thus $\frac{1}{EG-F^2}$ is always defined. In differential geometry, E, F, and G are known as the "coefficients of the first fundamental form" [22]. For computational efficiency, we go a step further and rewrite this as

$$\nabla f = \frac{1}{EG - F^2} \left[(GR_t - FR_p) f_t + (ER_p - FR_t) f_p \right] .$$

It is important to note, that the coefficients of f_t and f_p do not depend on the function values and thus when considering iterative methods, they only need to be computed once. We recall that we defined $f = uw$ and use the product rule applied to uw to solve for u_x, so we obtain

$$[uw]_x = u_x w + u w_x,$$
$$u_x = \frac{[uw]_x - uwx}{w}.$$

The partials with respect to y and z can be computed in a similar fashion. We note the importance of the u in the second term since it is the global approximation meaning that it communicates information from all six regions. Once the local approximations for the surface gradient have been obtained, we use the partition of unity formulation to compute the global approximation,

$$u_x = \sum_{j \in I} \frac{[uw_j]_x - u[w_j]_x}{\sum_{j \in I} w_j}.$$

2.4.1 Comparison with Radial Basis Function

In this section, we study the accuracy of the Laplacian operator of the spectral cubed sphere method and compare to a global radial basis function (RBF) collocation method. The algorithm used for the RBF approximation is studied in detail in [28]. We consider the function $u(x, y, z) = \sin(100xyz)$ and its Laplacian $\Delta u(x, y, z)$ and approximate them using both the spectral cubed sphere method (SCSM) and the global RBFs. Figure 2.8 shows the relative error for various values of the shape parameter, ϵ, for the global RBF approximation. It is well known that high order accuracy in global RBF approximations is often hard to achieve due to poor conditioning of the interpolation matrix. As expected, the error converges faster than our method for a small number of points, however, it then hits a minimum and begins to increase never reaching machine precision. We note that the y-axis is bound to show the comparison with the RBF approximation and if we zoomed out we would see that machine precision is reached around 9×10^4 total points as shown in Figure 2.7.

2.5 Numerical Examples

In this section, we consider various time dependent problems in order to study stability and accuracy. Note that for this method there are no boundary conditions on the sphere. The method of lines is used for the following problems with the spectral cubed sphere method approximating the spacial component combined with an explicit time stepping scheme.

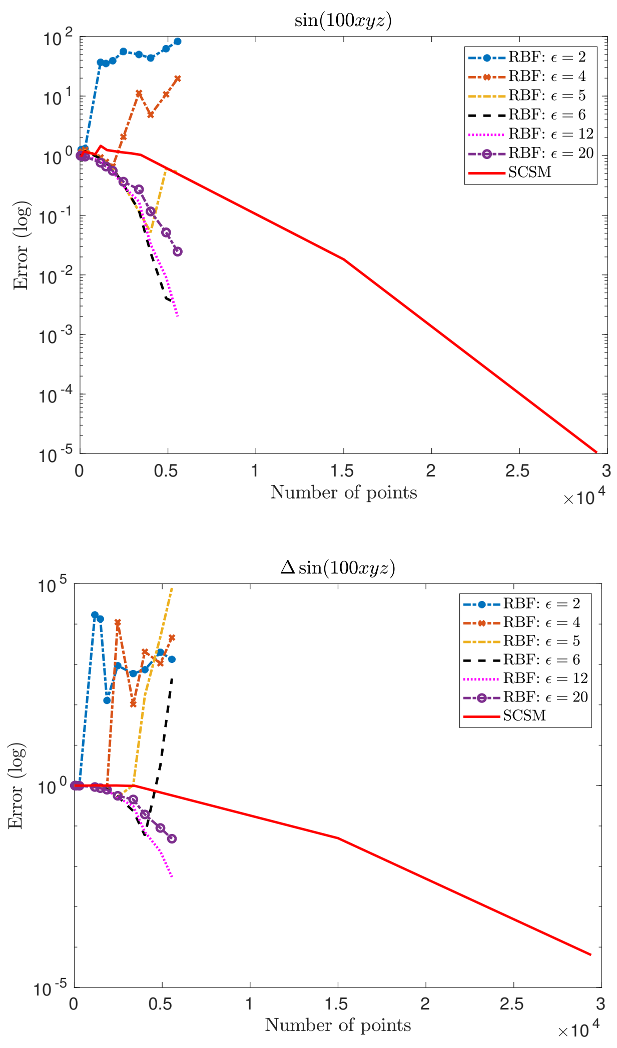

Figure 2.8: Relative Error for Approximations of $\sin(100xyz)$ and $\Delta \sin(100xyz)$ Using the Spectral Cubed Sphere Method and a Global RBF Collocation Method with Various Values of the Shape Parameter ϵ.

2.5.1 Filters

We introduce two filters that are used for stability in the time stepping schemes in some of the following problems. A common approach for filtering is to do so in spectral space so one of the filters used is a low pass filter applied to each of the subdomains individually. A key benefit of this filter is the ease of implementation.

To apply this filter, (2.3) is replaced by,

$$F_N[u](x,y) = \sum_{m=-N/2}^{N/2} \sum_{n=-N/2}^{N/2} \sigma\left(2n/N\right) \sigma\left(2m/N\right) \hat{u}_{m,n} \exp\left(\frac{i\pi nx}{d} + \frac{i\pi my}{d}\right),$$

where σ is a filter function. We consider the exponential filter function defined in [24],

$$\sigma(k) = \begin{cases} 1 & |k| \leq \eta \\ \exp(-\alpha(\frac{k-\eta}{1-\eta})^{\rho}) & |k| > \eta \end{cases}.$$

Here, α is chosen such that $\sigma(1)$ is machine zero and the choice for the parameters η and ρ is fully explored in the following section. Although this filter is simple to implement, the lack of filtering on the global approximation is a drawback since it is applied only on the subdomains. The second filter we consider is adding artificial viscosity globally to the PDE formulation. This filter essentially "smooths out" the solution and mitigates instabilities introduced when patching the local approximations together with the weighted average. Given the general formulation of a time dependent PDE:

$$u_t = f,$$

where f here is a function of u and its derivatives, we consider the PDE with added viscosity:

$$u_t = f + \mu \Delta u,$$

where μ is small in magnitude.

2.5.2 Transport Problem

In this section, we consider the following transport problem on the sphere,

$$u_t = \overrightarrow{v} \cdot \nabla u,$$
$$u(0, x, y, z) = \exp(-1((x-1)^2 + y^2 + z^2)),$$

where $\overrightarrow{v}$ is a velocity field for a constant rotation around the z-axis. Hence, the solution is equal to the initial condition at each full rotation. For stability, we apply the second filter defined above and add numerical diffusion and instead solve the following problem

$$u_t = \overrightarrow{v} \cdot \nabla u + \mu \Delta u,$$
$$u(0, x, y, z) = \exp(-1((x-1)^2 + y^2 + z^2)),$$

with $\mu = 10^{-4}$. The spectral cubed sphere method is used with the parameters $\eta = 0.4$ and $p = 4$. The fourth order Runge-Kutta method is used for the time stepping scheme with a time step of $dt = 10^{-4}$. Figure 2.9 displays the initial condition and the solution u after 4 full rotations or when $t = 8\pi$. The solution appears as we

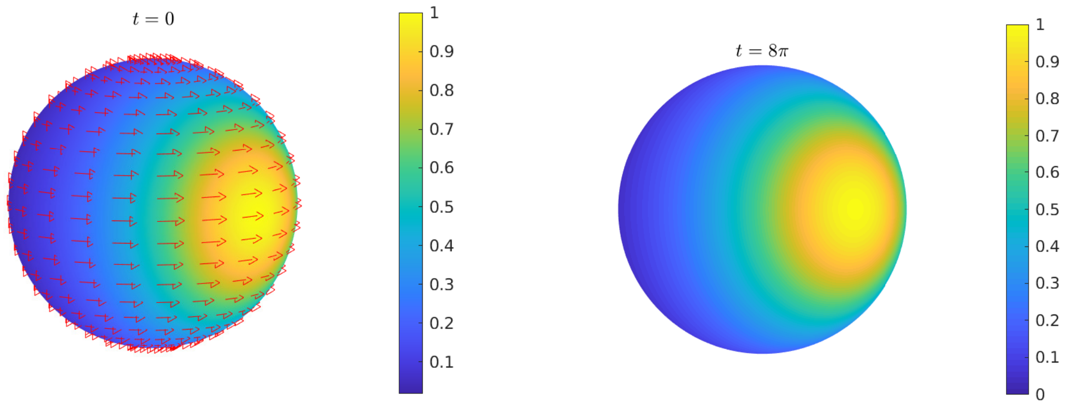

Figure 2.9: Initial Condition and Numerical Solution for the Transport Equation at $t = 8\pi$ (4 Full Rotations) with Time Step $dt = 10^{-4}$.

expected. However, the effect produced by adding the diffusion term is visible in the magnitude of the peak which is 1 at $t = 0$ and about 0.990 when $t = 8\pi$.

We now consider some convergence properties and the sensitivity of the parameters. Figure 2.10 shows the geometric convergence of the largest dt sufficient for stability of one full rotation for various values of N. Figures 2.11, 2.12, and 2.13

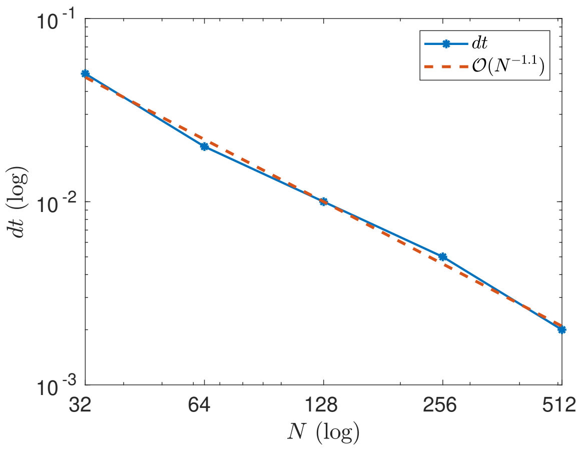

Figure 2.10: Convergence of the Largest dt Sufficient for Stability of One Full Rotation of the Numerical Solution for the Transport Equation in Terms of N.

explore the effects of the filter and the choice of parameters on the stability of the transport problem. Each figure shows two images: a map of the parameter values for which the transport problem is stable for a full rotation ($t = 2\pi$) shown in red and the corresponding error. Note that the error is only shown for the values that do not diverge as the diverging spaces are left white. Figure 2.11 considers the parameters dt and η and show the largest value of t before hitting our criteria for divergence. The values of η vary from 0.1 to 1 where $\eta = 1$ equates to no filtering at all and the smaller the values of η implies more filtering. The values of dt range from 0.002 to 0.01 as dt larger than 0.01 results in instability regardless of the filter. It is seen that as dt gets larger, more filtering is needed for stability. However, it should be noted that filtering too much results in a loss of accuracy as can be seen in the error plot.

Figure 2.12 shows the effects of the parameters dt and ρ. The value of ρ ranges from 1 to 10 and smaller values of ρ correspond to more filtering. Similar results as in Figure 2.11 are found since the solution is unstable for dt larger than 0.01 and larger dt require more filtering. Comparing the error for the two images implies that there is more sensitivity in the η parameter as the error plot for ρ is more consistent across all the different values. The last figure, Figure 2.13 shows the effect of η and ρ together. In this case, $dt = 0.01$ and the results agree with the previous two experiments as we see that more filtering corresponding to the left top corner ensures stability at the cost of accuracy.

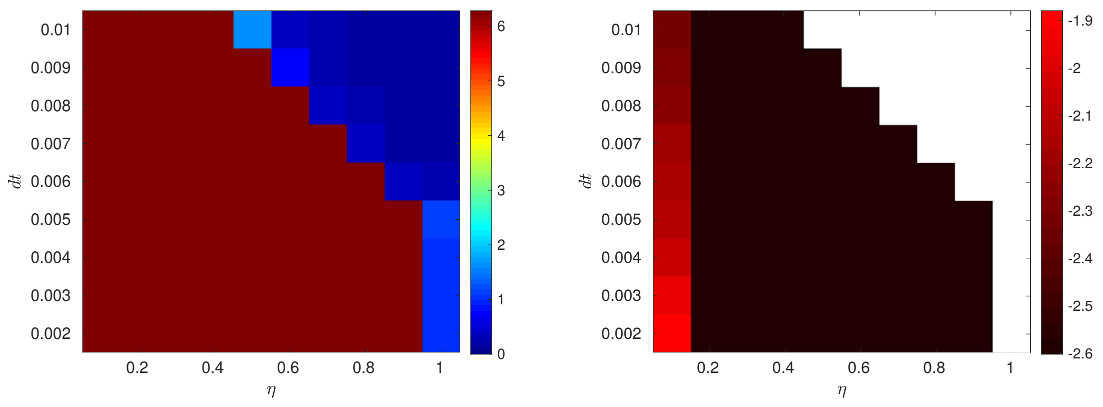

Figure 2.11: Left: Parameter Values for Which the Numerical Solution for the Transport Problem is Stable for a Full Rotation ($t = 2\pi$) Indicated by the Red Squares when dt and η are Varied. The Rest of the Colors Imply the Value of t Between 0 and 2π When the Solution Meets Our Divergence Criteria. Right: The Error for the Numerical Solution After One Full Rotation. If the Solution Diverges the Error is Not Shown and is Left White.

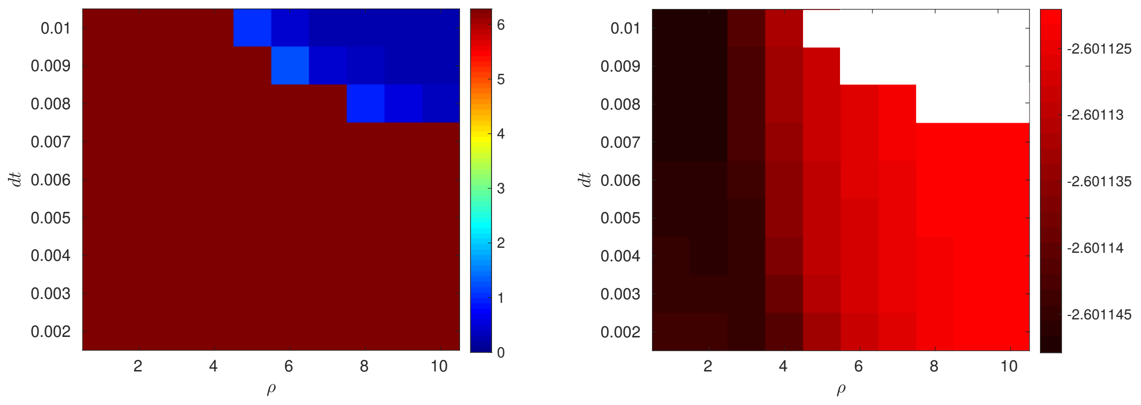

Figure 2.12: Left: Parameter Values for Which the Numerical Solution for the Transport Problem is Stable for a Full Rotation ($t = 2\pi$) Indicated by the Red Squares When dt and ρ are Varied. The Rest of the Colors Imply the Value of t Between 0 and 2π When the Solution Meets Our Divergence Criteria. Right: The Error for the Numerical Solution After One Full Rotation. If the Solution Diverges the Error is Not Shown and is Left White.

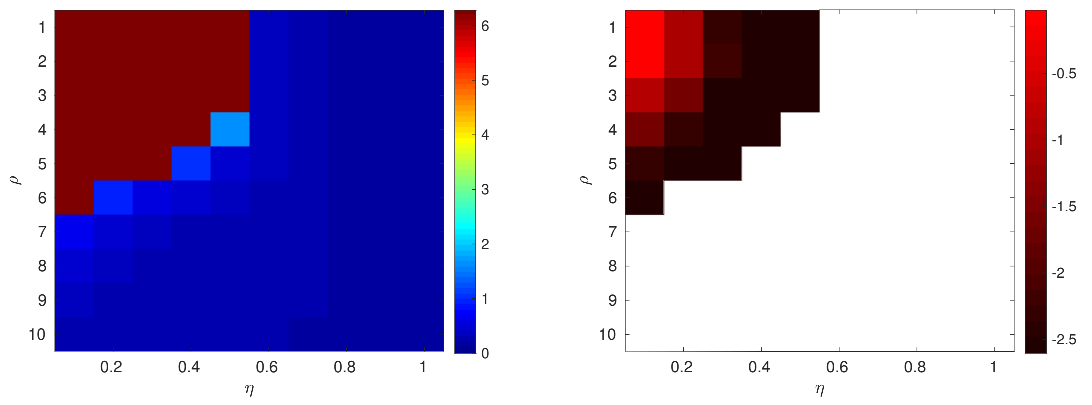

Figure 2.13: Left: Parameter Values for Which the Numerical Solution for the Transport Problem is Stable for a Full Rotation ($t = 2\pi$) Indicated by the Red Squares when η and ρ are Varied. The Rest of the Colors Imply the Value of t Between 0 and 2π When the Solution Meets Our Divergence Criteria. Right: The Error for the Numerical Solution After One Full Rotation. If the Solution Diverges the Error is Not Shown and is Left White.

2.5.3 *Diffusion Equation*

In this section, we consider the diffusion equation on the sphere:

$$u_t = \Delta u,$$

$$u(0, x, y, z) = xy.$$

The initial condition is chosen to be a spherical harmonic as that allows us to find the solution analytically which is given by

$$u(t, x, y, z) = e^{-6t}xy.$$

Figure 2.14 shows the initial condition and the solution at $t = 1.5$ using a forward Euler time stepping scheme with a time step of $dt = 10^{-6}$. The magnitude of the error at $t = 1.5$ is about $||u - \tilde{u}||_\infty = 10^{-7}$.

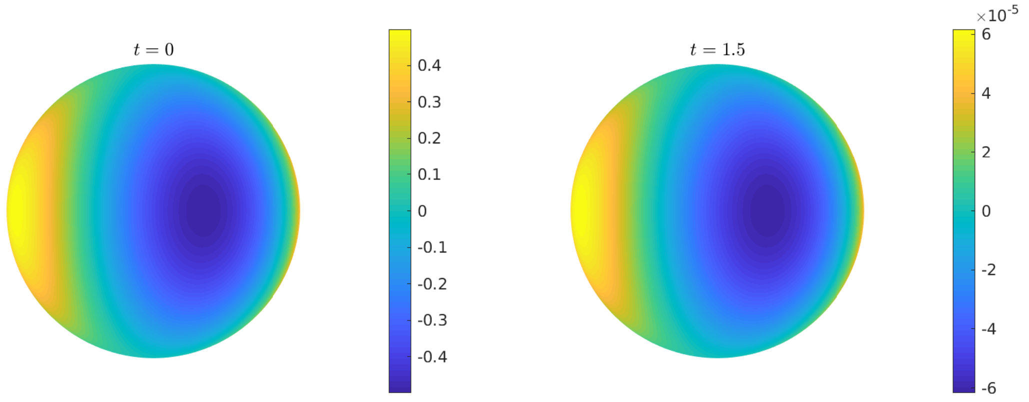

Figure 2.14: Initial Condition and Numerical Solution for the Diffusion Equation at $t = 1.5$ with Time Step $dt = 10^{-6}$.

As in the previous problem, we consider some convergence properties. Figure 2.15 shows the convergence of the largest dt sufficient for stability to reach $t = 0.4$ where the magnitude of the solution is 10% of the initial condition. As with the transport equation, we investigated the sensitivity of the parameters and found similar results that are therefore are omitted.

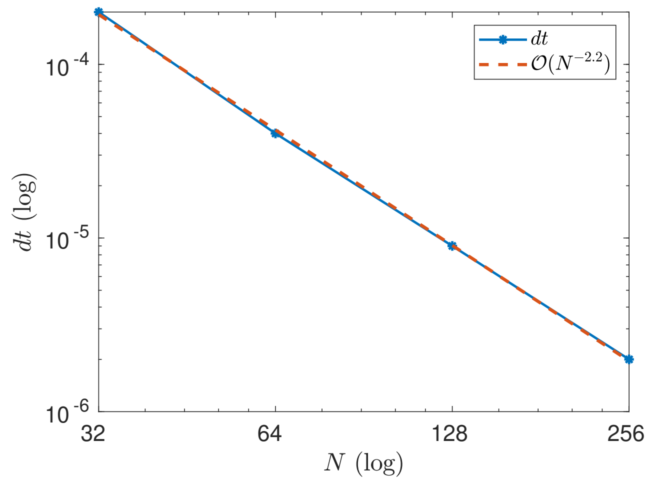

Figure 2.15: Convergence of the Largest dt Sufficient for Stability to Reach $t = 0.4$ Where the Magnitude of the Solution is 10% of the Initial Condition.

Chapter 3

RADIAL BASIS FUNCTION - PARTITION OF UNITY

In this chapter, we explore using radial basis functions (RBFs) for the local approximations in the partition of unity method described in Section 1.1. We recall that the global approximation is given by a weighted average of the local approximations on each subdomain. In this chapter, the point distribution does not have an underlying grid and thus RBFs are an attractive choice since they are a meshfree method and work well with scattered nodes. It is also well known that RBFs provide spectral like convergence for smooth solutions approximated with infinitely smooth RBFs. In this chapter we adhere to the variable notation in the following Table 3.1.

M	Number of subdomains
$\mathbf{k}_i$, $i \in \{1 \dots M\}$	Subdomain centers
w_i, $i \in \{1 \dots M\}$	Weight functions for PU method
N	Number of interpolation nodes
$\mathbf{x}_j$, $j \in \{1 \dots N\}$	RBF centers
ϕ_j, $j \in \{1 \dots N\}$	RBFs

Table 3.1: Variable Notation for the RBF Partition of Unity Method.

3.1 Radial Basis Function Interpolation

In this section, we introduce radial basis function interpolation and define the surface differential operators. Given a set of scattered nodes $\{\mathbf{x}_j\}_{j=0}^{N}$ the basic RBF interpolant takes the form

$$\tilde{u}(\mathbf{x}) = \sum_{j=0}^{N} c_j \phi(||\mathbf{x} - \mathbf{x}_j||) \tag{3.1}$$

where ϕ is a radial kernel and $||\cdot||$ is the Euclidean norm. The radial kernel is defined as a radially symmetric function and some common choices are presented in Table 3.2. Note that since the interpolant only depends on $\mathbf{x}$ and $\mathbf{x}_j$ inside the norm then changing dimensions is trivial. Throughout this dissertation, we consider the case were the number of basis functions equals the number of interpolation nodes. Thus solving for the coefficients of the interpolant is equivalent to solving the following system of equations:

$$\begin{bmatrix} \phi(||\mathbf{x}_0 - \mathbf{x}_0||) & \phi(||\mathbf{x}_0 - \mathbf{x}_1||) & \dots & \phi(||\mathbf{x}_0 - \mathbf{x}_N||) \\ \phi(||\mathbf{x}_1 - \mathbf{x}_0||) & \phi(||\mathbf{x}_1 - \mathbf{x}_1||) & \dots & \phi(||\mathbf{x}_1 - \mathbf{x}_N||) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(||\mathbf{x}_N - \mathbf{x}_0||) & \phi(||\mathbf{x}_N - \mathbf{x}_1||) & \dots & \phi(||\mathbf{x}_N - \mathbf{x}_N||) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \end{bmatrix}. \tag{3.2}$$

Here, the matrix composed of RBF evaluations is referred to as the basis evaluation matrix.

Gaussian	$\phi(r) = e^{-(\epsilon r)^2}$
Multiquadric	$\phi(r) = \sqrt{1 + (\epsilon r)^2}$
Inverse multiquadric	$\phi(r) = 1/\sqrt{(1 + (\epsilon r)^2)}$
Polyharmonic spline	$\phi(r) = r^k, \quad k = 1, 3, 5, ...$
Thin plate splines	$\phi(r) = r^2 \ln(r)$
Compactly supported RBFs	$\phi(r) = \begin{cases} \exp(\frac{-1}{1-(\epsilon r)^2}) & \text{for } r < \frac{1}{\epsilon} \\ 0 & \text{otherwise} \end{cases}$

Table 3.2: Commonly Used Radial Basis Function Kernels.

3.1.1 Shape Parameter

Many of the commonly used radial basis function kernels feature a shape parameter, traditionally labeled ϵ, as can be seen in the RBFs found in Table 3.2. The shape parameter controls the flatness of the radial kernel and plays an important role on the accuracy and stability of the interpolant. Figure 3.1 shows the commonly used multiquadric kernel with various values for the shape parameter. However, the

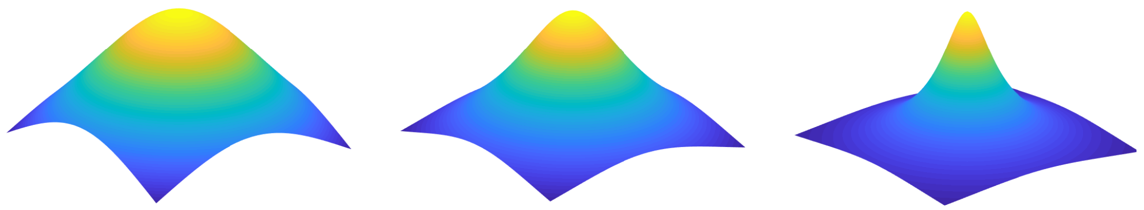

Figure 3.1: Inverse Multiquadric Radial Basis Functions with Shape Parameter $\epsilon = 1$, 2, and 5 from Left to Right.

problem known in the literature as the "uncertainty principle" presented in [39] describes one of the major limitations of computing the RBF coefficients using equation (3.2). The issue is characterized by the trade-off faced when using small values for the shape parameter since decreasing ϵ increases the convergence order but also increases the condition number of the interpolation matrix. This problem is most commonly encountered when using a large number of data points. We explore this issue with a simple example and consider the function $f(x, y) = x$ for $(x, y) \in [-1, 1] \times [-1, 1]$. Figure 3.2 shows the condition number for the basis evaluation matrix using an inverse multiquadric kernel with various values of ϵ where N^2 is the number of points used. Figure 3.2 also shows the error of the RBF interpolant for the various values of ϵ and N. Comparing both figures we see that as ϵ increases the condition number of the matrix decreases however the error actually increases. There are two main ap-

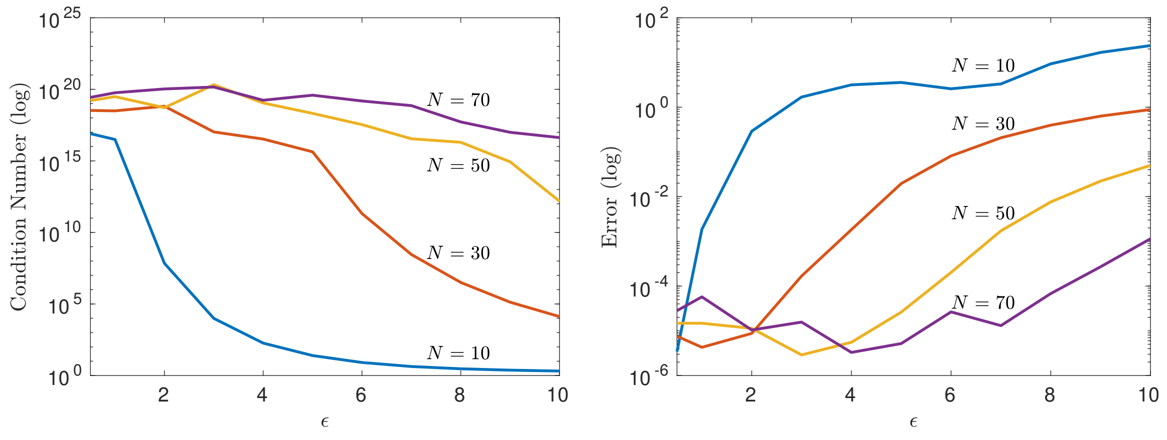

Figure 3.2: Left: Condition Number of the Basis Evaluation Matrix for Various Values of ϵ and N. Right: Error for the Radial Basis Function Approximation of $f(x, y) = x$.

proaches to overcome this issue, the Contour-Padé method described in [19] and the RBF-QR method explored in [18] and [17]. However, since the work in this dissertation utilizes the partition of unity method, the number of nodes on each subdomain is small and an acceptable ϵ can be chosen without obtaining a severely ill conditioned interpolation matrix.

3.1.2 Differentiation Operators on Surfaces

The following section presents the discrete approximation of the differentiation operators on the surface used for the radial basis function partition of unity (RBF-PU) method. The surface gradient is computed by using the normal vector to construct the projected gradient for a general RBF kernel and then discretized to get the differentiation matrix as done in [21] and summarized below. We begin by first computing the continuous surface gradient in three dimensions for simplicity. Given a smooth manifold $\mathbb{M}$ embedded in $\mathbb{R}^3$ and a vector $\mathbf{x}$ on $\mathbb{M}$. We denote the unit normal vector at $\mathbf{x}$ by $\mathbf{n} = (n^x, n^y, n^z)$ and we let $\mathbf{P}$ be the projection into the vector space tangent to $\mathbb{M}$ at $\mathbf{x}$. Then the surface gradient operator, $\nabla_{\mathbb{M}}$, at $\mathbf{x}$ is given by

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

If e^x, e^y, e^z, are the standard unit vectors in the x, y, and z directions in $\mathbb{R}$, then

$$\nabla_M = \begin{bmatrix} (e^x \cdot P)\nabla \\ (e^y \cdot P)\nabla \\ (e^z \cdot P)\nabla \end{bmatrix} = \begin{bmatrix} (e^x - n^x\mathbf{n}) \cdot \nabla \\ (e^y - n^y\mathbf{n}) \cdot \nabla \\ (e^z - n^z\mathbf{n}) \cdot \nabla \end{bmatrix} = \begin{bmatrix} \mathbf{p}^x \cdot \nabla \\ \mathbf{p}^y \cdot \nabla \\ \mathbf{p}^z \cdot \nabla \end{bmatrix} =: \begin{bmatrix} G^x \\ G^y \\ G^z \end{bmatrix}.$$

The surface gradient is then applied to a general radial kernel function. We denote the radial kernel by ϕ and the radial basis functions as $\phi_j(\mathbf{x}) = \phi(||\mathbf{x} - \mathbf{x}_j||)$ with $||\cdot||$ being the 2-norm. The chain rule then gives us

$$\begin{aligned} \nabla\phi_j(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x}\phi_j(\mathbf{x}) \\ \frac{\partial}{\partial y}\phi_j(\mathbf{x}) \\ \frac{\partial}{\partial z}\phi_j(\mathbf{x}) \end{bmatrix} &= \begin{bmatrix} \phi'(||\mathbf{x} - \mathbf{x}_j||)\frac{\partial}{\partial x}||\mathbf{x} - \mathbf{x}_j|| \\ \phi'(||\mathbf{x} - \mathbf{x}_j||)\frac{\partial}{\partial y}||\mathbf{x} - \mathbf{x}_j|| \\ \phi'(||\mathbf{x} - \mathbf{x}_j||)\frac{\partial}{\partial z}||\mathbf{x} - \mathbf{x}_j|| \end{bmatrix} \\ &= \begin{bmatrix} \frac{x - x_j}{||\mathbf{x} - \mathbf{x}_j||}\phi'(||\mathbf{x} - \mathbf{x}_j||) \\ \frac{y - y_j}{||\mathbf{x} - \mathbf{x}_j||}\phi'(||\mathbf{x} - \mathbf{x}_j||) \\ \frac{z - z_j}{||\mathbf{x} - \mathbf{x}_j||}\phi'(||\mathbf{x} - \mathbf{x}_j||) \end{bmatrix} = \frac{\mathbf{x} - \mathbf{x}_j}{||\mathbf{x} - \mathbf{x}_j||}\phi'(||\mathbf{x} - \mathbf{x}_j||). \end{aligned}$$

Here, we have used $'$ to denote the derivative with respect to the Euclidean distance. The surface gradient can then be computed and here we do this for x but it is analogous for y and z,

$$\begin{aligned} G^x\phi_j(\mathbf{x}) &= \mathbf{p}^x \cdot \nabla\phi_j(\mathbf{x}) \\ &= (e^x - n^x\mathbf{n}) \cdot \nabla\phi_j(\mathbf{x}) \\ &= (e^x - n^x\mathbf{n})\frac{\mathbf{x} - \mathbf{x}_j}{||\mathbf{x} - \mathbf{x}_j||}\phi'(||\mathbf{x} - \mathbf{x}_j||). \end{aligned}$$

Thus

$$\begin{aligned}
G^x\phi_j(\mathbf{x}) &= \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} n^x n^x \\ n^x n^y \\ n^x n^z \end{bmatrix} \right) \cdot \begin{bmatrix} x - x_j \\ y - y_j \\ z - z_j \end{bmatrix} \frac{\phi'(||\mathbf{x} - \mathbf{x}_j||)}{||\mathbf{x} - \mathbf{x}_j||} \\
&= \begin{bmatrix} 1 - n^x n^x \\ -n^x n^y \\ -n^x n^z \end{bmatrix} \cdot \begin{bmatrix} x - x_j \\ y - y_j \\ z - z_j \end{bmatrix} \frac{\phi'(||\mathbf{x} - \mathbf{x}_j||)}{||\mathbf{x} - \mathbf{x}_j||} \\
&= (1 - n^x n^x)(x - x_j) - n^x n^y (y - y_j) - n^x n^z (z - z_j) \frac{\phi'(||\mathbf{x} - \mathbf{x}_j||)}{||\mathbf{x} - \mathbf{x}_j||}.
\end{aligned}$$

Finally, we take the continuous surface gradient and discretize it to define a differentiation matrix. We now compute the RBF approximation of the surface gradient using the expression above. We recall the interpolant from (3.1),

$$\tilde{u}(\mathbf{x}) = \sum_{j=0}^{N} c_j \phi(||\mathbf{x} - \mathbf{x}_j||).$$

Applying the projected gradient operator yields,

$$\begin{aligned}
G^x \tilde{u}(\mathbf{x}) &= \sum_{j=0}^{N} c_j (G^x \phi_j(\mathbf{x})) \\
&= \sum_{j=0}^{N} c_j \Big((1 - n^x n^x)(x - x_j) - n^x n^y (y - y_j) - n^x n^z (z - z_j) \frac{\phi'(||\mathbf{x} - \mathbf{x}_j||)}{||\mathbf{x} - \mathbf{x}_j||} \Big).
\end{aligned}$$

We recall the coefficients are defined by collocation and by assuming A to be the basis evaluation matrix, the coefficients are determined by $\mathbf{c} = A^{-1}u(\mathbf{x})$. Then, by defining the matrix D^x such that the columns are defined by

$$D^x_j = (1 - n^x n^x)(x - x_j) - n^x n^y (y - y_j) - n^x n^z (z - z_j) \frac{\phi'(||\mathbf{x} - \mathbf{x}_j||)}{||\mathbf{x} - \mathbf{x}_j||},$$

this yields,

$$\begin{aligned} G^x \tilde{u}(\mathbf{x}) &= \sum_{j=0}^{N} c_j D_j^x \\ &= D^x \mathbf{c} \\ &= D^x (A^{-1} u(\mathbf{x})). \end{aligned}$$

Hence, the differentiation matrix for the first component of the surface gradient is given by the matrix $(D^x A^{-1})$. The surface Laplacian operator is therefore given by

$$L = G^x G^x + G^y G^y + G^z G^z. \tag{3.3}$$

3.2 RBF Partition of Unity

Since RBF interpolation requires us to solve a linear system that can be computationally expensive for large scale problems, we instead consider a localized RBF approach with the partition of unity method. The rest of the work in this dissertation focuses on using the RBF-PU method to solve PDEs on various surfaces including the unit sphere, a red blood cell, a torus, and a bunny. We use the method of lines with the partition of unity method combined with the discrete differentiation operators defined in Section 3.1.2 for the local approximations for the spatial component.

3.2.1 Subdomains and Partition of Unity Weight Function

The partition of unity scheme combines local approximations on overlapping patches as defined in Section 1.1. Various subdomain shapes have been used in the literature such as squares, ellipses and ellipsoids as well as more general patch layouts as explored in [23] and [38]. However, we exclusively use disks defined on the surface to cover the domain. We implement this by choosing the centers of the subdomains $\{\mathbf{k}_i\}_{i=0}^{M}$ and defining the subdomains as $\{\mathbf{x} \in \mathbb{M} | \; ||\mathbf{x} - \mathbf{k}_i|| < \delta\}$ for some predetermined δ. The number of centers and magnitude of δ determines the amount of overlap between the subdomains and the number of subdomains that each one overlaps with. Figure 3.3 shows an example of the unit sphere decomposed into 30 overlapping regions. The choice of centers for the subdomains is described in the following section.

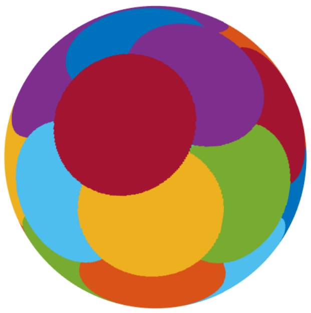

Figure 3.3: Unit Sphere Decomposed into 30 Subdomains for the Partition of Unity Method.

In this section, we also introduce the weight function utilized for the partition of unity method. As in [3], we use Shepard's method based on Wendland's compactly supported RBFs as the weight function. We use Wendland's C^2 compactly supported radial basis function [46] applied to the distance from the subdomain center for the weight function,

$$w(r) = (1-r)_+^4(4r+1).$$

Assuming $\{u_i\}$ denote our local approximations, then the following Shepard's method evaluation gives us the global approximation

$$\tilde{u}(\mathbf{x}) = \frac{\sum_{i=1}^{M} w_i(||\mathbf{x}-\mathbf{x}_i||)u_i(\mathbf{x})}{\sum_{i=1}^{M} w_i(||\mathbf{x}-\mathbf{x}_i||)} = \sum_{i=1}^{M}\left[\frac{w_i(||\mathbf{x}-\mathbf{x}_i||)}{\sum_{i=1}^{M} w_i(||\mathbf{x}-\mathbf{x}_i||)}\right]u_i(\mathbf{x}).$$

The second formulation is typically more convenient when implementing this method.

3.3 Node Generation

In this section, we describe the algorithm used to generate the nodes used for the RBF-PU method. It is well known that using "good nodes" is critical for accurate and stable approximations. Although choosing nodes that lie on a coordinate system is a common approach, these distributions often result in singularities and distortions. This is most obviously seen with the over clustering and singularities on the poles of the sphere with the latitude longitude coordinate system. On the other hand, scattered nodes don't inherit these issues as they do not lie on any underlying grid. Even so, choosing good nodes is still a very difficult task as there is no known set of "optimal points" for RBF interpolation. However, nodes which are roughly uniformly distributed have been shown to be desirable when working with RBFs on surfaces [33], [25]. Although it may be easy to generate equally spaced nodes in simple domains, that is not the case when working on surfaces. Through out this work, we consider Fekete points as it has been shown that they are asymptotically equally spaced [6]. In order to define Fekete points, we first define the Vandermonde matrix for any set of ordered points $\mathbf{z} = \{z_1, \dots, z_k\}$ and ordered set of polynomials $\mathbf{q} = \{q_1, \dots, q_k\}$ as

$$V(\mathbf{z};\mathbf{q}) = \begin{bmatrix} q_1(z_1) & q_2(z_1) & \dots & q_k(z_1) \\ q_1(z_2) & q_2(z_2) & \dots & q_k(z_2) \\ \vdots & \ddots & & \vdots \\ q_1(z_k) & q_2(z_k) & \dots & q_k(z_k) \end{bmatrix}.$$

Fekete points are defined as the set of points which maximize the determinant of the Vandermonde matrix, V, for a set of functions $\{q_1, \dots, q_k\}$ or more specifically

$$\max_{\mathbf{z}} |V(\mathbf{z};\mathbf{q})|.$$

Evidently, computing Fekete points quickly becomes a computationally expensive large scale problem since it a nonlinear optimization problem. Given this is an NP-

Hard problem [5], using true Fekete points is not practical and we opt for using approximate Fekete points. In [41], Sommariva and Vianello present a straightforward algorithm for computing approximate Fekete points using a column pivoting QR factorization. Although this technique was presented for polynomial interpolation, we find that it does well with RBF interpolation as well. The method considers the Vandermonde matrix and takes advantage of the fact that the QR factorization algorithm is a greedy algorithm that selects pivot columns based on maximizing the associated volume of the submatrices. Algorithm 1 taken from [41] is a sketch of the greedy QR factorization algorithm used.

Algorithm 1 Greedy Algorithm for QR Factorization

```
%Initialize Index
$id = [\ ]$
for $k = 1, \dots, N$ do
    select column with largest norm, $i_k$, and add it to the index, $id = [id, i_k]$
    remove orthogonal projection from remaining columns
end for
```

The index of the permutations from this algorithm therefore gives a heuristic ranking of the "importance" of the nodes. We implement this algorithm for choosing our interpolation nodes as well as the centers of the subdomains in the following numerical examples. Figure 3.4 shows the candidate nodes on the sphere and the chosen interpolation nodes chosen from said candidate nodes using the QR factorization algorithm. Figure 3.5 is a similar example but on the surface of a red blood cell.

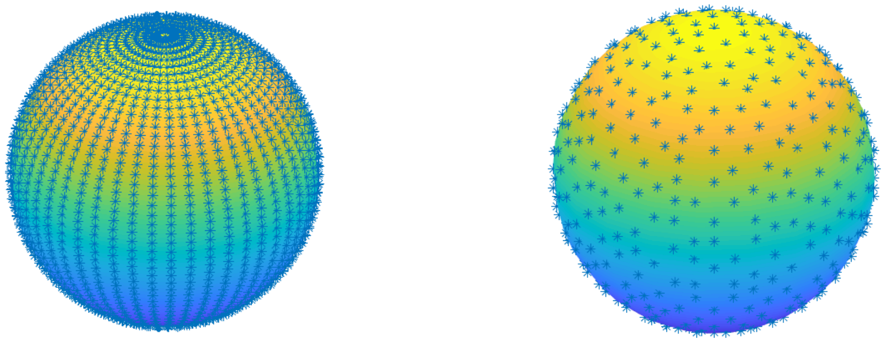

Figure 3.4: Candidate Points and Nodes Generated by the QR Factorization Algorithm on the Sphere.

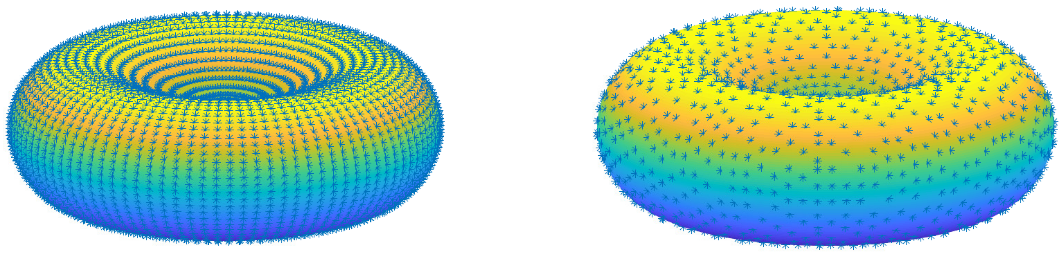

Figure 3.5: Candidate Points and Nodes Generated by the QR Factorization Algorithm on the Surface of a Red Blood Cell.

3.4 Numerical Examples

In this section, we use the RBF-PU method to numerically solve two time dependent PDEs and study the accuracy and stability. We consider PDEs defined on the unit sphere and the surface of a red blood cell and note that there are no boundary conditions for these problems.

3.4.1 Method of Lines

We use the method of lines to solve the initial value PDEs considered in this chapter. In the following section, we solve the diffusion equation and a reaction-diffusion system on various surfaces. Thus we formulate the general PDE problem as the following:

$$\frac{\partial u}{\partial t} = \delta \Delta u + f(t, u). \tag{3.4}$$

Here $\delta > 0$ is the diffusion coefficient and f is the forcing term. In the case of the diffusion equation in the following section, the forcing term is simply zero. If we allow $\mathbf{u}$ to be the vector of samples of u, then the method of lines approximation takes the form,

$$\frac{d\mathbf{u}}{dt} = \delta L_w \mathbf{u} + f(t, \mathbf{u}). \tag{3.5}$$

Here, L_w is the weighted average of the differentiation matrices defined in 3.1.2. In the following section, we use appropriate time stepping-schemes to advance in time and use this formulation to inquire about stability based on the eigenvalues of the differentiation matrix.

3.4.2 Unit Sphere

We now apply the method described above to two problems. We first consider the diffusion equation on the unit sphere,

$$\begin{aligned} u_t(\mathbf{x}, t) &= \Delta u(\mathbf{x}, t), \qquad ||\mathbf{x}||_2 = 1 \\ u(\mathbf{x}, 0) &= xyz + 2. \end{aligned}$$

Here, the initial condition shown in Figure 3.6 is defined using a spherical harmonic function, $Y(\mathbf{x}) = xyz$. The choice of initial condition allows us to solve the PDE analytically and the exact solution is given by

$$u(\mathbf{x}, t) = e^{-12t} xyz + 2.$$

The method of lines is used with the RBF-PU method described above for the spatial discretization and the forward Euler time stepping scheme is used to progress in time. A time step of $\Delta t = 0.001$ is used with 500 total nodes over 20 subdomains. The centers of the subdomains and the node points are chosen using the QR factorization algorithm described in Section 3.3. Figure 3.6 shows the initial condition as well as the solution which converges to a constant value of 2 as was expected and the max error at $t = 10$ is approximately 8.6×10^{-9}.

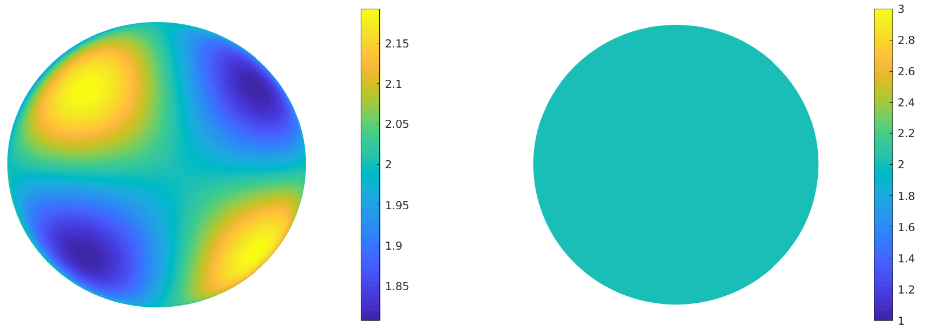

Figure 3.6: Left: The Initial Condition on the Unit Sphere. Right: The Numerical Solution for the Diffusion Equation at $t = 10$ with Time Step $\Delta t = 10^{-3}$.

The eigenvalues of the Laplacian operator on the sphere defined in Section 3.1.2 are shown in Figure 3.7. Note that the true eigenvalues of the Laplace operator on the sphere should all be real non-positive integers. However, the eigenvalues observed in Figure 3.7 show a couple of eigenvalues with imaginary parts but we point out the scale of the two axis and note that the magnitude of the imaginary parts are two orders of magnitude smaller than the real parts. We attribute the error in the eigenvalues to the partition of unity method since we found it to be unavoidable in all of our experiments. We again note that the eigenvalues all have non-positive real parts. It is known that a necessary condition for stability is for the eigenvalues of the differentiation matrix to be in the stability domain of the time stepping scheme. Since the stability region of the forward Euler method is defined by the circle on the left half plane defined by $\{z \in \mathbb{C} | \; |z+1| \leq z\}$, an appropriately small time step ensures the eigenvalues all lie with in the stability region.

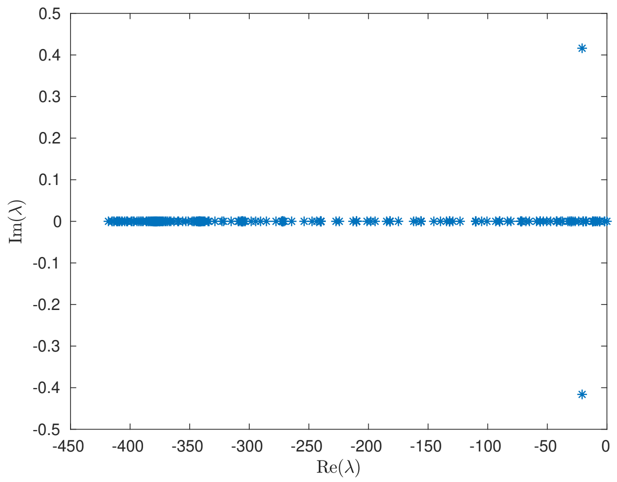

Figure 3.7: Eigenvalues of the Laplacian Operator for the RBF-PU Method on the Unit Sphere.

Figure 3.8 shows the semi-log plots of the relative error with respect to the square root of the total number of interpolation nodes. We now study the convergence of the error of the numerical solution. Note that only the global number of nodes is varied and the number of subdomains is kept constant. The convergence of the relative error is shown for $\epsilon = 1$ and $\epsilon = 2$ for three different kernels: Gaussian, Multiquadric, and Inverse Multiquadric. We note that the error for all three kernels converges faster in the top figure which corresponds with the smaller ϵ, however, the error stops decaying due to the ill-conditioning of the interpolation matrices. For $\epsilon = 2$, on the other hand, ill-conditioning is not an issue for the number of points used in this experiment.

3.4.3 Reaction-Diffusion System on a Red Blood Cell

We now consider a more interesting problem, a system of reaction-diffusion equations taken from [21]. This type of system is often used as a model in biology, geology, physics, ecology and chemistry. A common application of this system is used for

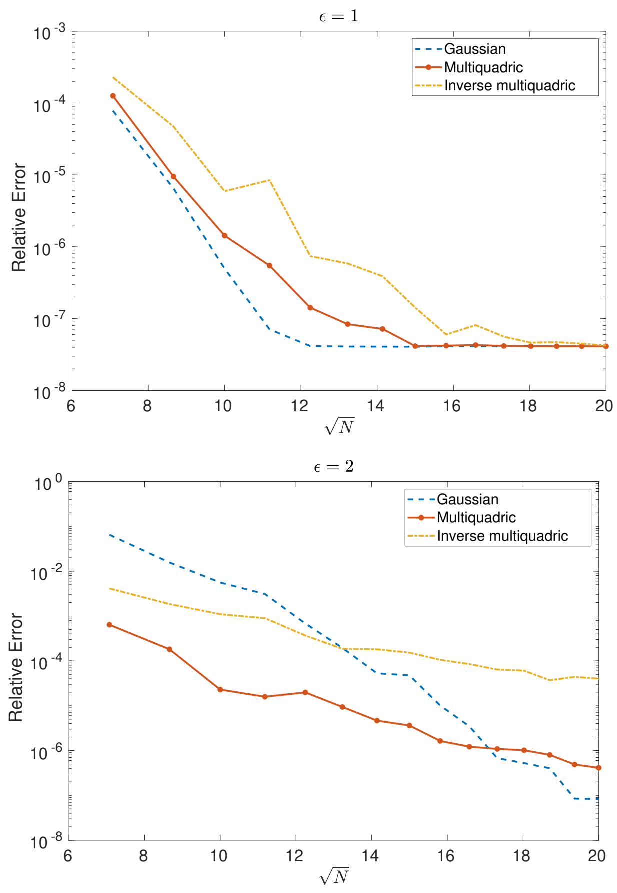

Figure 3.8: Relative Error of the Numerical Solution of the Diffusion Equation for Various Values of N.

modeling of local chemical reactions in which the substances diffuse over a surface or natural pattern formations found in biology. We look at the Turing system presented in [44],

$$\begin{aligned}\frac{\partial u}{\partial t} &= \delta_u \Delta u + \alpha u(1 - \tau_1 v^2) + v(1 - \tau_2 u)\\ \frac{\partial v}{\partial t} &= \delta_v \Delta v + \beta v(1 + \frac{\alpha \tau_1}{\beta} uv) + u(\gamma + \tau_2 u).\end{aligned}$$

This particular nonlinear system leads to stable pattern formations and has gained recent attention in the computer graphics field for producing interesting textures on surfaces. The system consists of two equation: the activator u and the inhibitor v. Based on the choice of parameters, the solution can converge to a state of various patterns. We concentrate on the parameters that produce stripes which is favored by having a larger τ_1 as well as spots which is done with a larger τ_2. In this particular example, the reaction-diffusion system is solved on the surface of the cell. The parameters for the equations are chosen such that we expect a pattern of spots and are shown in Table 3.3.

δ_v	δ_u	α	β	γ	τ_1	τ_2
4.5×10^{-3}	0.516*δ_v	0.899	-0.91	-0.899	0.02	0.2

Table 3.3: Parameter Values for the Turing System on the Surface of a Red Blood Cell.

The solution is found using the method of lines described in Section 3.4.1 with the RBF partition of unity method implemented in the spatial discretization and the Semi-implicit Backwards Difference Formula of order 2 (SBDF2) for the time stepping scheme. The SBDF2 is a semi-implicit technique which uses an implicit backward differentiation formula for the diffusion term and the second order Adams-Bashforth for the forcing terms. The algorithm we used is followed closely by the one presented in [21]. Since the forcing terms are handled explicitly by the SBDF2 scheme then the forcing terms use the previous iterations. However, since an implicit method is used for the diffusion term, the current iteration of the solution must be solved for at each iteration. Thus the differentiation matrix needs to be inverted each time step but is done so efficiently by pre-computing a sparse LU decomposition. The initial conditions for both u and v are defined as zero except for a strip of values around the equator drawn from a random uniform distribution between -0.5 and 0.5 as shown in Figure 3.9. Figure 3.9 shows the solution at $t = 300$ using $\Delta t = 0.1$ and 3,000 total nodes over 300 subdomains. Note the pattern of spots is observed as expected. We now investigate one aspect of stability for the time stepping scheme

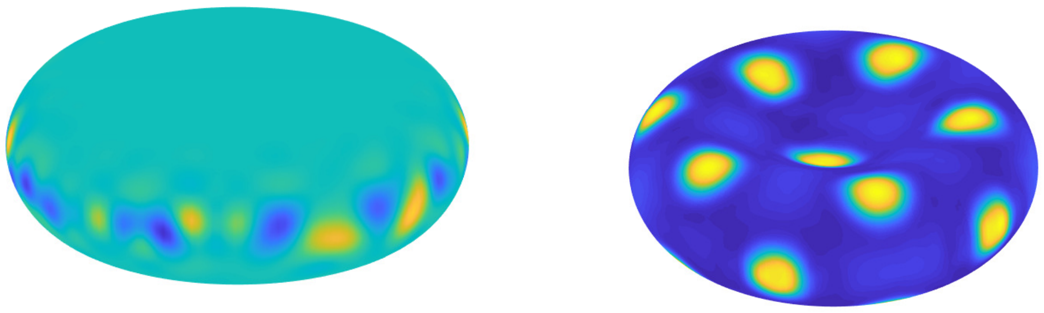

Figure 3.9: Left: The Initial Condition for the System of Reaction-Diffusion Equations on the Surface of a Red Blood Cell. Right: The Numerical Solution Using the Parameters Presented in Table 3.3 for $t = 300$ and Time Step $\Delta t = 0.1$.

implemented. The eigenvalues of the discrete Laplacian operator on the surface of the cell are shown in Figure 3.10 where all eigenvalues have negative real parts. We recall that the Laplacian operator is handled in an implicit backward differentiation fashion which has a stability region defined by the exterior of a circle located in the right half plane. Then it can be expected that a necessary requirement for stability is to have the eigenvalues of the Laplacian operator in the left half plane as is observed in Figure 3.10.

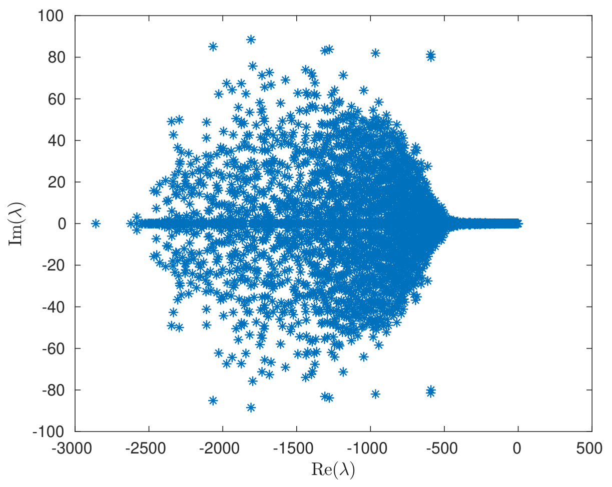

Figure 3.10: The Eigenvalues of the Laplacian Operator for the RBF-PU Method on the Surface of a Red Blood Cell.

Chapter 4

RBF - PU ON LARGE DATASETS

In this chapter, the radial basis function partition of unity method is adapted for applications with large data sets. The RBF-PU method is ideal for large and complicated surfaces since the more intricate surfaces require a large number of points for accuracy and highlight the benefits of the domain decomposition implemented in our method. We specifically concentrate on solving the reaction-diffusion PDE with a large set of points on the surface of a torus and then on the surface of the Stanford bunny.

4.1 Point Node Selection

Although the RBF-PU method described in Chapter 3 is an efficient technique that only takes a small fraction of the computational time when compared to other approaches such as the spectral cubed sphere method introduced in Chapter 2, it can be further improved. In this section, we introduce two techniques to increase the efficiency of the point node selection process.

4.1.1 Randomized QR Factorization

We recall that the RBF-PU method described in Chapter 3 uses a column pivoting QR factorization algorithm to find the near optimal points used for the solution of the PDE. However, the QR factorization in this algorithm takes up a considerable amount of the computational time for the whole method. Thus in this section, we implement an algorithm developed in [29], in which randomization is used to accelerate the pivot selection for the QR factorization. This is done using randomized projections, the idea being that instead of computing the column pivoting QR factorization of a matrix A, we identify the pivot columns by computing the column pivoting QR factorization of the much smaller randomized projection of A. Note that since we are considering a random projection of the matrix A, the matrices have approximately the same linear dependencies between its columns and thus result in similar pivot choices. Formally, given a $m \times n$ matrix A, the $b \times n$ sampling matrix Y is created from the product with G, a Gaussian random matrix of size $b \times n$, so $Y = GA$. Thus, the column pivoting QR factorization is performed on this matrix, Y, in order to identify the index of the pivot columns. This computation is more efficient since b is assumed to be much smaller than m so Y is small compared to A. This technique is implemented by using a package developed by Per-Gunnar Martinsson in [29]. This allows us to efficiently use the column pivoting QR factorization to find good nodes for our RBF-PU method for large data sets in the following sections.

4.1.2 Point Node Preselection

In this section, we develop a second method to accelerate the point node selection process. A preselection technique is implemented to decrease the time of the global point node selection process described in the section above. The surface is decomposed into various subdomains as in the partition of unity method described in Section 3.2.1. Then a set of points is chosen on each subdomain using the column pivoting QR factorization algorithm. Since the set of points on each subdomain is not very large then randomized QR factorization is not necessary. The nodes selected on each subdomain are then concatenated and the randomized QR factorization algorithm is used to select the global set of point nodes as described in the section above. The efficiency of this process comes from the fact that computing multiple small QR factorizations is faster than one large QR factorization. Note that a QR factorization is still computed on a large matrix even with the preselection, however, the preselection process eliminated a large fraction of the potential candidate points and the matrix although still large is significantly smaller.

4.1.3 Time Comparison

The computational time difference between these methods is showcased by looking at the point selection process on the surface of a torus. We consider $45,000$ candidate points and choose $15,000$ of those points. Table 4.1 shows the time comparison for the basic MATLAB built in QR factorization, the Randomized QR factorization, and the preselection with the built in QR factorization as well as the randomized QR factorization. Using the basic MATLAB built in function for the QR decomposition for the point selection process was the slowest. Implementing just one of these techniques, mainly the randomized QR factorization method, reduced the computational time significantly by a third. Note that the randomized QR decomposition used here is implemented in C using the LAPACK library. However, we feel this is a fair comparison since MATLAB uses the LAPACK library in the QR decomposition as well and for the randomized QR factorization point selection process, only the QR decomposition is done outside of MATLAB. We also consider the subdomain preselection alone, in which we use 50 subdomains. On each subdomain, we select 500 nodes and since some nodes are selected in multiple subdomains, we end up with 20,209 preselected points. From there, we use the basic MATLAB built-in QR factorization to choose the final 15,000 points. This approach is slightly faster than using just the randomized QR factorization. Finally, we combine both methods and do the preselection on the 50 subdomains as well as use the randomized QR factorization and we see that the wall time is 14 times faster than using just the built in MATLAB QR factorization. This is the method that will be implemented for the rest of this chapter.

Method	Wall Time (minutes)
Built in MATLAB QR	70
Randomized QR	22
Preselection and Built in MATLAB QR	18
Preselection and Randomized QR	5

Table 4.1: Time Comparison of the Column Pivoting QR Factorization Algorithms for the Point Selection Process.

4.2 Numerical Examples

4.2.1 Reaction-Diffusion System on a Torus

In this section, we focus on large surfaces and look at the system of reaction-diffusion equations described in Section 3.4.3 on the surface of a torus defined parametrically by

$$x(\theta, \phi) = (20 + 9\cos(\theta))\cos(\phi)$$
$$y(\theta, \phi) = (20 + 9\cos(\theta))\sin(\phi)$$
$$z(\theta, \phi) = 9\sin(\theta)$$

for $(\theta, \phi) \in [0, 2\pi] \times [0, 2\pi]$. We apply the RBF-PU method described in Chapter 3 with the QR factorization speed ups discussed in the previous section. We begin by considering 45,000 candidate points shown in Figure 4.1. The domain is decomposed into 50 subdomains as seen in Figure 4.1. Figure 4.2 shows the 18,473 points selected in the preselection process where 750 points were chosen on each subdomain. The final 12,000 point nodes chosen for computing the solution are shown in Figure 4.2.

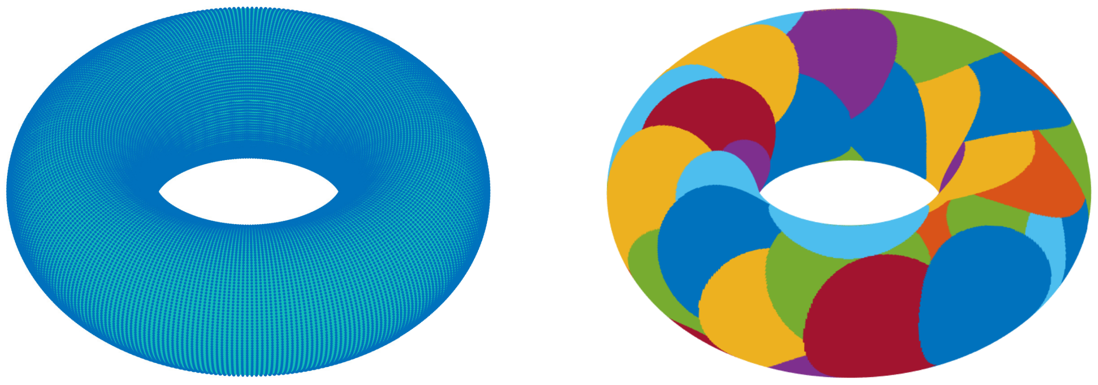

Figure 4.1: Left: The Candidate Points for the Point Selection Process. Right: The 50 Subdomains Used in the Partition of Unity Method.

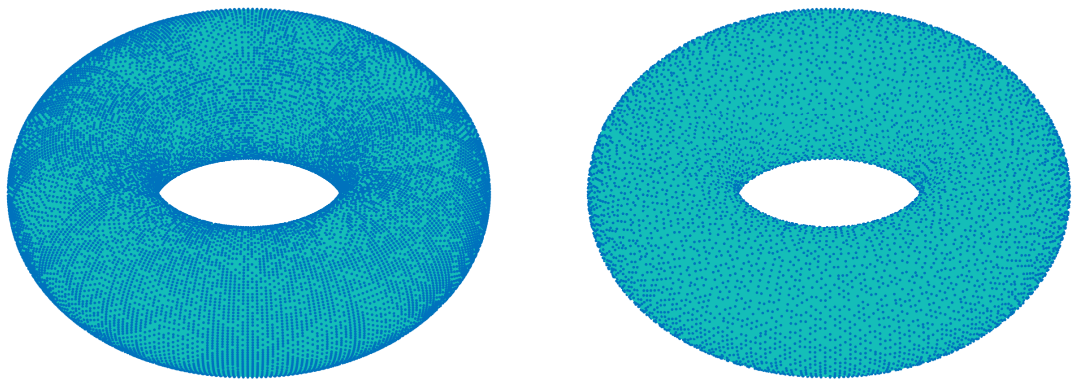

Figure 4.2: Left: The 23,000 Points Selected in the Preselection Process and the Final 12,000 Points Selected in the Global Point Selection Process.

We now revisit the reaction-diffusion equations introduced in section 3.4.3,

$$\begin{aligned}\frac{\partial u}{\partial t} &= \delta_u \Delta u + \alpha u(1 - \tau_1 v^2) + v(1 - \tau_2 u) \\ \frac{\partial v}{\partial t} &= \delta_v \Delta v + \beta v(1 + \frac{\alpha \tau_1}{\beta} uv) + u(\gamma + \tau_2 u).\end{aligned}$$

We recall, this particular nonlinear system leads to stable pattern formations of either spots or stripes. The choice of parameters dictates the pattern that the system will converge to and in this exercise they are chosen such that the system converges to a pattern of spots and are shown in Table 4.2.

δ_v	δ_u	α	β	γ	τ_1	τ_2
5	0.516*δ_v	0.899	-0.91	-0.899	0.02	0.2

Table 4.2: Parameter Values for the Turing System on the Surface of a Torus.

Again, the solution is found using the method of lines where SBDF2 is used for the time stepping scheme with a time step of $\Delta t = 0.01$. The initial condition is defined by uniformly distributed random values between -0.5 to 0.5. Figure 4.3 shows the solution at $t = 300$ and the pattern of spots can be seen as expected.

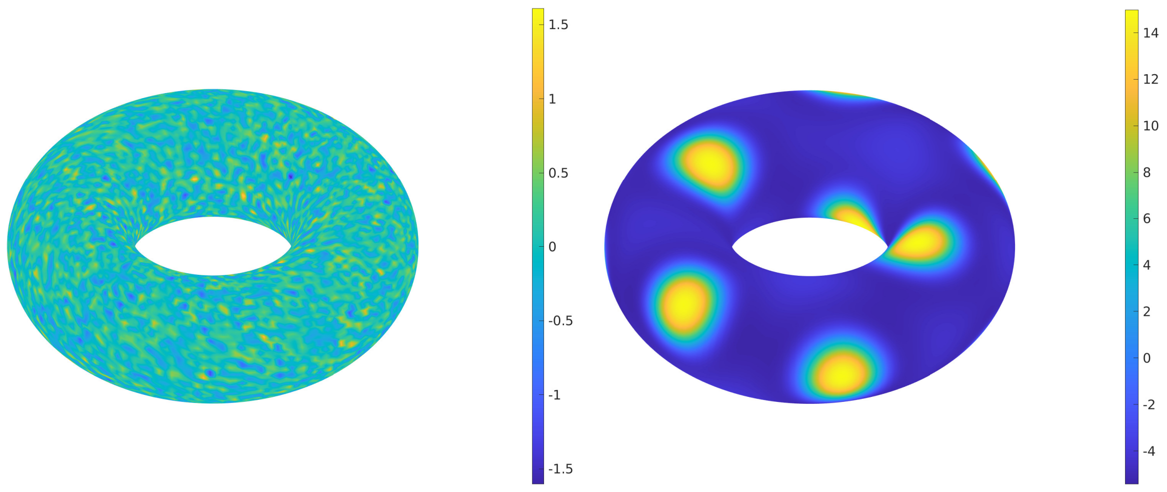

Figure 4.3: Left: The Initial Condition for the System of Reaction-Diffusion Equations on the Surface of a Torus. Right: The Numerical Solution Using the Parameters Presented in Table 4.2 at $t = 300$ and Time Step $\Delta t = 0.01$.

As stated in Section 3.4.3, a necessary condition for the stability in the time stepping scheme in the method of lines is for the eigenvalues of the Laplacian operator to be in the left half plane. Figure 4.4 shows the eigenvalues of the surface Laplacian operator for the torus and we note that all the eigenvalues have negative real parts.

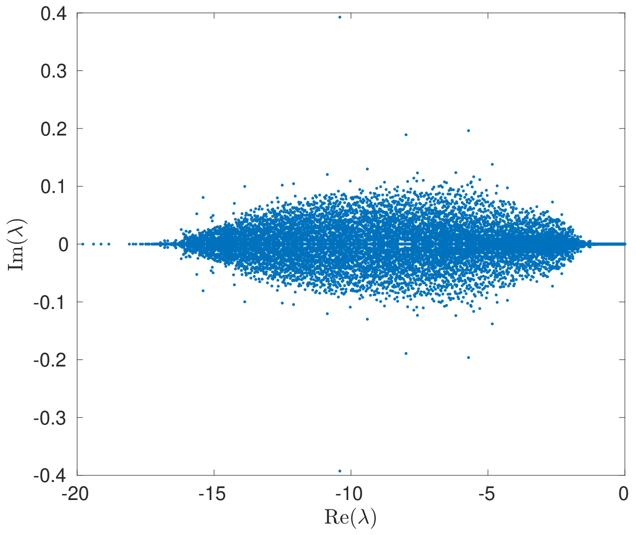

Figure 4.4: Eigenvalues for the Laplacian Operator for the RBF-PU Method on the Surface of a Torus.

4.2.2 Reaction-Diffusion System on the Stanford Bunny

In this section, we apply the RBF partition of unity method to the system of reaction-diffusion equations described in Chapter 3 with the point selection speed up once more but on a more intricate surface, specifically on the surface of a rabbit. The system was similarly solved in [40] using a radial basis function finite difference method. The point cloud data set used here was derived from the Stanford Bunny model and can be seen in Figure 4.5.

The original point cloud contained 35,947 points and the point nodes were chosen by implementing the preselection on 50 subdomains and then using the randomized

QR factorization algorithm to choose the global set of point nodes used for the solution. Figure 4.5 shows the original point cloud as well as the 50 subdomains. Figure 4.6 shows the 24,137 total points chosen from the preselection process which selected 500 points on each subdomain as well as the 10,000 points chosen from the final global selection.

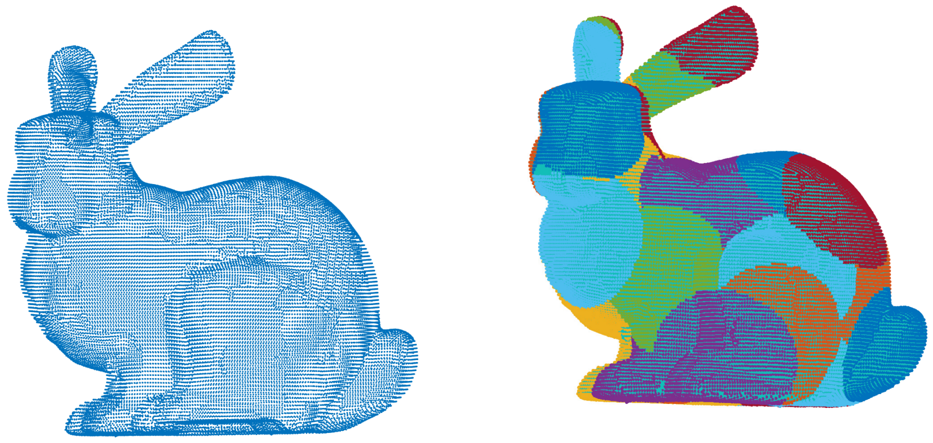

Figure 4.5: Left: The Candidate Points for the Point Selection Process. Right: The 50 Subdomains Used in the Partition of Unity Method.

We state the reaction-diffusion equations again for convenience,

$$\begin{aligned}\frac{\partial u}{\partial t} &= \delta_u \Delta u + \alpha u(1-\tau_1 v^2) + v(1-\tau_2 u)\\ \frac{\partial v}{\partial t} &= \delta_v \Delta v + \beta v(1+\frac{\alpha\tau_1}{\beta}uv) + u(\gamma+\tau_2 u).\end{aligned}$$

We recall, the choice of parameters dictates the state that the system will converge to and we run two experiments to showcase both patterns. The parameters chosen in these numerical examples are shown in Table 4.3 below and are chosen such that we expect to see the spots in one case and stripes in the other by emphasizing τ_1 in the case with the stripes and τ_2 with the spots.

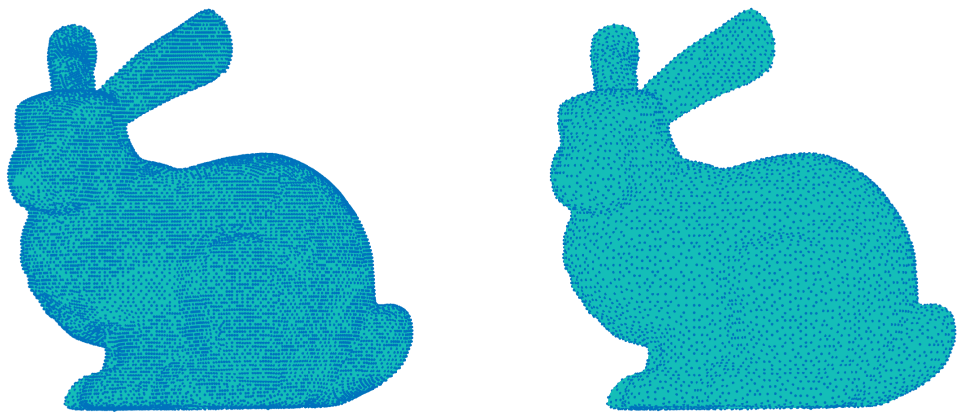

Figure 4.6: Left: The 24,137 Points Selected in the Preselection Process and the Final 10,000 points Selected in the Global Point Selection Process.

	δ_v	δ_u	α	β	γ	τ_1	τ_2
Spots	5×10^{-5}	0.516*δ_v	0.899	-0.91	-0.899	0.02	0.2
Stripes	2.5×10^{-5}	0.516*δ_v	0.899	-0.91	-0.899	0.35	0

Table 4.3: Parameter Values for the Turing System on the Stanford Bunny.

As in the previous cases, the solution is found using the method of lines which is described in more detail in Section 3.4.3 where SBDF2 is used for the time stepping scheme with a time step of $\Delta t = 0.1$. The same initial condition is used in both cases and is again defined by uniformly distributed random values between -0.5 to 0.5. Figure 4.8 shows the solution at $t = 300$ for the case were the spots are expected. We note that the stripes take longer for the pattern to stabilize and Figure 4.9 shows the solution at $t = 3,000$ where the stripes can be seen.

As in the previous two cases on the surface of the red blood cell and torus, the eigenvalues of the Laplacian operator must lie in the left half plane for stability.

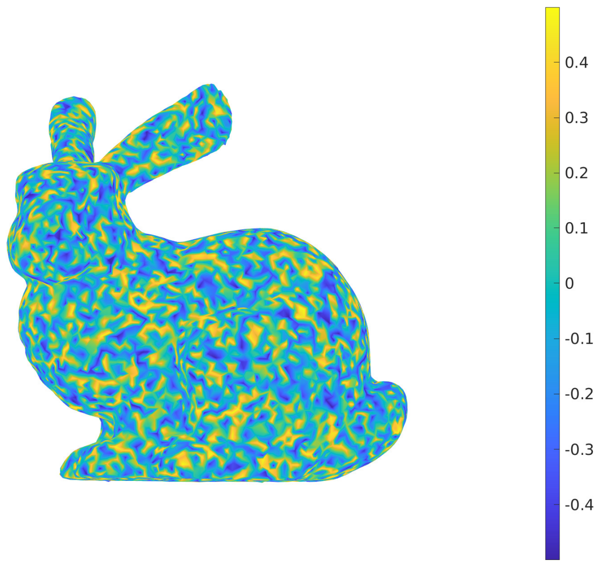

Figure 4.7: The Initial Condition for the System of Reaction Diffusion Equations on the Surface of the Stanford Bunny.

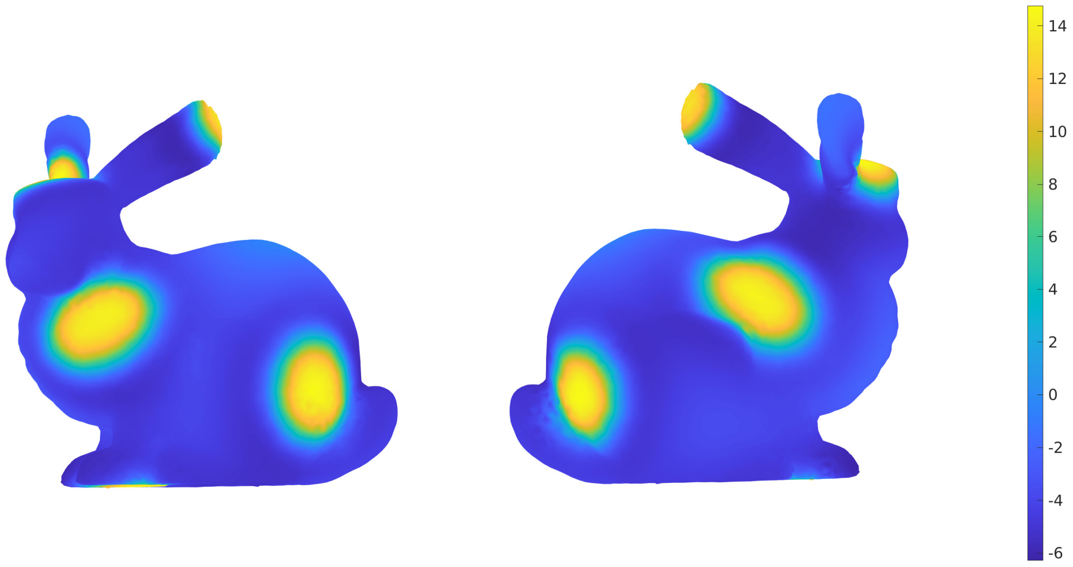

Figure 4.8: The Numerical Solution Using the Parameters Presented in the First Row of Table 4.3 for $t = 300$ and Time Step $\Delta t = 0.1$.

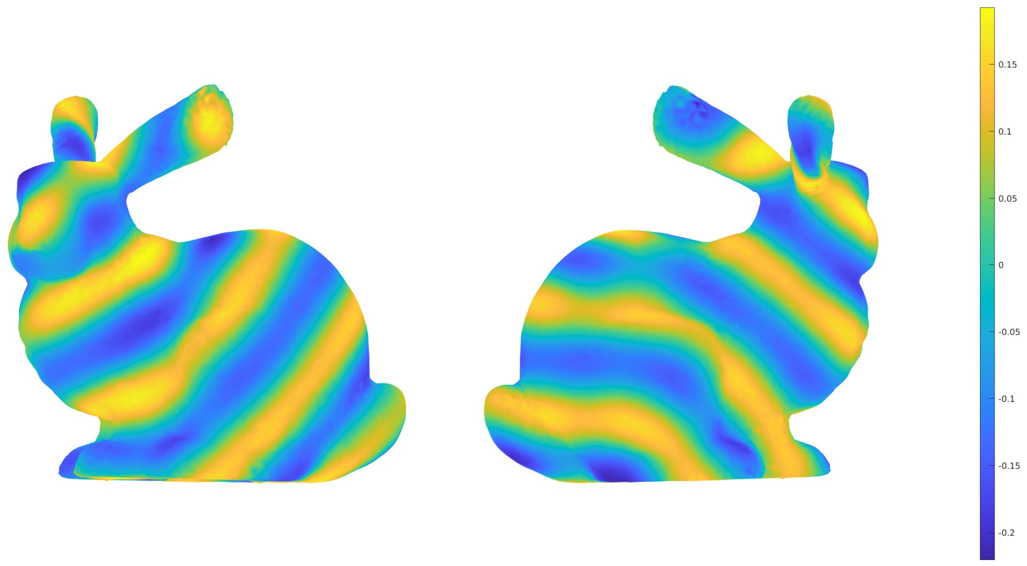

Figure 4.9: The Numerical Solution Using the Parameters Presented in the Second Row of Table 4.3 for $t = 3,000$ and Time Step $\Delta t = 0.1$.

Figure 4.10 shows the eigenvalues of the surface Laplacian operator for the surface of the bunny and we note that all the eigenvalues have negative real parts.

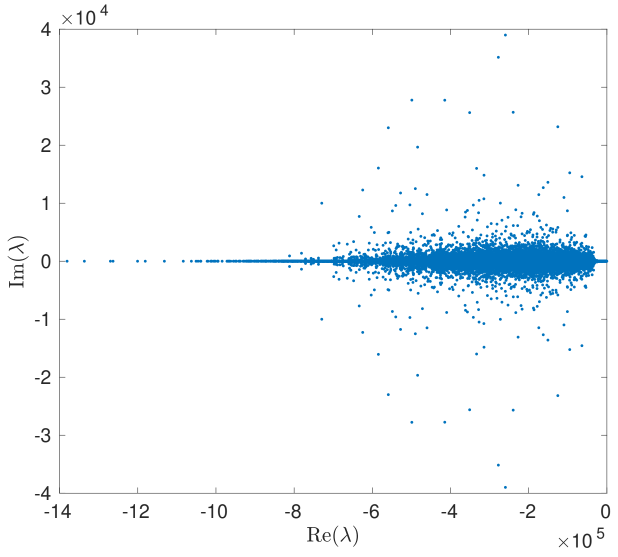

Figure 4.10: Eigenvalues for the Laplacian Operator for the RBF-PU Method on the Surface of the Stanford Bunny.

Chapter 5

FINAL REMARKS

5.1 Ongoing Work

The work in this dissertation can be extended in various directions. The main area we plan to focus on in the future is in parallelizing the code used in the radial basis function partition of unity method. One of the key feature of the partition of unity method is the ability to decompose a large surface into multiple subdomains. The radial basis function partition of unity code can be improved to run the computations on each subdomain in parallel and effectively cutting a fraction of the wall time. The main obstacle in accomplishing this is to efficiently communicate between the subdomains when computing the weighted averages.

5.2 Conclusion

This dissertation explores two methods for approximating functions on surfaces and the necessary computations to solve a select variety of PDEs. The novel spectral cubed sphere method is introduced in Chapter 2. This technique uses the partition of unity method to decompose the sphere into six overlapping regions which are projected onto a circumscribed cube. The window Fourier method is then used on the independent regions and averaged to create a global approximation. The speed associated with spectral methods is taken advantage of by using the fast Fourier transform on each of the subdomains. The spectral version of the cubed sphere method developed in this paper achieves machine precision for continuous functions defined on the unit sphere. Four functions are approximated and geometric convergence is observed.

Computations of the surface gradient and divergence are described in order to investigate the numerical solution of some time dependent partial differential equations. The diffusion equation and the transport equation are studied in detail. Two filters are introduced for stability in time. A filter in the spectral domain is used for local filtering and a second filter is defined by adding artificial viscosity to the global approximation. Both the transport and diffusion equation show geometric convergence for the step size in t.

The radial basis function partition of unity method is explored in Chapter 3. This technique uses radial basis functions for the local approximations in the partition of unity method which decomposes the domain and creates a global approximation using a weighted average. The surface gradient for this approach is computed by projecting the gradient onto the surface using normal vectors. The Laplace operator is then computed using the divergence of the surface gradient. The RBF partition of unity method is explored in conjunction with a point node selection process. A column pivoting QR factorization algorithm is used to generate near optimal points on various surfaces. The accuracy of the RBF-PU method is demonstrated with two applications. The RBF-PU method is used to solve the diffusion equation on the unit sphere as well as a system of reaction-diffusion equations on the surface of a red blood cell. Geometric convergence of the error is observed for the solution of the diffusion equation on the unit sphere. The eigenvalues of the surface Laplacian operator were presented for both cases and were found to all have have non-positive real parts.

Chapter 4 further explores the radial basis function partition of unity method but concentrates on the point node selection process for applications on intricate surfaces. The column pivoting QR factorization algorithm used for selecting points in Chapter 3 is adapted to be used for large cases. A randomized projection of the basis evaluation matrix is used to implement the column pivoting QR factorization algorithm on a smaller matrix. The speed of the node point selection process is further improved by implementing a preselection process. The preselection process uses the subdomains from the partition of unity method to choose a set of near optimal points on each subdomain which are then concatenated and the randomized QR factorization algorithm is used to select the global set of point nodes. By implementing both techniques, the wall time of the point node selection process was found to be 14 times faster. The RBF partition of unity method with the point node selection speed ups is used to solve the reaction-diffusion equations on the surface of a torus and the surface of the Stanford bunny.

Acknowledgments

We would like to thank Jason Yalim and Nicholas Chmielweski for their modifications to the randomized QR factorization package and for the help in implementing it. We also acknowledge the funding support from NSF grants DMS1502640 and DMS1522639.

REFERENCES

[1] Josefin Ahlkrona and Victor Shcherbakov. A meshfree approach to non-Newtonian free surface ice flow: Application to the Haut Glacier d'Arolla. *Journal of Computational Physics*, 330:633–649, 2017.

[2] Kendall Atkinson and Weimin Han. *Spherical harmonics and approximations on the unit sphere: An introduction*, volume 2044. Springer Science & Business Media, 2012.

[3] Ivo Babuška and Jens M Melenk. The partition of unity method. *International Journal for Numerical Methods in Engineering*, 40(4):727–758, 1997.

[4] Alexander H Barnett, Jeremy Magland, and Ludvig af Klinteberg. A parallel nonuniform fast Fourier transform library based on an exponential of semicircle" kernel. *SIAM Journal on Scientific Computing*, 41(5):C479–C504, 2019.

[5] Len Bos, Stefano De Marchi, Alvise Sommariva, and Marco Vianello. Computing multivariate Fekete and Leja points by numerical linear algebra. *SIAM Journal on Numerical Analysis*, 48(5):1984–1999, 2010.

[6] LP Bos and Ulrike Maier. On the asymptotics of Fekete-type points for univariate radial basis interpolation. *Journal of Approximation Theory*, 119(2):252–270, 2002.

[7] John P Boyd. Asymptotic Fourier coefficients for a C^∞ bell (smoothed-"top-hat") & the Fourier extension problem. *Journal of Scientific Computing*, 29(1):1–24, 2006.

[8] Mark AJ Chaplain, Mahadevan Ganesh, and Ivan G Graham. Spatio-temporal pattern formation on spherical surfaces: numerical simulation and application to solid tumour growth. *Journal of mathematical biology*, 42(5):387–423, 2001.

[9] Hyeong-Bin Cheong. Application of double Fourier series to the shallow-water equations on a sphere. *Journal of Computational Physics*, 165(1):261–287, 2000.

[10] Jean Coiffier. *Fundamentals of numerical weather prediction.* Cambridge University Press, 2011.

[11] Feng Dai and Yuan Xu. *Approximation theory and harmonic analysis on spheres and balls*, volume 23. Springer, 2013.

[12] Kathryn P Drake, Edward J Fuselier, and Grady B Wright. A partition of unity method for divergence-free or curl-free radial basis function approximation. *SIAM Journal on Scientific Computing*, 43(3):A1950–A1974, 2021.

[13] Gerhard Dziuk. Finite elements for the Beltrami operator on arbitrary surfaces. In *Partial differential equations and calculus of variations*, pages 142–155. Springer, 1988.

[14] Gerhard Dziuk and Charles M Elliott. Surface finite elements for parabolic equations. *Journal of Computational Mathematics*, pages 385–407, 2007.

[15] Gregory E Fasshauer. *Meshfree approximation methods with MATLAB*, volume 6. World Scientific, 2007.

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

[17] Bengt Fornberg, Elisabeth Larsson, and Natasha Flyer. Stable computations with Gaussian radial basis functions. *SIAM Journal on Scientific Computing*, 33(2):869–892, 2011.

[18] Bengt Fornberg and Cécile Piret. A stable algorithm for flat radial basis functions on a sphere. *SIAM Journal on Scientific Computing*, 30(1):60–80, 2008.

[19] Bengt Fornberg and Grady Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. *Computers & Mathematics with Applications*, 48(5-6):853–867, 2004.

[20] Matteo Frigo and Steven G Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[21] Edward J Fuselier and Grady B Wright. A high-order kernel method for diffusion and reaction-diffusion equations on surfaces. *Journal of Scientific Computing*, 56(3):535–565, 2013.

[22] Alfred Gray, Elsa Abbena, and Simon Salamon. *Modern differential geometry of curves and surfaces with Mathematica®*. Chapman and Hall/CRC, 2017.

[23] Alfa Heryudono, Elisabeth Larsson, Alison Ramage, and Lina von Sydow. Preconditioning for radial basis function partition of unity methods. *Journal of Scientific Computing*, 67(3):1089–1109, 2016.

[24] Jan S Hesthaven, Sigal Gottlieb, and David Gottlieb. *Spectral methods for time-dependent problems*, volume 21. Cambridge University Press, 2007.

[25] Armin Iske. *Multiresolution methods in scattered data modelling*, volume 37. Springer Science & Business Media, 2004.

[26] Anita T Layton and William F Spotz. A semi-Lagrangian double Fourier method for the shallow water equations on the sphere. *Journal of Computational Physics*, 189(1):180–196, 2003.

[27] Damiana Lazzaro and Laura B Montefusco. Radial basis functions for the multivariate interpolation of large scattered data sets. *Journal of Computational and Applied Mathematics*, 140(1-2):521–536, 2002.

[28] Jordan M Martel and Rodrigo B Platte. Stability of radial basis function methods for convection problems on the circle and sphere. *Journal of Scientific Computing*, 69(2):487–505, 2016.

[29] Per-Gunnar Martinsson, Gregorio Quintana OrtI, Nathan Heavner, and Robert van de Geijn. Householder QR factorization with randomization for column pivoting (HQRRP). *SIAM Journal on Scientific Computing*, 39(2):C96–C115, 2017.

[30] Jens M Melenk and Ivo Babuška. The partition of unity finite element method: basic theory and applications. *Computer Methods in Applied Mechanics and Engineering*, 139(1-4):289–314, 1996.

[31] Facundo Mémoli, Guillermo Sapiro, and Paul Thompson. Implicit brain imaging. *NeuroImage*, 23:S179–S188, 2004.

[32] Philip E Merilees. The pseudospectral approximation applied to the shallow water equations on a sphere. *Atmosphere*, 11(1):13–20, 1973.

[33] Francis J Narcowich, Joseph D Ward, and Grady B Wright. Divergence-free RBFs on surfaces. *Journal of Fourier Analysis and Applications*, 13(6):643–663, 2007.

[34] Cécile Piret. The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces. *Journal of Computational Physics*, 231(14):4662–4675, 2012.

[35] Rodrigo B Platte. A windowed Fourier method for approximation of non-periodic functions on equispaced nodes. In *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2014*, pages 405–413. Springer, 2015.

[36] Rodrigo B Platte and Anne Gelb. A hybrid Fourier–Chebyshev method for partial differential equations. *Journal of Scientific Computing*, 39(2):244–264, 2009.

[37] Corrado Ronchi, Roberto Iacono, and Pier S Paolucci. The "cubed sphere": a new method for the solution of partial differential equations in spherical geometry. *Journal of Computational Physics*, 124(1):93–114, 1996.

[38] Ali Safdari-Vaighani, Alfa Heryudono, and Elisabeth Larsson. A radial basis function partition of unity collocation method for convection–diffusion equations arising in financial applications. *Journal of Scientific Computing*, 64(2):341–367, 2015.

[39] Robert Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3(3):251–264, 1995.

[40] Varun Shankar, Grady B Wright, Robert M Kirby, and Aaron L Fogelson. A radial basis function (RBF)-finite difference (FD) method for diffusion and reaction–diffusion equations on surfaces. *Journal of Scientific Computing*, 63(3):745–768, 2015.

[41] Alvise Sommariva and Marco Vianello. Computing approximate Fekete points by QR factorizations of Vandermonde matrices. *Computers & Mathematics with Applications*, 57(8):1324–1336, 2009.

[42] Alex Townsend, Heather Wilber, and Grady B Wright. Computing with functions in spherical and polar geometries I. the sphere. *SIAM Journal on Scientific Computing*, 38(4):C403–C425, 2016.

[43] Lloyd N Trefethen. *Spectral methods in MATLAB*. SIAM, 2000.

[44] Alan Mathison Turing. The chemical basis of morphogenesis. *Bulletin of mathematical biology*, 52(1):153–197, 1990.

[45] Greg Turk. Generating textures on arbitrary surfaces using reaction-diffusion. *Acm Siggraph Computer Graphics*, 25(4):289–298, 1991.

[46] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1):389–396, 1995.

[47] Holger Wendland. Fast evaluation of radial basis functions: Methods based on partition of unity. In *Approximation Theory X: Wavelets, Splines, and Applications*. Citeseer, 2002.

[48] Andrew Witkin and Michael Kass. Reaction-diffusion textures. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, pages 299–308, 1991.

APPENDIX A

LINKS TO CODE

This appendix provides details for the code used for both the spectral cubed sphere method presented in Chapter 2 of this dissertation and the radial basis function partition of unity method found in Chapters 3 and 4.

The software for the spectral cubed sphere method can be downloaded from `https://github.com/gjislas/CS-PUM.git`. Note that the FFTW3 [20] and FINUFFT [4] dependencies are required. Additionally, the codes for certain numerical experiments can be found in the "experiments" folder. In particular, the codes for the convergence results found in Figure 2.7 and the numerical solution for the transport equation on the sphere are available for reproducibility.

A sample of the code used for the radial basis function partition of unity method can be found on `https://github.com/gjislas/RBF-PUM.git` which requires installation of the HQRRP package [29]. The code available can be used to replicate the results for the reaction diffusion equations on the surface of the Stanford bunny presented in Chapter 4 of this dissertation.