A RADIAL BASIS FUNCTION PARTITION OF UNITY METHOD FOR TRANSPORT ON THE SPHERE

by

Kevin Aiton

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Mathematics

Boise State University

May 2014

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Kevin Aiton

Thesis Title: A Radial Basis Function Partition of Unity Method for Transport on the Sphere

Date of Final Oral Examination: 06 December 2013

The following individuals read and discussed the thesis submitted by student Kevin Aiton, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Grady Wright, Ph.D.	Chair, Supervisory Committee
Donna Calhoun, Ph.D.	Member, Supervisory Committee
Inanc Senocak, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Grady Wright, Ph.D., Chair of the Supervisory Committee. The thesis was approved for the Graduate College by John R. Pelton, Ph.D., Dean of the Graduate College.

ACKNOWLEDGMENTS

This work was supported, in part, by the National Science Foundation (NSF) under grant DMS-0934581 and also by the Boise State Mathematics Department under the 2013 summer graduate fellowship.

I would like to express gratitude to Professor Grady Wright. He has been both kind and patient. I would also like to thank the MOOSE development team at Idaho National Laboratory for letting me work remotely so I could work on my thesis. Finally, I would like to give gratitude to my family. Without their support, I would not have been able to finish this thesis.

ABSTRACT

The transport phenomena dominates geophysical fluid motions on all scales making the numerical solution of the transport problem fundamentally important for the overall accuracy of any fluid solver. In this thesis, we describe a new high-order, computationally efficient method for numerically solving the transport equation on the sphere. This method combines radial basis functions (RBFs) and a partition of unity method (PUM). The method is mesh-free, allowing near optimal discretization of the surface of the sphere, and is free of any coordinate singularities. The basic idea of the method is to start with a set of nodes that are quasi-uniformly distributed on the sphere. Next, the surface of the sphere is partitioned into overlapping spherical caps so that each cap contains roughly the same number of nodes. All spatial derivatives of the PDE are approximated locally within the caps using RBFs. The approximations from each cap are then aggregated into one global approximation of the spatial derivatives using an appropriate weight function in the PUM. Finally, we use a method-of-lines approach to advance the system in time. We analyze the computational complexity of this method as compared to global methods based on RBFs and present results for several well-known test cases that probe the suitability of numerical methods for modeling transport in spherical geometries. We conclude with possible future directions of the work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GA – Gaussian

IMQ – Inverse multiquadric

PDE – Partial Differential Equation

RBF – Radial Basis Function

RBF-PUM – RBF-Partition of Unity Method

RK4 – Standard fourth-order Runge Kutta method

LIST OF SYMBOLS

$g\big\vert_X$	For a real valued function $g : D \to \mathbb{R}$ and a finite set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in D$, $g\big\vert_X = [g(\mathbf{x}_1), g(\mathbf{x}_2), \ldots, g(\mathbf{x}_n)]^T$.
μ	Hyperviscosity parameter
Δt	Time-step
∇	Gradient operator in Cartesian coordinates
$\mathbb{S}^2$	The unit sphere in three dimensions.

CHAPTER 1

INTRODUCTION

Mathematical modeling of climate and weather often requires the numerical solution of partial differential equations (PDEs) on the surface of the sphere. Several challenges arise in solving these problems. First while these PDEs can often naturally be parameterized in spherical coordinates, this can't be done if the problem is intended to be solved over the entire sphere without severe unphysical distortions near the poles. This is because any two dimensional coordinate system on the sphere will have at least one singularitiy. Second, these coordinate singularities manifest themselves as apparent singularities in the differential operators of the PDE, complicating numerical discretizations near the singularities. Third, the geometry of the sphere makes it difficult to produce a regular grid or mesh that covers the sphere, as is required for most numerical methods for PDEs. Even popular mappings like the cubed sphere (mapping the faces of an inscribed cube to the sphere) introduce irregularities or distortions in the grid that can effect numerical solutions.

Numerical methods based on Radial Basis Functions (RBFs) provide a promising path to avoiding these issues [11, 12]. These methods can easily be expressed in Cartesian coordinates allowing the PDEs to be solved directly on the sphere without any coordinate singularities or singularities in the differential operator. They also do not require a grid, hence nodes can be placed "optimally" on the sphere. Finally,

they provide high-orders of accuracy (exponential or spectral) for smooth solutions. The downside of these methods is that the computational complexity can be high scaling like $\mathcal{O}(N^2)$ for the global RBF method, where N is the number of degrees of freedom.

In this thesis, we develop a novel RBF method that reduces the computational complexity to $\mathcal{O}(N)$, while still resulting in high orders of accuracy. We apply this method to the numerical solution of the transport equation on the sphere. Transport processes dominate geophysical fluid motions on all scales, so this is the first PDE that new numerical methods for modeling climate and weather are tested on.

Below we briefly introduce RBF interpolation, as it is a major ingredient to our new method. We then discuss the global RBF approach to solving the transport equation from (1.6) since our new method follows a similar approach.

1.1 Radial Basis Function Interpolation

RBFs can be used for interpolating data $\{f_i\}_{i=1}^N \subset \mathbb{R}$ sampled on a finite set of nodes $X = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^d$. The RBF interpolant $s(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ is of the form

$$s(\mathbf{x}) = \sum_{i=1}^{N} c_i \phi(\|\mathbf{x} - \mathbf{x}_i\|), \tag{1.1}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is some radial kernel and $\|\cdot\|$ represents the Euclidian or two-norm. The coefficients $\{c_i\}_{i=1}^N$ are determined by requiring $s(\mathbf{x}_i) = f_i$, which can be expressed as the solution to the following system

$$\underbrace{\begin{bmatrix} \phi(\|\mathbf{x}_1-\mathbf{x}_1\|) & \phi(\|\mathbf{x}_1-\mathbf{x}_2\|) & \dots & \phi(\|\mathbf{x}_1-\mathbf{x}_N\|) \\ \phi(\|\mathbf{x}_2-\mathbf{x}_1\|) & \phi(\|\mathbf{x}_2-\mathbf{x}_2\|) & \dots & \phi(\|\mathbf{x}_2-\mathbf{x}_N\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_N-\mathbf{x}_1\|) & \phi(\|\mathbf{x}_N-\mathbf{x}_2\|) & \dots & \phi(\|\mathbf{x}_N-\mathbf{x}_N\|) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}}_{\underline{c}} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}}_{\underline{f}}. \tag{1.2}$$

We call A the interpolation matrix. For certain types of radial kernels (such as the first three entries of Table 1.1), A is positive definite provided that the nodes are distinct [3]. Thus, existence of a unique interpolant is guaranteed and the method is well-posed. Considering that the interpolant is only dependent on the Euclidian distance between the nodes the interpolant can easily be used for interpolating scattered data in arbitrary dimensions, and on submanifolds of $\mathbb{R}^d$ such as the unit sphere $\mathbb{S}^2$ [19].

Table 1.1: Examples of commonly used radial kernels. Here $\varepsilon > 0$ is called the shape parameter.

Name	$\phi(r)$
Gaussian (GA)	$e^{-(\varepsilon r)^2}$
Laguerre-Gaussian (LGA)	$\left(\frac{5}{2} - (\varepsilon r)^2\right) e^{-(\varepsilon r)^2}$
Inverse Multiquadratics (IMQ)	$\left(1 + (\varepsilon r)^2\right)^{-1/2}$
Multiquadratics (MQ)	$\left(1 + (\varepsilon r)^2\right)^{1/2}$

It has been shown in practice that augmenting the basic RBF interpolant so that it also includes some low order polynomial terms can improve accuracy, especially near domain boundaries [13]. In the case of interpolating on subdomains of $\mathbb{R}^3$, such as patches of the surface of the sphere as we considered in this thesis, this means appending some low order trivariate polynomial terms. For example, the augmented RBF interpolant with an included linear polynomial takes the form

$$s(\mathbf{x}) = \sum_{i=1}^{N} c_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) + \sum_{j=1}^{4} b_j q_j(\mathbf{x}), \tag{1.3}$$

where, for example, $q_1(\mathbf{x}) = 1$, $q_2(\mathbf{x}) = x$, $q_3(\mathbf{x}) = y$ and $q_4(\mathbf{x}) = z$. In addition to interpolation of the data, the following conditions are included for uniquely determining the interpolation coefficients:

$$\sum_{i=1}^{N} c_i q_j(\mathbf{x}_i) = 0, \text{for } j = 1, 2, 3, 4. \tag{1.4}$$

Letting $\underline{b} = [b_1, b_2, b_3, b_4]^T$, and Q be a $N \times 4$ matrix where $Q_{ij} = q_j(\mathbf{x}_i)$. In matrix-vector form, these coefficients can be given as the solution to the linear system:

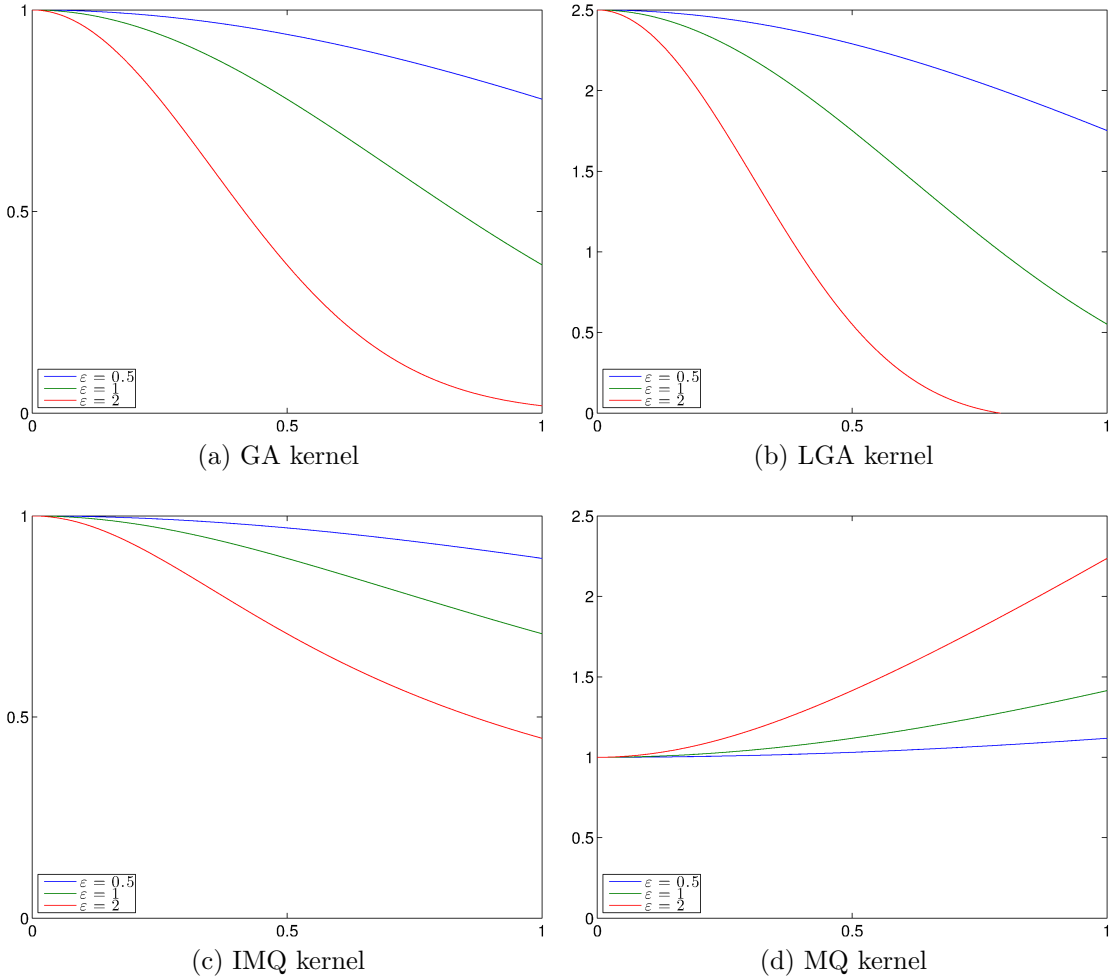
$$\begin{bmatrix} A & Q \\ Q^T & 0 \end{bmatrix} = \begin{bmatrix} \underline{c} \\ \underline{b} \end{bmatrix} = \begin{bmatrix} \underline{f} \\ 0 \end{bmatrix}. \tag{1.5}$$

If the interpolation and nodes lie on the surface of the sphere, but not all on a circle, then this linear system is guaranteed to be non-singular for all radial kernels in Table 1.1 (as well as many others); see [7, 32].

1.2 The Shape Parameter

In this study, we focus on radial kernels that feature a shape parameter ε, such as those in Table 1.1. Decreasing ε increases the flatness of these kernels; this can be seen in Figure 1.1. The value of ε can have a dramatic effect on the accuracy of the resulting interpolant. However, choosing an optimal value is still very much an open question.

In an early study of inverse multiquadric (IMQ) RBF interpolation in $\mathbb{R}^2$, Hardy



(a) GA kernel (b) LGA kernel

(c) IMQ kernel (d) MQ kernel

Figure 1.1: Plots of the radial kernels found in Table 1.1. For each plot, the kernel is ploted for shape parameter values $\varepsilon = 0.5, 1, 2$.

used $\varepsilon = \frac{1}{0.815d}$, where $d = \frac{1}{N}\sum_{i=1}^{N} d_i$ such that d_i is distance between $\mathbf{x}_i$ and its nearest neighbor [22]. In another early study, Franke uses $\varepsilon = \frac{0.8\sqrt{N}}{D}$ where D is the diameter of the smallest circle containing the interpolation nodes [18] . These methods try to balance the accuracy of the interpolant and the conditioning of the interpolation matrix in Eq. 1.2 or 1.5.

Schaback [29] was able to prove that both the accuracy of the RBF interpolant Eq. 1.1 and the condition number of the interpolating matrix A in Eq. 1.2 cannot

both be kept low which initially researchers incorrectly thought implied that there were limits on using small values of the shape parameter ε. The issue with using small values of ε are that shifts of the radial kernels become less distinct so that the columns of A look more alike, which leads to ill-conditioning. This is described as the "*uncertainty relation.*" However, there is a misconception about this uncertainty relation. It does not mean that very high accuracies are impossible to achieve with RBFs, it only means that computing RBF interpolants by solving the linear system Eq. 1.2 cannot be used to achieve very high accuracies.

The first algorithm to bypass the ill-conditioning problem associated with small ε was the Contour-Padé method [17]. Since then several other methods have been developed with the most promising being the RBF-QR techniques [8, 14, 16]. The first RBF-QR method of Fornberg and Pirét [16] demonstrated that as $\varepsilon \to 0$ (the flat limit) RBF interpolants converge to standard spherical harmonic interpolants for approximation on the sphere.

We do not use these stable algorithms in this study. The reason is that our method consists of computing RBF interpolants on a small collection of spherical caps on the sphere as explained in more detail in Section 2.1. At present, methods such as Contour-Padé and RBF-QR break down for interpolation on these types of domains. Research is underway to fix this deficiency, and once complete, will be able to be used directly in our new method.

In this new study, we choose ε so that for the interpolation matrix A, $\text{cond}\,(A) \approx t_{\text{cond}}$, where t_{cond} is a target condition number; this is similar to the first methods by Hardy and Franke in that it balances accuracy and the condition number. This technique means that ε increases as the density of the nodes increases (i.e., the spacing between nodes decreases). However, this can lead to a problem known as "*saturation*

error" [4, 25], which says that if the condition number is held fixed as the density of the the node set increases $(N \to \infty)$, then there is a point at which the error of the interpolant cannot get any lower regardless of increasing N. This is explained in more detail by Maz'ya and Schmidt [25].

1.3 Using RBFs to Solve Partial Differential Equations on the Sphere

The first attempt to solve Partial Differential Equations (PDEs) using RBFs was by Kansa [24]. Using the multiquadric as the kernel, Kansa used RBFs in a collocation approach to solve certain problems from fluid mechanics. Flyer and Wright [11, 12] were the first to apply RBFs to hyperbolic PDEs on the surface of the sphere, including the transport equation and full nonlinear shallow water wave equations. We describe their method for the transport equation since it is similar to the new method we have developed.

The transport of a quantity h on the sphere with no external forcing or body forces is governed by the hyperbolic PDE

$$\frac{\partial}{\partial t} h\left(\mathbf{x}, t\right) = -\mathbf{u}(\mathbf{x}, t) \cdot \left(P \nabla h\left(\mathbf{x}, t\right)\right), \mathbf{x} \in \mathbb{S}^2, t > 0 \tag{1.6}$$

where $\mathbf{u}(\mathbf{x}, t)$ is some incompressible velocity field tangent to the sphere and $P\nabla$ represents the surface gradient operator on the sphere. This operator is written with respect to Cartesian coordinates to avoid singularities that would occur in any two dimensional parameterization of the sphere, such as spherical coordinates. Thus, ∇ is the standard 3D gradient with respect to Cartesian coordinates and P is a linear

operator that projects vectors in $\mathbb{R}^3$ to vectors tangent to the sphere.

The specific construction of P is as follows: let $\mathbf{x} \in \mathbb{S}^2$ and $\mathbf{u} \in \mathbb{R}^3$. If $\mathbf{n}$ is the surface normal of $\mathbb{S}^2$ at $\mathbf{x}$ then $\mathbf{n}\mathbf{n}^T\mathbf{u}$ gives the projection of $\mathbf{u}$ onto $\mathbf{x}$ and $\mathbf{u} - \mathbf{n}\mathbf{n}^T\mathbf{u}$ gives the projection of $\mathbf{u}$ onto the plane tangent to the sphere at $\mathbf{x}$. The surface normal to $\mathbb{S}^2$ at $\mathbf{x}$ is $\mathbf{x}$. Thus, if $\mathbf{x} = (x, y, z)$, then P can be defined as:

$$P = \mathbf{I} - \mathbf{x}\mathbf{x}^T = \begin{bmatrix} (1-x^2) & -xy & -xz \\ -xy & (1-y^2) & -yz \\ -xz & -yz & (1-z^2) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_x^T \\ \mathbf{p}_y^T \\ \mathbf{p}_z^T \end{bmatrix}. \tag{1.7}$$

The standard gradient ∇ can now be constrained to the surface of the sphere:

$$P\nabla = \begin{bmatrix} \mathbf{p}_x \cdot \nabla \\ \mathbf{p}_y \cdot \nabla \\ \mathbf{p}_z \cdot \nabla \end{bmatrix}. \tag{1.8}$$

To solve Eq. 1.6 with the global RBF method, the ∇ operator is first discretized using the RBF interpolant form Eq. 1.1. Let $X = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{S}^2$ be the nodes at which the solution to the PDE will be computed. The partial derivative of Eq. 1.1 with respect to x is

$$\frac{\partial}{\partial x} s(\mathbf{x}) = \sum_{i=1}^{N} c_i \frac{\partial}{\partial x} \phi(\|\mathbf{x} - \mathbf{x}_i\|) \tag{1.9}$$

where the coefficients $\{c_i\}_{i=1}^N$ are determined by the system Eq. 1.2 such that $\underline{c} = A^{-1}\underline{f}$. Since $s(\mathbf{x})$ approximates a function $f(\mathbf{x})$, the partial derivative at $\mathbf{x} = \mathbf{x}_j$ can be approximated as

$$\frac{\partial}{\partial x} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_j} \approx \frac{\partial}{\partial x} s(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_j} = \sum_{i=1}^{N} c_i \frac{\partial}{\partial x} \phi(\|\mathbf{x} - \mathbf{x}_i\|)\big|_{\mathbf{x}=\mathbf{x}_j}. \tag{1.10}$$

By defining a matrix B^x such that $B^x_{ij} = \frac{\partial}{\partial x}\phi\left(\|\mathbf{x}_j - \mathbf{x}_i\|\right)$, then the partial derivative of f at all nodes in X can be approximated as:

$$\frac{\partial}{\partial x} f\big|_X \approx \frac{\partial}{\partial x} s\big|_X = B^x \underline{c} = \underbrace{B^x A^{-1}}_{D^x} \underline{f} \tag{1.11}$$

where we have used the shorthand notation $g\big|_X = \left[g\left(\mathbf{x}_1\right), g\left(\mathbf{x}_2\right), \ldots, g\left(\mathbf{x}_n\right)\right]^T$. Here D^x is referred to as a differentiation matrix. Matrices that compute the partial derivatives with respect to y and z of Eq. 1.1 at X (D^y and D^z) can be constructed in a similar way.

Let $\mathbf{x}_i = (x_i, y_i, z_i) \in X$ and let $\underline{x} = [x_1, x_2, \ldots, x_N]^T$, $\underline{y} = [y_1, y_2, \ldots, y_N]^T$, and $\underline{z} = [z_1, z_2, \ldots, z_N]^T$. Now each component of $P\nabla$ can be approximated at X using the differentiation matrices D^x, D^y, D^z as follows:

$$\begin{aligned}
G^x &:= \operatorname{diag}\left(1 - \underline{x} \circ \underline{x}\right) D^x + \operatorname{diag}\left(-\underline{x} \circ \underline{y}\right) D^y + \operatorname{diag}\left(-\underline{x} \circ \underline{z}\right) D^z \\
G^y &:= \operatorname{diag}\left(-\underline{x} \circ \underline{y}\right) D^x + \operatorname{diag}\left(1 - \underline{y} \circ \underline{y}\right) D^y + \operatorname{diag}\left(-\underline{y} \circ \underline{z}\right) D^z \\
G^z &:= \operatorname{diag}\left(-\underline{x} \circ \underline{z}\right) D^x + \operatorname{diag}\left(-\underline{y} \circ \underline{z}\right) D^y + \operatorname{diag}\left(1 - \underline{z} \circ \underline{z}\right) D^z
\end{aligned} \tag{1.12}$$

where $\circ$ is the Hadamard product, or element wise multiplication operator. Finally, letting $\underline{u}, \underline{v}, \underline{w}$ represents the components of $\mathbf{u}$ sampled at X, a linear operator D can be constructed to approximate the differential operator on the right hand side of Eq. 1.6 as follows:

$$D = \operatorname{diag}(\underline{u}) G^x + \operatorname{diag}(\underline{v}) G^y + \operatorname{diag}(\underline{w}) G^z. \tag{1.13}$$

This process would be similar if the interpolant from Eq. 1.4 was used, where the coefficients are determined instead by the system Eq. 1.5.

The differentiation matrix D is then used in a method-of-lines (MOL) approach for approximating the solution to Eq. 1.6. In this method, the initial condition $h(\mathbf{x}, 0)$ at X, $h_0 = h[(\mathbf{x}_1, 0), h(\mathbf{x}_2, 0), \ldots, h(\mathbf{x}_n, 0)]^T$ and the spatial derivatives in the RHS of Eq. 1.6 are replaced by D, leading to the semi-discrete system:

$$\frac{d}{dt}\underline{h} = -D\underline{h}. \tag{1.14}$$

This system can then be advanced in time using a standard ODE solver such as the classical fourth-order Runge-Kutta method (RK4).

The global RBF method has several desirable features. First, since this method uses RBF interpolation it does not depend on a mesh (or it's *mesh-free*). Thus, the nodes can be distributed in an optimal way to "uniformly" cover the sphere. Additionally, the method uses Cartesian coordinates and is therefore free of coordinate singularities. Lastly, this method compares favorably to other spectral methods in terms of accuracy per degree-of-freedom and time step that can be used for stable time integration [11, 12].

There are, however, limitations with this method. First, the time complexity of constructing the matrix D in Eq. 1.13 is $\mathcal{O}(N^3)$, where N is the number of nodes. The space complexity of D is $\mathcal{O}(N^2)$ since D is dense. Finally, each matrix multiplication with D required for time integration of Eq. 1.14 has a time complexity of $\mathcal{O}(N^2)$. Thus, this method is not practical for large N as is required in realistic simulations of atmospheric flows.

1.4 Overview of Thesis

In this thesis, we introduce the Radial Basis Function Partition of Unity Method (RBF-PUM) to address the computational complexity issues with global RBF method [11, 12]. The basic idea is to first distribute a set of nodes quasi-uniformly over the surface of the sphere as done in the global method. Next, the surface of the sphere is partitioned into overlapping spherical caps so that each cap contains roughly the same number of n nodes. All spatial derivatives of the PDE are approximated locally within the caps using the standard RBF method. The approximations from each cap are then aggregated into one global approximation of the spatial derivatives using an appropriate weight function in the PUM.

The time-complexity associated with constructing a differentiation matrix (1.13) with this new method is reduced to $\mathcal{O}(N \log N)$, while each multiplication of D by a vector has a time complexity of $\mathcal{O}(N)$. Thus, the computational cost scales linearly with N for each time-step of the time integration. The accuracy of this new method no longer exhibits an exponential convergence rate, but it still provides very high (near exponential) accuracy for smooth initial conditions. Additionally, the new method remains mesh-free, and free of any coordinate singularities.

In Chapter 2, the RBF-PUM method is introduced. Section 2.1 gives details on the construction of the RBF-PUM method. In Section 2.2, we show how the RBF-PUM can be parameterized with respect to the number of nodes N, number of nodes per spherical cap n, and the average number of spherical caps a node belongs to q. Details about computing the RBF-PUM matrices are given in Section 2.3. Section 2.4 gives details about the time complexity of constructing the RBF-PUM differentiation matrices, while in Section 2.5 we analyze the sparsity of these matrices. Finally, in

Section 2.6, we show how the RBF-PUM can be stabilized for time-integration.

In Chapter 3, we test the RBF-PUM method on a set of standard test problems from the literature. We analyze results from the cosine bell test [33], deformational flow test [28] with non-smooth cosine bell and Gaussian bell initial conditions and finally stationary vortex roll-up test [27]. These numerical results demonstrate that the new method exhibits near exponential convergence for smooth solutions. Lastly, we provide analysis based on the numerical data.

We conclude in Chapter 4 with comments on future directions of the work.

CHAPTER 2

RBF PARTITION OF UNITY METHOD (RBF-PUM)

In the RBF-PUM method, local interpolants are constructed on subsets (or patches) of $\mathbb{S}^2$ and the combined using weight functions $\{w_i\}$ that form a partition of unity. The method was first introduced by Cavoretto and DeRossi in [5] for interpolation problems on the sphere. Below we present the method first as an interpolation technique then describe how it can be used to approximate spatial derivatives on the sphere. We start the discussion with a description of how the sphere is partitioned and the weight functions are constructed. We follow this by showing how the differentiation matrix D from Section 1.3 can be constructed using the RBF-PUM method. Next, we analyze the time complexity of constructing D and its sparsity. Finally, we show how D can be stabilized via hyperviscosity.

2.1 Constructing the RBF Partition of Unity Interpolant

A partition of unity can be defined as follows [31]:

Definition 1. *A **partition of unity** on a topological space S is a family $\{w_i\}$ of continuous functions*

$$w_i : S \to \mathbb{R}_+$$

such that:

1. $\{\operatorname{supp} w_i\}$ *is a locally finite covering of* S,

2. $\forall \mathbf{x} \in S : \sum_i w_i(\mathbf{x}) = 1$.

In our application, S is the unit sphere $\mathbb{S}^2$ and the details on constructing the partition of unity are as follows:

Let $X = \{\mathbf{x}_i\}_{i=1}^N$ be a set of scattered nodes on $\mathbb{S}^2$ and $\Omega_1, \Omega_2, \ldots, \Omega_M$ be a set of distinct spherical caps on $\mathbb{S}^2$ such that

1. $\bigcup_{i=0}^{M} \Omega_i = \mathbb{S}^2$ i.e. the caps cover the surface of sphere; and
2. Each cap contains at least one node in X.

We refer to the spherical caps as patches. For each patch Ω_k, define $\boldsymbol{\xi}_k \in \mathbb{S}^2$ as the center of patch Ω_k and ρ_k as the radius of the patch, measured as the Euclidean distance from $\boldsymbol{\xi}_k$. For each patch, we define a continuous compactly supported weight function ψ_k on Ω_k as follows:

$$\psi_k(\mathbf{x}) = \psi\left(\frac{\|\mathbf{x} - \boldsymbol{\xi}_k\|}{\rho_k}\right), \tag{2.1}$$

where ψ has compact support over the interval $[0, 1)$. In this study, we use the cubic B-spline

$$\psi(r) = \begin{cases} \frac{2}{3} + 4\,(r-1)\,r^2 & \text{if } 0 \le r < \frac{1}{2}, \\ -\frac{4}{3}(r-1)^3 & \text{if } \frac{1}{2} < r \le 1, \\ 0 & \text{if } r > 1. \end{cases} \tag{2.2}$$

which has two continuous derivatives on $[0,1]$. Each ψ_k has compact support on Ω_k. We define w_k as follows:

$$w_k(\mathbf{x}) = \frac{\psi_k(\mathbf{x})}{\sum_{i=0}^{M} \psi_i(\mathbf{x})}. \tag{2.3}$$

Since each w_k has compact support on Ω_k, and $\forall \mathbf{x} \in \mathbb{S}^2 \sum_{k=0}^{M} w_k(\mathbf{x}) \equiv 1$, $\{w_k\}$ forms a partition of unity on $\mathbb{S}^2$.

These weight functions are used in the RBF-PUM interpolant as follows. For each Ω_k, define X_k as the set of $\mathbf{x} \in X$ such that $\mathbf{x} \in \Omega_k$. Let s_k be the global RBF interpolant from Section 1.1, either Eq. 1.1 or 1.3, defined on the nodes X_k. The RBF partition of unity interpolant is given

$$s(\mathbf{x}) = \sum_{k=0}^{M} w_k(\mathbf{x})\, s_k(\mathbf{x}). \tag{2.4}$$

Suppose $\{f_i\}_{i=1}^{N} \subset \mathbb{R}$ is data sampled at X, and let $\mathbf{x}_i \in X$. We know from Section 1.1 that if $\mathbf{x} \in \Omega_k$, then $s_k(\mathbf{x}_i) = f_i$. By construction, if $\mathbf{x}_i \notin \Omega_k$, then $w_k(\mathbf{x}_i) = 0$ (since the weight functions have compact support over their associated patches). Thus,

$$\sum_{i=0}^{M} w_k(\mathbf{x}_i) = \sum_{\Omega_k \ni \mathbf{x}_i} w_k(\mathbf{x}_i) \equiv 1, \tag{2.5}$$

which implies

$$s(\mathbf{x}_i) = \sum_{i=0}^{M} w_k(\mathbf{x}_i)\, s_k(\mathbf{x}_i) = \sum_{\Omega_k \ni \mathbf{x}_i} w_k(\mathbf{x}_i)\, f_i = f_i \sum_{\Omega_k \ni \mathbf{x}_i} w_k(\mathbf{x}_i) = f_i, \tag{2.6}$$

or that $s(\mathbf{x})$ interpolates $\{f_i\}_{i=1}^{N}$ over X.

2.2 Choosing the Nodes and Patches

Since the interpolant of (2.4) does not require the nodes X to be on a grid or mesh, we are free to choose them however we wish for our application. In this study, we focus on node sets that are quasi-uniformly distributed on the sphere so as to get near optimal resolution of the entire sphere. Since there exists no equidistant node sets on the sphere where the number of nodes is greater than 20, there are several techniques for generating quasi-uniform points on the sphere. We use the maximum determinant (MD) method for generating these nodes [35]. The MD method choses the nodes in such a way that the determinant of an interpolating matrix that depends on spherical harmonics is maximized. These node sets have been generated for various number of nodes and can be freely downloaded from [30]. For patch centers, we use the minimum energy (ME) points. These node sets are computed by minimizing the Reisz energy (with a power of 2) of the node set over the sphere [21]. For information on how we generated the patch centers, see Appendix A.

Given a MD node set, we determine the patches based on two criteria:

1. the approximate number of nodes each patch will contain,

2. the average number of patches a node belongs to.

Suppose there are N nodes and we want approximately n nodes per patch. Because the nodes are quasi-uniformly distributed, we can expect the area per node ratio over the entire sphere to roughly equal the area per node ratio on the patch. If ρ_k is the radius of patch Ω_k, then this area relationship gives

$$\frac{4\pi}{N} \approx \frac{\pi \rho_k^2}{n}. \tag{2.7}$$

Thus, we can approximate ρ_k as

$$\rho_k \approx 2\sqrt{\frac{n}{N}}. \tag{2.8}$$

Since the patch centers are also quasi-uniformly distributed, all patch radii can be chosen the same way, i.e $\rho_k = \rho$, for all k.

For each node $\mathbf{x}_k$, let q_k be the number of patches $\mathbf{x}_k$ belongs to. This can be used as a measure of overlap in that if the values of $\{q_i\}_{i=1}^N$ are high, there is more overlap between the patches. Considering that all patches have radius ρ, the number of patches that $\mathbf{x}_k$ belongs to is the same as the number of patch centers in a spherical cap centered at $\mathbf{x}_k$ with radius ρ. We choose q to represent the average of $\{q_i\}_{i=1}^N$. Suppose there are M patch centers. Since these are quasi-uniformly distributed we can expect that the area per center ratio for the sphere is approximately equal to the area per center ratio for the spherical cap centered at any node, i.e.

$$\frac{4\pi}{M} \approx \frac{\pi\rho^2}{q}. \tag{2.9}$$

We can thus approximate M as

$$M \approx \left\lceil \frac{4q}{\rho^2} \right\rceil \approx \left\lceil q\frac{N}{n} \right\rceil. \tag{2.10}$$

We numerically verify in Appendix B that when we chose M and ρ with (2.8) and (2.10), that the actual number of number nodes per patch and average number of patches a node belongs to corresponds very closely with there respective parameters n and q.

2.3 Approximating the Surface Gradient Operators

We use the RBF-PUM in a similar way to the global RBF method [11, 12] discussed in Section 1.3 to discretize the spatial derivative operators associated with the right hand side of Eq. 1.6. The first thing that needs to be considered is how to construct the discrete operators for the components of the surface gradient. Interpolation in the RBF-PUM occurs at two levels: globally and locally. Locally we use the direct RBF method for each of the patches where globally we use the RBF-PUM on the sphere itself. The partial derivative of the RBF-PUM interpolant in Eq 2.4 with respect to x is

$$\frac{\partial}{\partial x} s\left(\mathbf{x}\right) = \sum_{k=1}^{M} \left[w_k\left(\mathbf{x}\right) \frac{\partial}{\partial x} s_k\left(\mathbf{x}\right) + s_k\left(\mathbf{x}\right) \frac{\partial}{\partial x} w_k\left(\mathbf{x}\right) \right]. \tag{2.11}$$

The equation for the partial derivative of the weight with respect to x is

$$\frac{\partial}{\partial x} w_k\left(\mathbf{x}\right) = \frac{\frac{\partial}{\partial x}\psi_k\left(\mathbf{x}\right) \sum\limits_{i=1}^{m} \psi_i\left(\mathbf{x}\right) - \psi_k\left(\mathbf{x}\right) \sum\limits_{i=1}^{m} \frac{\partial}{\partial x}\psi_i\left(\mathbf{x}\right)}{\left[\sum\limits_{i=1}^{m} \psi_i\left(\mathbf{x}\right) \right]^2}. \tag{2.12}$$

Since $w_k\left(\mathbf{x}\right)$ has compact support on Ω_k, both $s_k\left(\mathbf{x}\right)$ and $\frac{\partial}{\partial x} s_k\left(\mathbf{x}\right)$ only need to be computed for $\mathbf{x} \in X_k$. We compute $\frac{\partial}{\partial x} s_k\big|_{X_k}$ similarly to the direct RBF method described in Section 1.3 with a differentiation matrix D_k^x.

Given that for all k, $\frac{\partial}{\partial x} s_k\big|_{X_k} = D_k^x f\big|_{X_k}$, $s_k\big|_{X_k} = f\big|_{X_k}$, and $w_k\left(\mathbf{x}\right)$ is independent of f, it is possible to construct a differentiation matrix D^x such that

$$\frac{\partial}{\partial x} f\big|_X \approx \frac{\partial}{\partial x} s\big|_X = D^x \underline{f}. \tag{2.13}$$

Differentiation matrices D^y, D^z can similarly be constructed for the spatial derivatives y and z, respectively. The components of the projected gradient G^x, G^y, and G^z from (1.12) can then be approximated in a similar fashion, but by using the RBF-PUM differentiation matrices instead. The differentiation matrix D for the advection operator from Eq. 1.13 can then be computed by replacing the components of the projected gradient from the global RBF method with that of the RBF-PUM. This new D can be used to solve the PDE (1.6) via the method of lines using the classical Runge-Kutta-Method (with some minor modification as discussed in Section 2.6).

2.4 Computational Complexity of Constructing RBF-PUM Differentiation Matrices

One advantage that the RBF-PUM has over the direct RBF method is that the computational complexity for construction is more manageable than the global method. To analyze this complexity, we start with a set of quasi-uniformly distributed nodes $X = \{\mathbf{x}_i\}_{i=1}^N$, a set of quasi-uniformly distributed patch centers $\{\boldsymbol{\xi}_k\}_{k=1}^M$, and a radius for all patches ρ. These are chosen with parameters N, n, and q as specified in Section 2.2.

Suppose that n and q are fixed, and let n_k be the number of nodes in patch Ω_k and q_i be the number of patches $\mathbf{x}_i$ belongs to. Even though n and q are fixed, it needs to be shown how n_k and q_i behave asymptotically for large N. We can use Proposition 14.1 from [32], which states that there exists constants c_1 and c_2 such that

$$c_1 N^{-1/3} \leq h_{X,\mathbb{S}^2} \leq c_2 N^{-1/3} \tag{2.14}$$

where $h_{X,\mathbb{S}^2}$ is the fill distance, or the largest radius r such that a ball of this radius centered at $\mathbf{x}_i$ (i.e., $B(\mathbf{x}_i, r)$) contains no other node. The proposition does this by considering the volume of spheres centered at $\mathbf{x}_i$. If we take advantage of the topology of $\mathbb{S}^2$, we can instead use spherical caps to reach this conclusion:

$$c_1 N^{-1/2} \leq h_{X,\mathbb{S}^2} \leq c_2 N^{-1/2}. \tag{2.15}$$

Corollary 14.2 from [32] argues using Proposition 14.1 that for a cube whose side is equal to $2cN^{-1/3}$, the number of number of nodes in that cube is bounded by the constant independent of N regardless of its location; it does this by considering the cube enclosed by a sphere. If we instead use spherical caps, a similar proof can be made to show that if the radius of a spherical cap is bounded by $cN^{-1/2}$, then the number of nodes in the spherical cap is bound by a constant independent of N (regardless of its location on the sphere).

From Section 2.2, we have that ρ is chosen so that $\rho = 2\sqrt{\frac{n}{N}}$. Thus, if n is fixed, there must exist a constant C independent of N such that for all patches Ω_k, $n_k \leq C$. Considering that M linearly depends on N by construction (2.10), it can be inferred that there exists a constant c such that $\rho \leq c\sqrt{\frac{1}{M}}$. Thus, if we consider that if there were spherical caps centered at $\mathbf{x}_i$ (where q_i would be the number of centers in the caps), a similar argument can be made that there exist a constant K independent of M such that $q_i < K$. Since M depends on N though, it must be the case that K is independent of N as well. Our numerical evidence strongly suggests that $n \approx \frac{1}{M}\sum_{i=1}^{M} n_i$ and $q \approx \frac{1}{N}\sum_{i=1}^{N} q_i$ so that we can argue $n_k = \mathcal{O}(n)$ and $q_k = \mathcal{O}(q)$.

To construct the partition of unity, we first must determine for each patch Ω_k the

nodes that belong to it. This can be done efficiently by building a kd-tree using the node points [32]. The time complexity of constructing the tree is $\mathcal{O}(N \log N)$, while the space complexity is $\mathcal{O}(N)$. The time to find the nodes that are ρ away from the patch center $\boldsymbol{\xi}_k$ is $\mathcal{O}(\log N)$, so that the time to determine this for all patches is $\mathcal{O}(M \log N)$. Considering that $M = \mathcal{O}(N)$ by construction (there will never be more patches than nodes), we have that the computational cost of building the kd-tree and determining the patches to be $\mathcal{O}(N \log N)$.

The differentiation matrices D_k^x, D_k^y, and D_k^z must be computed for each patch Ω_k. The computational cost for constructing the differentiation matrices for Ω_k is $\mathcal{O}(n^3)$, making the total time for all patches $\mathcal{O}(Mn^3)$. From Eq. 2.10, it can be inferred that $M = \mathcal{O}\left(q\frac{N}{n}\right)$ so that we have $\mathcal{O}(Nqn^2)$. Computing these matrices is embarrassingly parallel because computing the differentiation matrices for patch Ω_k requires no information from any other patch. Next, the partition of unity weight functions and partial derivative values have to be computed. The time complexity for calculating the partition of unity weights for $\mathbf{x}_i$ is $\mathcal{O}(q)$, making the complexity $\mathcal{O}(Nq)$ for all the nodes.

Finally, we have to assemble the RBF-PUM differentiation matrices D^x,D^y and D^z. Let's consider computing the i_{th} column of D^x, which is the vector of values f_i would be multiplied with for the RBF-PUM partial derivative with respect to x for all nodes. Looking at Eq. 2.11, the only D_k^x that would be used for this column are the D_k^x such that $\mathbf{x}_i \in \Omega_k$; we would also have to use $\frac{\partial}{\partial x} w_k(\mathbf{x})$ for that patches $\mathbf{x}_i$ belongs to. Thus, the number of computations needed to compute this column would be $\mathcal{O}(qn + q) = \mathcal{O}(qn)$. This makes the computational cost of assembling D^x $\mathcal{O}(Nnq)$, since there are N columns of D^x. This is similarly true for D^y and D^z.

The computational cost of constructing the differentiation matrices for each patch,

evaluating the partition of unity weight functions and partial derivative values, and constructing the differentiation matrices for the RBF-PUM is $\mathcal{O}(Nqn^2)$. The total computational cost for construction is therefore $\mathcal{O}(Nqn^2 + N\log N)$. Since $n \ll N$ and $q = \mathcal{O}(1)$, this is a significant savings over the global method, which has a cost of $\mathcal{O}(N^3)$.

2.5 Sparsity of the RBF-PUM Differentation Matrices

It can be shown that the RBF-PUM differentiation matrices are significantly more sparse than the differentiation matrices from the global RBF method in Section 1.3 (which are in fact dense). In this section, we make estimates on the sparsity of the RBF-PUM differentiation matrices and compare these to the values seen in practice.

Let's consider D^x_{ij}, which is the value we multiply f_i by to compute (2.11) at $\mathbf{x}_j$. From Section 2.2, it is clear that $w_k(\mathbf{x}_j) = \frac{\partial}{\partial x} w_k(\mathbf{x}_j) = 0$ if and only if $\mathbf{x}_j \notin \Omega_k$. Also if f_i is used to compute $s_k(\mathbf{x}_j)$ or $\frac{\partial}{\partial x} s_k(\mathbf{x}_j)$, then $\mathbf{x}_i$ is in Ω_k. Thus, if there is no patch Ω_k such that $\mathbf{x}_i$ and $\mathbf{x}_j$ both belong to it, then $D_{ij} = 0$. This means for the i_{th} column of D^x, the number of non zeros is bounded by the number of nodes that are also in a patch with $\mathbf{x}_i$, or $\left|\bigcup_{X_k \ni \mathbf{x}_i} X_k\right|$. This is similarly true for D^y and D^z. Let nnz be the number of non-zero entries in a given RBF-PUM differentiation matrix. Then,

$$nnz \leq \sum_{i=0}^{N} \left| \bigcup_{X_k \ni \mathbf{x}_i} X_k \right| \leq \sum_{i=0}^{N} \sum_{X_k \ni \mathbf{x}_i} n_k = \mathcal{O}(Nnq). \tag{2.16}$$

Since $n \ll N$ and $q = \mathcal{O}(1)$, the matrices have nice sparsity properties.

In part (a) of Tables 2.1-2.3, we display the ratio of the actual nnz of the RBF-PUM differentiation matrices to our estimate Nnq for different n and N with varying q. As we can see from these values, our estimate is a bit pessimistic, with the

constant in front of Nnq being less than one. Additionally, this estimate decreases with increasing q. In part (b) of Tables 2.1-2.3, we display the percent full of the computed RBF-PUM differentiation matrices confirming the nice sparsity properties of those matrices. In Figure 2.1, we illustrate how the patches and sparsity of the differentiation matrices change with increasing q for the case of $N = 4096$ nodes and $n = 100$.

Table 2.1: Ratio of number of non zeros in a differentiation matrix compared to our estimate and percent full the differentiation matrices are with q=3.

(a) Ratio

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	0.71	0.71	0.72	0.72	0.72	0.72	0.72
144	0.71	0.71	0.72	0.72	0.72	0.72	0.72
196	0.71	0.71	0.71	0.72	0.72	0.72	0.72
256	0.71	0.71	0.71	0.71	0.71	0.72	0.72

(b) Percent Full

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	5.22%	3.35%	2.33%	1.72%	1.32%	1.04%	0.84%
144	7.54%	4.82%	3.36%	2.47%	1.89%	1.49%	1.21%
196	10.21%	6.54%	4.56%	3.36%	2.57%	2.03%	1.64%
256	13.23%	8.53%	5.93%	4.37%	3.35%	2.65%	2.15%

Table 2.2: Ratio of number of non zeros in a differentiation matrix compared to our estimate and percent full the differentiation matrices are with q=3.5.

(a) Ratio

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	0.65	0.65	0.65	0.65	0.66	0.65	0.66
144	0.65	0.65	0.65	0.65	0.65	0.65	0.66
196	0.65	0.65	0.65	0.65	0.65	0.65	0.65
256	0.64	0.65	0.65	0.65	0.65	0.65	0.65

(b) Percent Full

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	5.56%	3.57%	2.48%	1.82%	1.40%	1.11%	0.90%
144	7.99%	5.14%	3.58%	2.62%	2.01%	1.59%	1.29%
196	10.85%	6.99%	4.86%	3.57%	2.74%	2.16%	1.75%
256	14.03%	9.09%	6.32%	4.66%	3.57%	2.82%	2.29%

Table 2.3: Ratio of number of non zeros in a differentiation matrix compared to our estimate and percent full the differentiation matrices are with q=4.

(a) Ratio

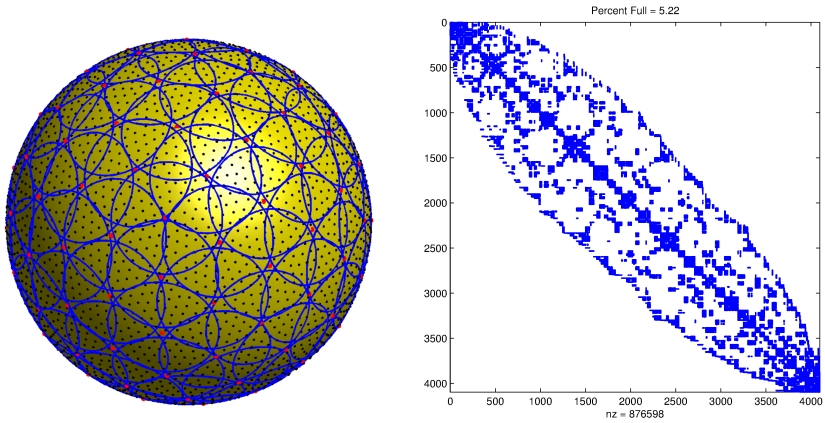
$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	0.60	0.60	0.60	0.60	0.60	0.60	0.60
144	0.60	0.60	0.60	0.60	0.60	0.60	0.60
196	0.59	0.60	0.60	0.60	0.60	0.60	0.60
256	0.59	0.60	0.60	0.60	0.60	0.60	0.60

(b) Percent Full

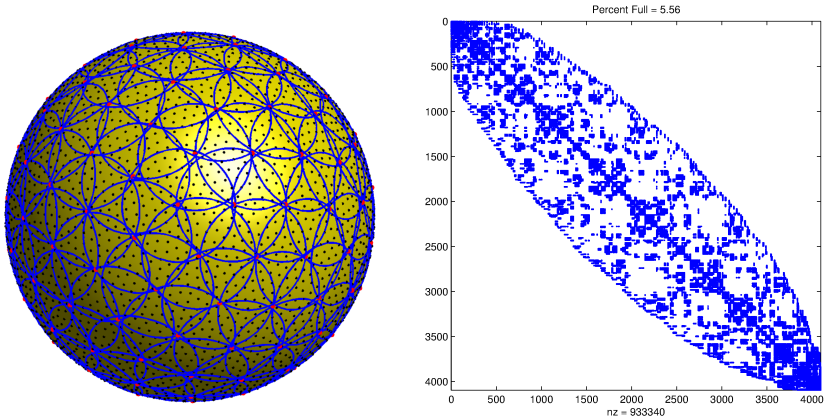
$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	5.84%	3.75%	2.61%	1.91%	1.47%	1.16%	0.94%
144	8.37%	5.38%	3.75%	2.76%	2.11%	1.67%	1.36%
196	11.36%	7.32%	5.10%	3.75%	2.87%	2.27%	1.84%
256	14.72%	9.53%	6.64%	4.89%	3.75%	2.96%	2.40%

Figure 2.1: Illustration of the nodes and patches used in the RBF-PUM for increasing q with parameters $N = 4096$ and $n = 100$. Here the black solid circles reperesent the nodes, blue spherical caps represent the patches, and the red solid circles represent the centers of the patches. The sparsity of the corresponding differentiation matrices is also shown for each q.

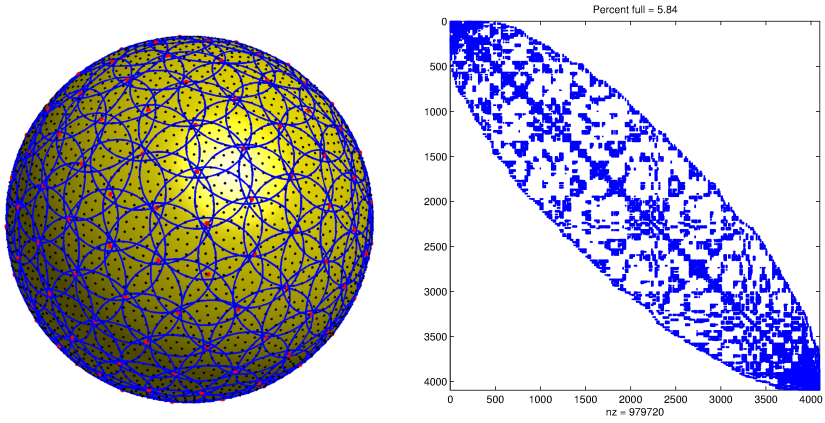
(a) q = 3



(b) q = 3.5



(c) q = 4



2.6 Stabilization via Hyperviscosity

Our semi-discrete or method-of-lines formulation of the transport equation (1.6) takes the form

$$\frac{d}{dt}\underline{h} = -D\underline{h}, \tag{2.17}$$

where D represents the RBF-PUM discretization of the the advection operator $\mathbf{u} \cdot (P\nabla)$ (see Eq. (1.13)). A necessary condition for stability of this formulation is that the eigenvalues of $-D$ must be in the stability domain of the ODE solver used for advancing the system in time. At the very least, this means that all the eigenvalues of $-D$ must be in the left half plane. Since the advection operator contains no natural dissipation term, this is an extraordinary condition to put on the numerical discretization scheme. The RBF-PUM method does not satisfy this requirement and, like many methods for solving hyperbolic PDEs on non-rectangular grids, a numerical "stabilization" term needs to be included to shift the eigenvalues to the left half plane.

A common approach for stabilizing high-order finite-difference and collocation methods is to include a hyperviscosity dissipation term in Eq. 2.17:

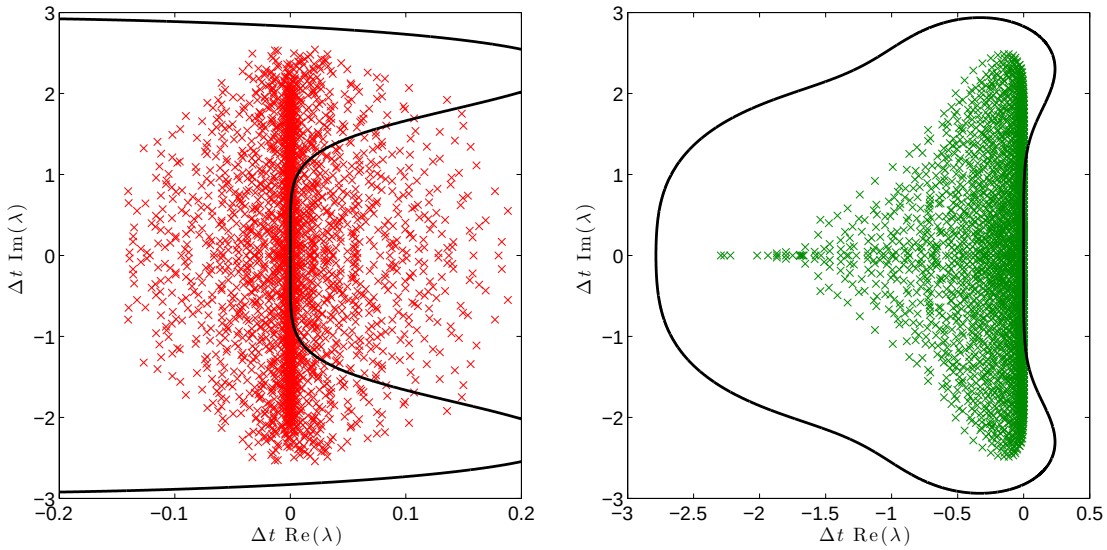
$$\frac{d}{dt}\underline{h} = -(D - \mu H)\underline{h}, \tag{2.18}$$

where H is the numerical discretization of the hyperviscosity term Δ^p, $p \in \mathbb{N}$, and μ is a weighting constant. The goal of this approach is to pick p and μ to stabilize the time integration of the numerical scheme without causing a deterioration in the accuracy of the spatial discretization. Typically, the higher-order the method, the larger the value of p is used so that the dissipation term only damps the highest frequency modes in the solution. This stabilization approach has been applied successfully to

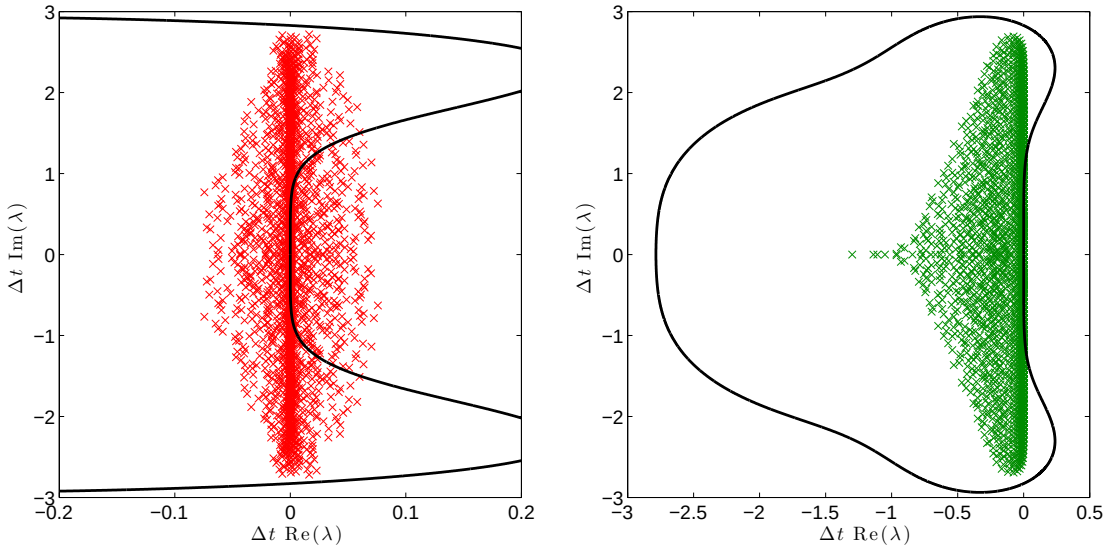
other RBF methods [2, 10, 15].

We adopt a similar approach to stabilizing the RBF-PUM method, however, instead of using a hyperviscosity term of the form Δ^p, we follow the approach proposed in [15] for stabilizing the global RBF method for the transport equation discussed in Section 1.3. In this approach, one uses the inverse of the RBF interpolation matrix A from Eq. 1.2 as H in Eq. 2.18. As argued in [15], the matrix A^{-1} acts like an approximation to a high power of the Laplace-Beltrami operator on the sphere. The issue with using this method directly is that it would require computing A^{-1} based on the whole set of nodes in X, which would require a computational cost of $\mathcal{O}(N^3)$. We instead construct H by first computing inverses of the interpolation matrices on each of the M patches, A_k^{-1}, $k = 1, \ldots, M$. We then combine these inverses using the partition of unity weight functions to get a sparse approximation to the global version of the hyperviscosity matrix A^{-1}. The computational complexity of constructing H is similar to that of constructing the RBF-PUM differentiation matrices D^x, D^y, and D^z, and the sparsity properties of H are identical to these matrices.

To illustrate the effects of the hyperviscosity term, we consider the advection operator corresponding to the velocity field $\mathbf{u} = \begin{bmatrix} 0 & z & -y \end{bmatrix}^T$, which corresponds to solid body rotation of the sphere (this is also the first test case we consider in Chapter 3). In the left column of Figure 2.2, we display the eigenvalues of the RBF-PUM differentiation matrix $-D$ for this advection operator using $N = 4096$ nodes and two values of n. The stability domain of the standard fourth order Runge-Kutta (RK4) method is also plotted in this figure and the eigenvalues have been scaled by $\Delta t = 2\pi/1600$. We can see that the eigenvalues for both values of n are scattered into the left half-plane and outside the stability domain of RK4, so that stable time integration would be impossible. In the right column of this figure, we display the



(a) $n = 100$



(b) $n = 256$

Figure 2.2: Eigenvalues of the RBF-PUM differentiation matrix for the advection operator corresponding to solid body rotation for the case of $N = 4096$ nodes, a target condition number of $t_{\text{cone}} = 10^{12}$, and $q = 4$. The left column shows the (scaled) eigenvalues of $-D$, corresponding to no hyperviscosity (see Eq. 2.17). The right column shows the (scaled) eigenvalues of $-(D - \mu H)$, corresponding to the stabilized differentiation matrix with hyperviscosity (see Eq. 2.18). For both value of n, $\mu = 10^{-8}$. The black curve in all plots corresponds to the stability domain of RK4 and the eigenvalues have been scaled by $\Delta t = 2\pi/1600$.

eigenvalues of the stabilized differentiation matrix $-(D - \mu H)$. We see that for both values of n, all eigenvalues have been shifted to the left-half plane and are contained in the stability domain of RK4. Thus, this version is suitable for stable time integration. In the next chapter, we see that this hyperviscosity stabilization does not have a noticeable effect on the accuracy of the RBF-PUM method.

CHAPTER 3

NUMERICAL RESULTS

In this chapter, we apply the RBF-PUM method to several standard benchmark problems in the literature to analyze its performance. All tests are for the transport equation:

$$\frac{\partial}{\partial t}h\left(\mathbf{x},t\right)+\mathbf{u}\left(\mathbf{x},t\right)\cdot P\nabla h\left(\mathbf{x},t\right)=0, \tag{3.1}$$

where $P\nabla$ is the surface gradient and $\mathbf{u}$ is tangent to the sphere. In some of these tests, $\mathbf{u}$ can be defined by a stream function $\psi\left(\mathbf{x},t\right)$

$$\mathbf{u}=\mathbf{x}\times\nabla\psi \tag{3.2}$$

where possible, we will state these tests in terms of ψ.

For all of the tests, we present results for two kernels: the Gaussian (GA) and inverse multiquadric (IMQ) listed in Table 1.1. We compute solutions for increasing values of N and n and analyze the convergence of the method. We test with values $N = 4096$, 6400, 9216, 12544, 16384, 20736, and 25600; $n = 100$, 144, 196, 256; and $q = 4$. The errors between the approximate and true solutions are computed using the l_2 and l_∞ norms. The convergence is measured as a function of $\sqrt{N}$ since the spacing between the nodes decreases asymptotically like $1/\sqrt{N}$, which follows from the fact that the nodes are quasi-uniformly distributed on the sphere. We compare

the results for two target condition numbers t_{cond} to examine the effect of saturation errors. For each test, the hyperviscosity parameter μ is fixed for all N and n. The standard fourth-order Runge-Kutta method (RK4) is used for the time integration with the time step Δt fixed (and not optimized) for each test. Finally, all tests were performed using MATLAB 2012b.

3.1 Cosine Bell Test

As a first test problem we will consider the standard Test Case 1 from Williamson et al. [33]. For this problem, the initial height field is the following cosine bell:

$$h(\mathbf{x}) = \begin{cases} \frac{1}{2}\left(1 + \cos\left(3\pi r\left(\mathbf{x}\right)\right)\right) & r\left(\mathbf{x}\right) < \frac{1}{3} \\ \\ 0 & r\left(\mathbf{x}\right) \geq \frac{1}{3} \end{cases} \tag{3.3}$$

where $r(\mathbf{x}) = \arccos(x)$, and $\mathbf{x} = (x, y, z)$. This initial condition has a jump in the second derivative at the support of the bell, which makes the test susceptible to both diffusive and dispersive errors. The stream function is time-independent and is given by

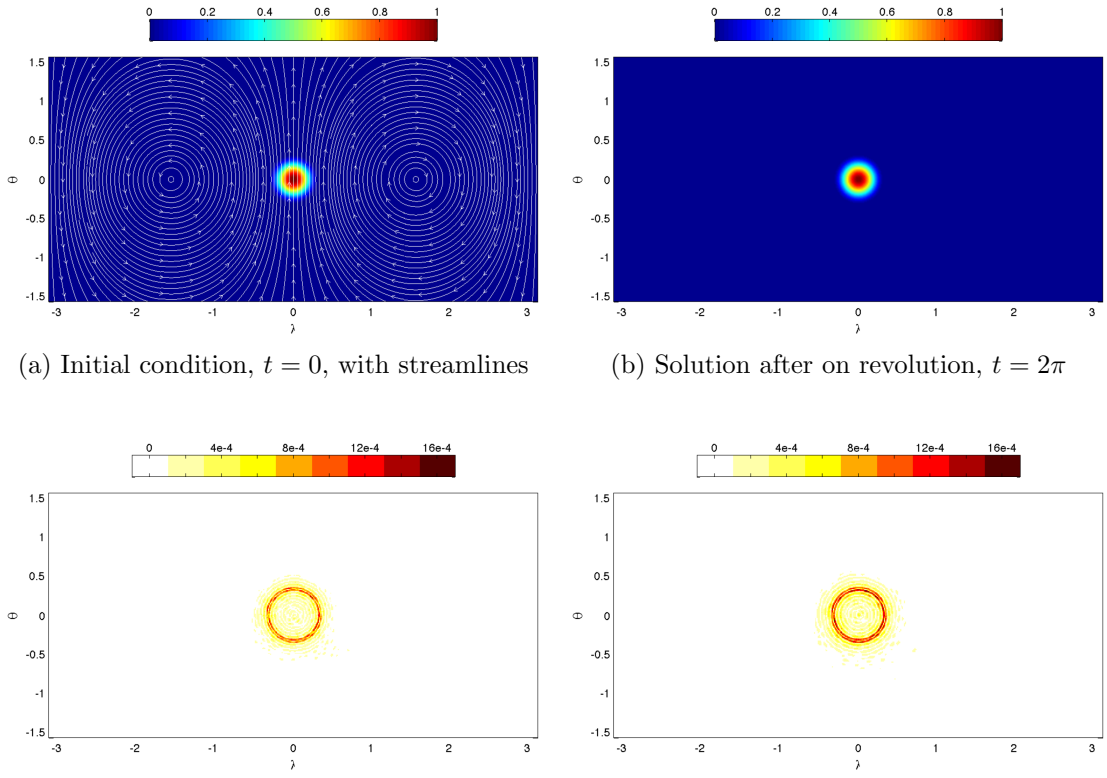
$$\psi\left(\mathbf{x}\right) = \quad x\sin\left(\alpha\right) - z\cos\left(\alpha\right) \tag{3.4}$$

This stream function results in solid body rotation at an angle of α with respect to the equator. In our test, we use the standard value of $\alpha = \frac{\pi}{2}$, which corresponds to the flow over the poles. The test is run up to time $t = 2\pi$, which corresponds to one full rotation at which point the solution is equal to Eq. 3.3. The error between the numerical and true solution is then computed.

Figures 3.1a and 3.1b show the initial condition and solution after one revolution;

these solutions are the same as dictated by the stream function (3.4). Figure 3.1c shows the magnitude of the error after one revolution and we can see that errors are concentrated near the discontinuities of the cosine bell. It can be seen in Figure 3.1d that the errors are still concentrated around the edges of the bell even after 10 revolutions. Thus, the scheme is performing quite well with respect to diffusive and dispersive errors.

In Figures 3.2 and 3.3, we plot the relative errors of the method for target condition numbers of $t_{\text{cond}} = 10^{14}$ and $t_{\text{cond}} = 10^{12}$. The figures show that the method producing approximately second order convergence for all n and both target condition numbers. This is the maximum convergence possible since the initial condition has a jump in its second derivative. As expected, increasing n decreases the error. Finally, we see that decreasing the target condition number does not have a significant effect on the errors and that both the IMQ and GA kernels are giving similar results.



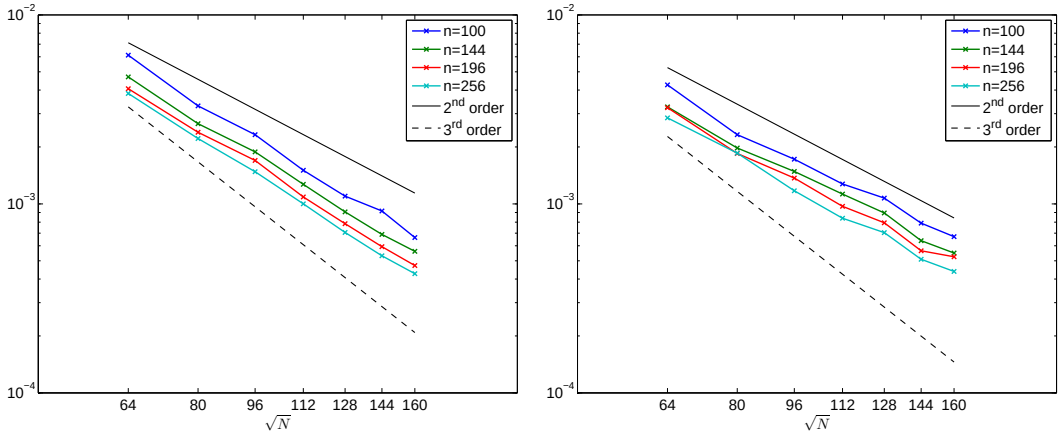
(a) Initial condition, $t = 0$, with streamlines

(b) Solution after on revolution, $t = 2\pi$

(c) Magnitude of the error after one revolution

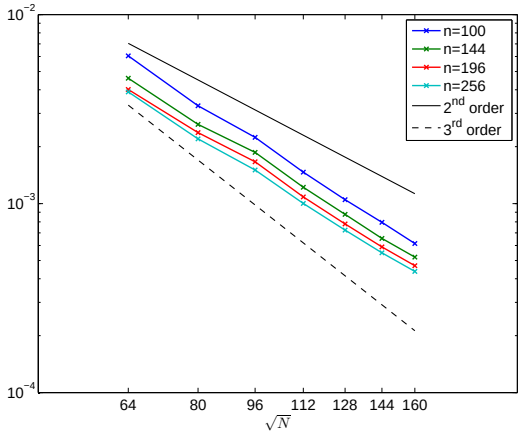
(d) Magnitude of the error after ten revolutions

Figure 3.1: Plots of the the solution and error using the GA kernel for $N = 12544$, $n = 144$, $\Delta t = 2\pi/1600$, $\mu = 8 \times 10^{-9}$.

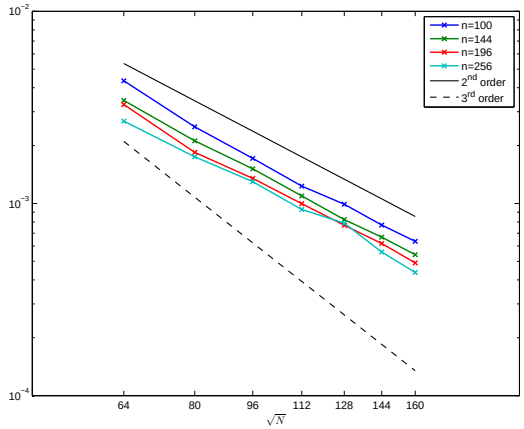


(a) GA l_∞ error vs. $\sqrt{N}$ (logscale)

(b) GA Relative l_∞ error vs. $\sqrt{N}$ (logscale)

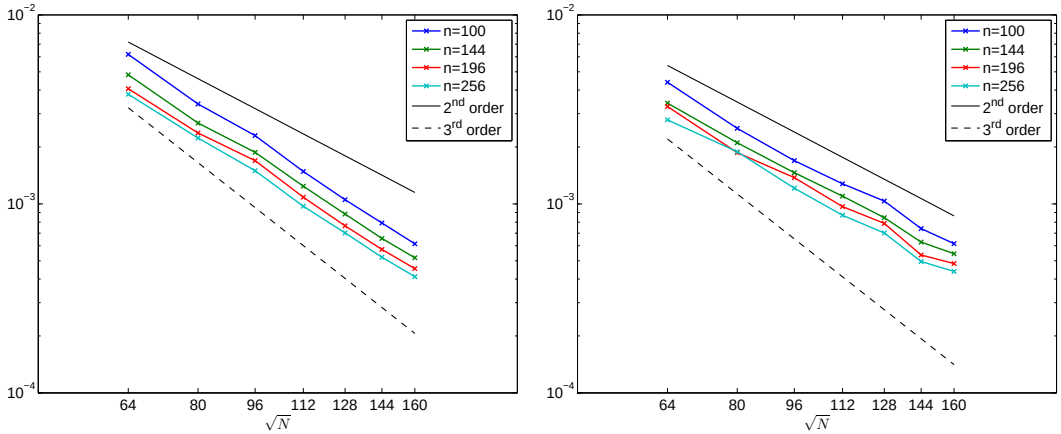


(c) IMQ Relative l_2 error vs. $\sqrt{N}$(logscale)



(d) IMQ Relative l_∞ error vs. $\sqrt{N}$ (logscale)

Figure 3.2: Convergence plots of the cosine bell test using the GA and IMQ kernels as a function of N and n, $\Delta t = 2\pi/1600$, and $t_{\text{cond}} = 10^{14}$; for the GA kernel test $\mu = 5 \times 10^{-11}$ while for the IMQ kernel $\mu = 6 \times 10^{-11}$.



(a) GA l_2 error vs. $\sqrt{N}$ (logscale)

(b) GA Relative l_∞ error vs. $\sqrt{N}$ (logscale)

(c) IMQ Relative l_2 error vs. $\sqrt{N}$(logscale)

(d) IMQ Relative l_∞ error vs. $\sqrt{N}$ (logscale)

Figure 3.3: Convergence plots of the cosine bell test using the GA and IMQ kernels as a function of N and n, $\Delta t = 2\pi/1600$, and $t_{\text{cond}} = 10^{12}$;$\mu = 5 \times 10^{-9}$ for both kernels.

3.2 Deformational Flow Tests

The next test is from Nair and Lauritzen [28]. The stream function for the flow is

$$\begin{aligned} \psi\left(\mathbf{x}\left(t\right)\right) &= 2\left[y\left(t\right)\right]^2 \cos\left(\frac{\pi t}{5} - \frac{2\pi}{5} z\left(t\right)\right), \\ \mathbf{x}\left(t\right) &= \left(\cos\left(\lambda - \frac{2\pi t}{5}\right)\cos\theta, \sin\left(\lambda - \frac{2\pi t}{5}\right), \sin\theta\right). \end{aligned} \tag{3.5}$$

This results in a velocity field that is a combination of solid-body rotation with a deformational component. There are two initial conditions considered. First is the non-smooth cosine bells, similar to the previous test:

$$\begin{aligned} h\left(\mathbf{x}\right) &= 0.1 + 0.9\left(h_1\left(\mathbf{x}\right) + h_2\left(\mathbf{x}\right)\right), \text{ where} \\ h_i(\mathbf{x}) &= \begin{cases} \frac{1}{2}\left(1 + \cos\left(2\pi r_i\left(\mathbf{x}\right)\right)\right), & r_i\left(\mathbf{x}\right) < \frac{1}{2}, \\ 0, & r_i\left(\mathbf{x}\right) \geq \frac{1}{2}, \end{cases} \\ r_i\left(\mathbf{x}\right) &= \arccos\left(x^T x_i\right), \\ x_1 &= \left(\frac{\sqrt{3}}{2}, \frac{1}{2}, 0\right), \\ x_2 &= \left(\frac{\sqrt{3}}{2}, -\frac{1}{2}, 0\right). \end{aligned} \tag{3.6}$$

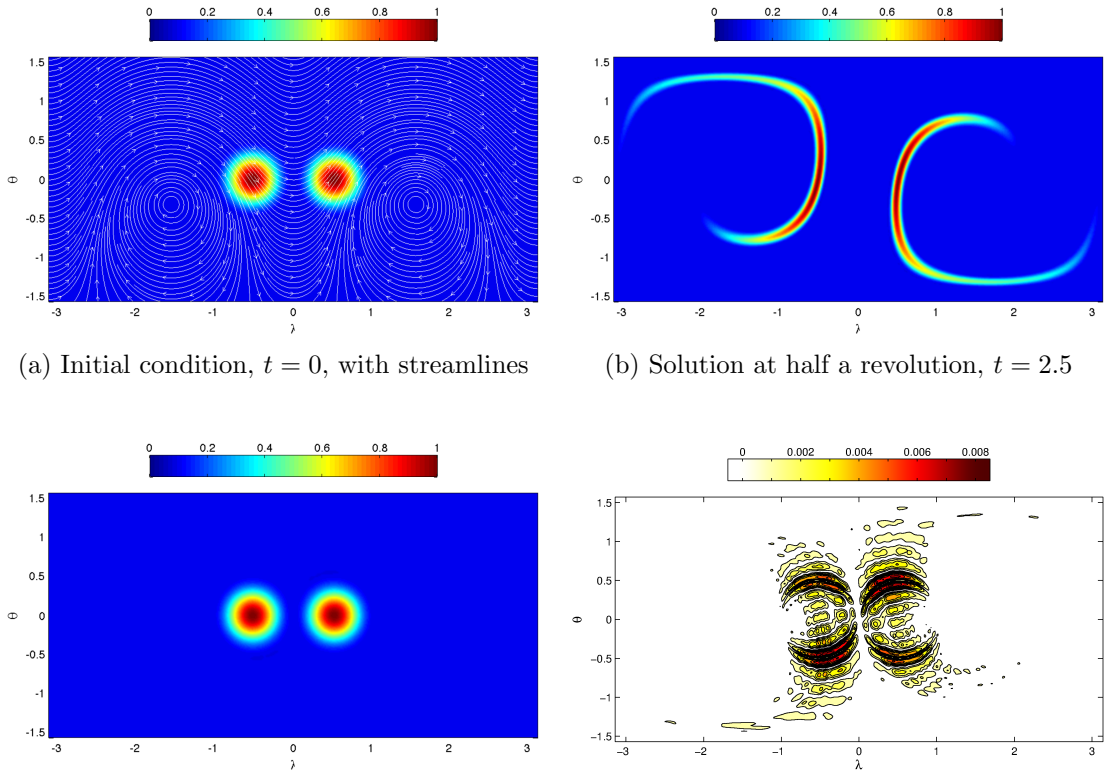
The second initial condition is the smooth Gaussian bells:

$$h(\mathbf{x}) = 0.95(\exp(-(4\|\mathbf{x} - \mathbf{x}_1\|)^2) + \exp(-4(\|\mathbf{x} - \mathbf{x}_2\|)). \tag{3.7}$$

This test advects the initial condition around the sphere while at the same time deforming them. At time $t = 2.5$, the flow field reverses and the initial condition returns to its initial positions at $t = 5$, where the errors are then measured.

Figures 3.4a, 3.4b, and 3.4c show the solution at the start, half a revolution, and one revolution respectively for the test using the non-smooth cosine bells. In Figure 3.4d, we see that the errors are most significant near the location of the discontinuity of the cosine bells. Figures 3.5 and 3.6 show that the convergence for both kernels appears to be second order. This is again related to the smoothness of the initial conditions. As with the cosine bell test in Section 3.1, there appears to be no significant change in errors when the target condition number is decreased and when the GA or IMQ kernels are used.

Figures 3.7a, 3.7b, and 3.7c show the solution at the start, half a revolution, and one revolution respectively but for the smooth Gaussian bells. The magnitude of the error Figure 3.7d is much lower than that of Figure 3.4d, but is still largely clustered around the location of the bells. The errors are plotted in Figures 3.5 and 3.6 on a log-linear scale instead of a log-log scale. The straight line behavior indicates that the errors appear to decrease at an exponential rate until a point where they level off. The point where they level off decreases with increasing condition number in line with the theory presented by Maz'ya and Schmidt [25]. The results show that saturation errors set in at roughly the same point for both the GA and IMQ kernels. However, even with saturation error, the method is still providing very accurate results compared to other methods that use a similar number of degrees of freedom [28].



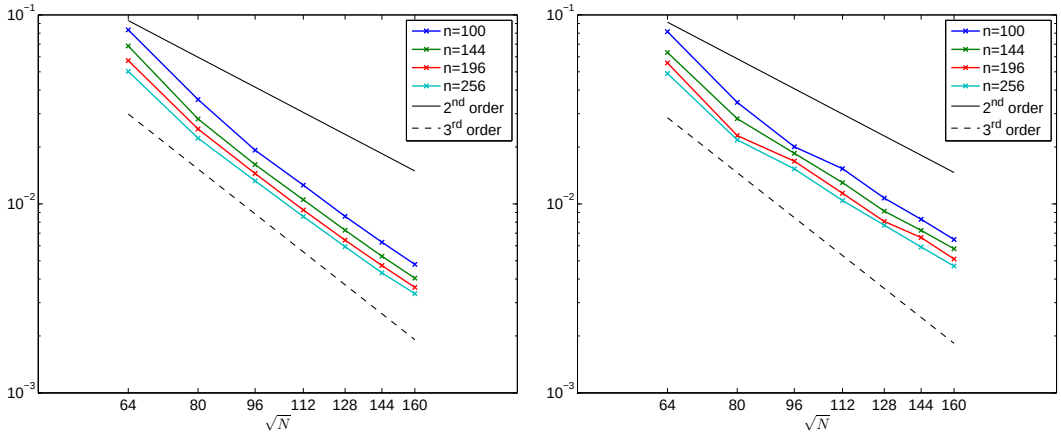
(a) Initial condition, $t = 0$, with streamlines

(b) Solution at half a revolution, $t = 2.5$

(c) Solution at time t=5 (full-revolution)

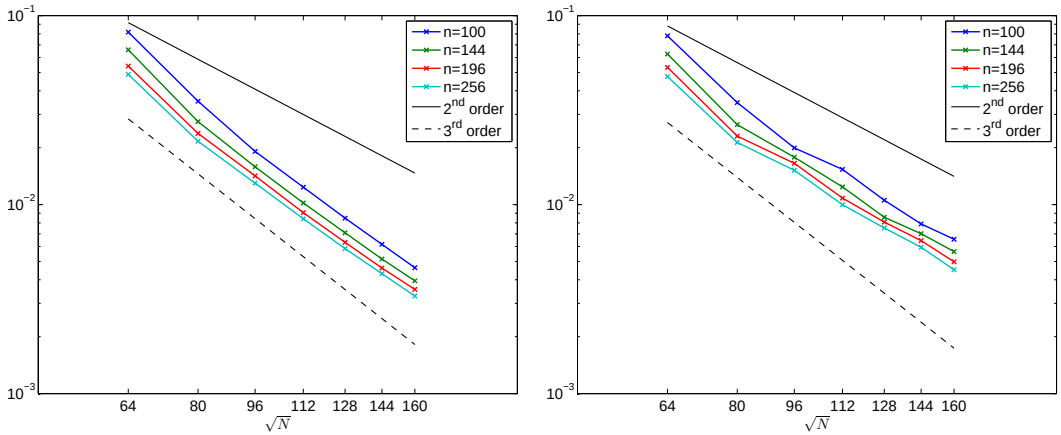
(d) Magnitude of the error at time t=5

Figure 3.4: Plots of the solution and error using non-smooth cosine bells with the GA kernel for $N = 20736$, $n = 100$, $\Delta t = 5/2400$, $\mu = 10^{-10}$, $t_{\text{cond}} = 10^{14}$.



(a) GA Relative l_2 error vs. $\sqrt{N}$(logscale)

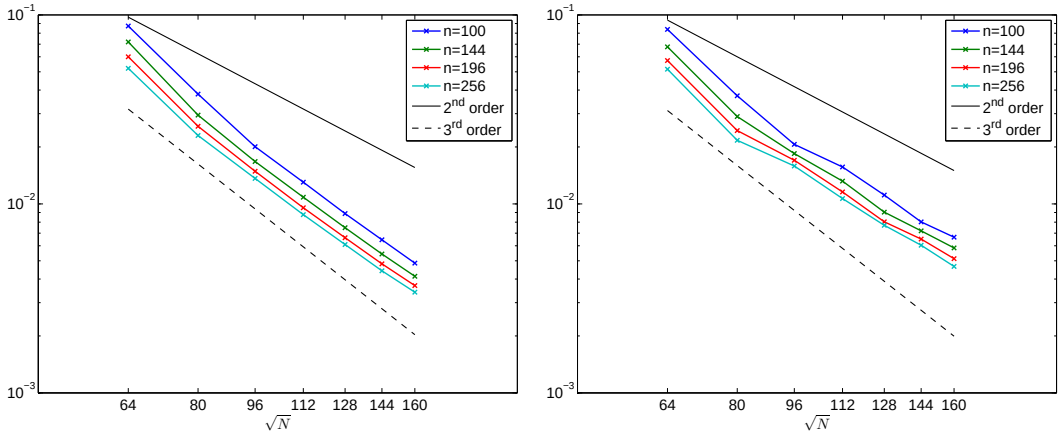
(b) GA Relative l_∞ error vs. $\sqrt{N}$ (logscale)



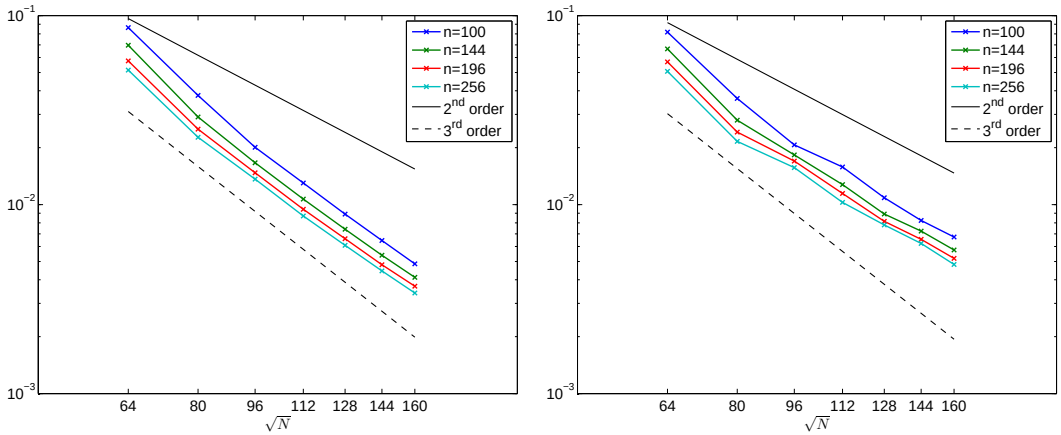
(c) IMQ Relative l_2 error vs. $\sqrt{N}$(logscale)

(d) IMQ Relative l_∞ error vs. $\sqrt{N}$ (logscale)

Figure 3.5: Convergence plots of the deformational flow test with cosine bell initial condition using the GA and IMQ kernels as a function of N and n using non-smooth cosine bells for $\Delta t = 5/2400$, and $t_{\text{cond}} = 10^{14}$; $\mu = 10^{-10}$ for both kernels.

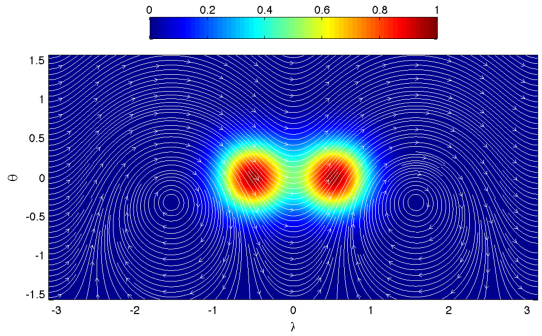


(a) GA Relative l_2 error vs. $\sqrt{N}$(logscale) (b) GA Relative l_∞ error vs. $\sqrt{N}$ (logscale)

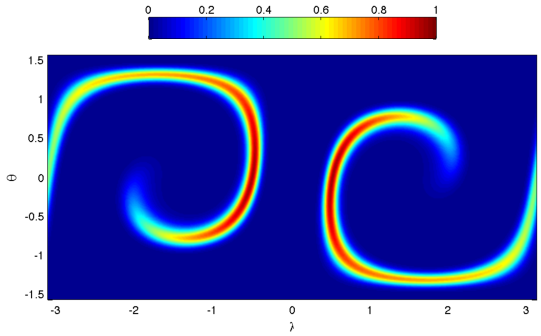


(c) IMQ Relative l_2 error vs. $\sqrt{N}$(logscale) (d) IMQ Relative l_∞ error vs. $\sqrt{N}$ (logscale)

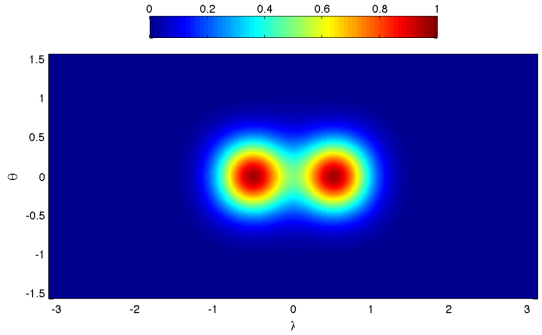
Figure 3.6: Convergence plots of the deformational flow test with cosine bell initial condition using the GA and IMQ kernels as a function of N and n using non-smooth cosine bells for $\Delta t = 5/2400$ and $t_{\text{cond}} = 10^{12}$;$\mu = 10^{-8}$ for both kernels.



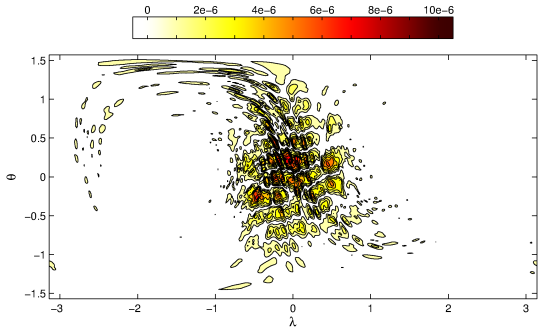
(a) Initial condition, $t = 0$, with streamlines



(b) Solution at half a revolution, $t = 2.5$

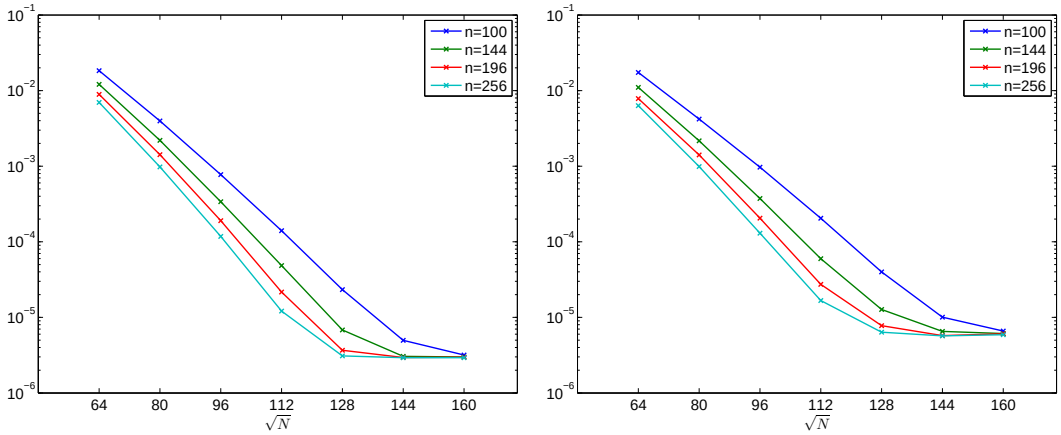


(c) Solution at time t=5 (full-revolution)



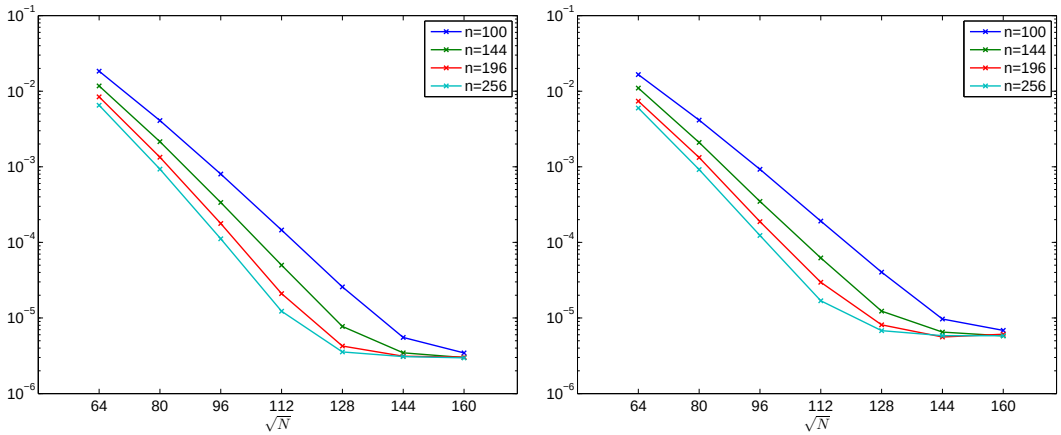
(d) Magnitude of the error at time t=5

Figure 3.7: Plots of the solution and error using smooth Gaussian bells with the GA kernel for $N = 20736$, $n = 100$, $\Delta t = 5/2400$, $\mu = 2.5 \times 10^{-10}$, $t_{\text{cond}} = 10^{14}$.



(a) GA Relative l_2 error vs. $\sqrt{N}$(loglinear scale)

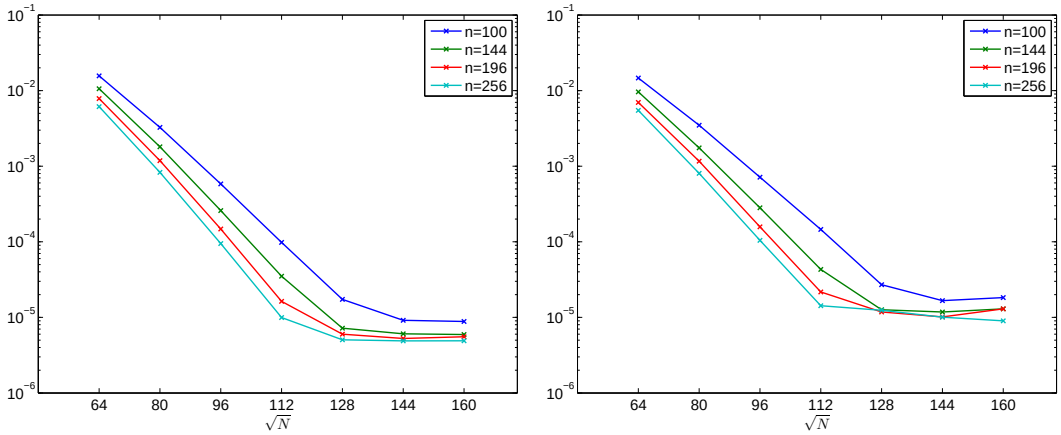
(b) GA Relative l_∞ error vs. $\sqrt{N}$ (loglinear scale)



(c) IMQ Relative l_2 error vs. $\sqrt{N}$(loglinear scale)

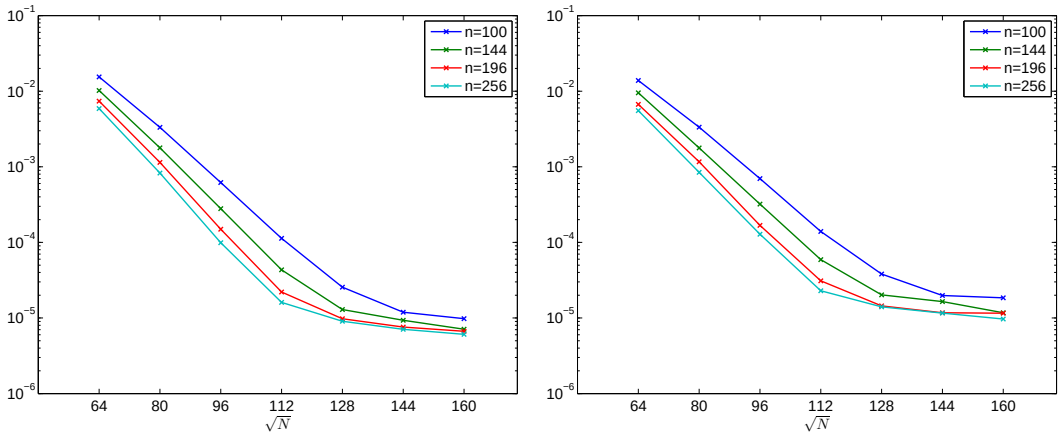
(d) IMQ Relative l_∞ error vs. $\sqrt{N}$ (loglinear scale)

Figure 3.8: Convergence plots of the deformational flow test with Gaussian bell initial condition using the GA and IMQ kernels as a function of N and n using smooth Gaussian bells for $\Delta t = 5/2400$ and $t_{\text{cond}} = 10^{14}$;$\mu = 2.5 \times 10^{-10}$ for both kernels.



(a) GA Relative l_2 error vs. $\sqrt{N}$ (loglinear scale)

(b) GA Relative l_∞ error vs. $\sqrt{N}$ (loglinear scale)



(c) IMQ Relative l_2 error vs. $\sqrt{N}$ (loglinear scale)

(d) IMQ Relative l_∞ error vs. $\sqrt{N}$ (loglinear scale)

Figure 3.9: Convergence plots of the deformational flow test with Gaussian bell initial condition using the GA and IMQ kernels as a function of N and n using smooth Gaussian bells for $\Delta t = 5/2400$ and $t_{\text{cond}} = 10^{12}$; $\mu = 10^{-8}$ for both kernels.

3.3 Stationary Vortex Roll-Up

In this test, two vortices are generated at the north and south poles of the sphere, providing an idealized model for cyclogenisis. This test was first introduced in [26]. The velocity field is given by

$$\begin{aligned} u &= \omega(\theta)\cos\theta, \\ v &= 0 \end{aligned} \tag{3.8}$$

with

$$\omega(\theta) = \begin{cases} \frac{3\sqrt{3}}{2}\operatorname{sech}^2(\rho(\theta))\tanh(\rho(\theta)) & \rho(\theta) \neq 0 \\ 0 & \rho(\theta) = 0 \end{cases} \tag{3.9}$$

where $\rho(\theta) = \rho_0 \cos(\theta)$ and ρ_0 is a parameter controlling the radial extent of the vortex; in this test $\rho_0 = 3$. The analytical solution to this PDE is given by:

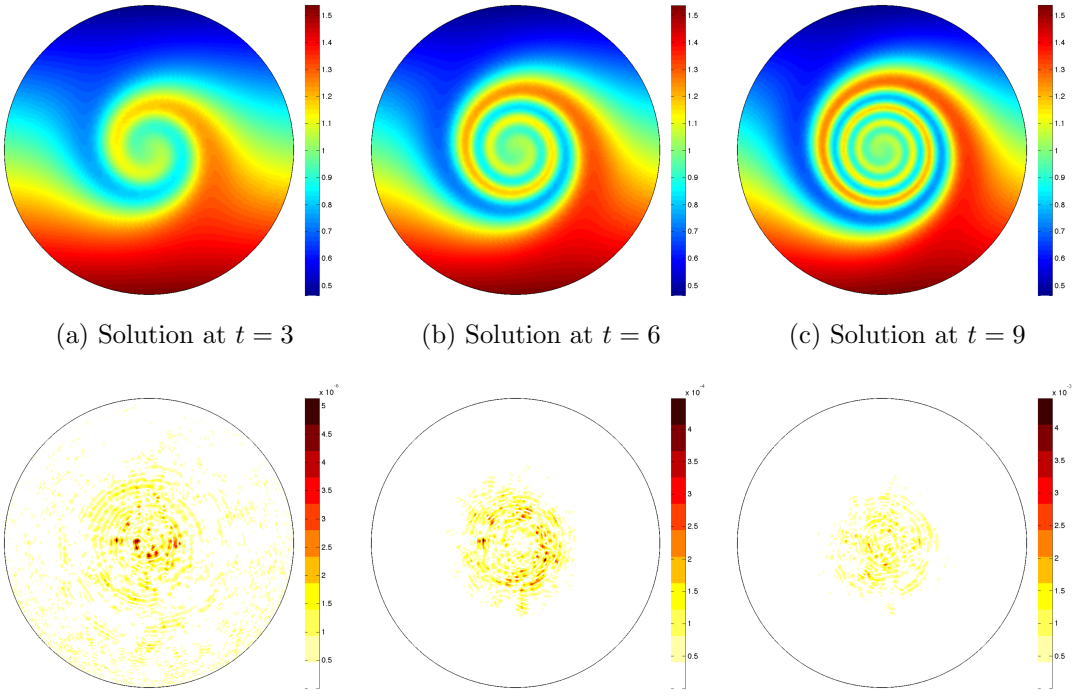
$$h(\lambda, \theta, t) = 1 - \tanh\left(\frac{\rho(\theta)}{5}\sin(\lambda - \omega(\theta)t)\right). \tag{3.10}$$

The initial condition is given by Eq. 3.10 at $t = 0$. The test calls for computing the errors in the numerical solution at time $t = 3$ using the analytical solution (3.10).

Numerical solutions for this test problem are plotted at times $t = 3$, $t = 6$ and $t = 9$ in Figures 3.10a, 3.10b, and 3.10c, respectively. The magnitude of the errors at these times are plotted in Figures 3.10d, 3.10e, and 3.10f. We see that as time increases, the errors become more and more concentrated at the centers of the vortices (where the gradients are the highest).

Figure 3.11 and 3.12 show the relative errors in the solution for the two target condition numbers. In these figures, the errors are plotted on a log-log scale and initially

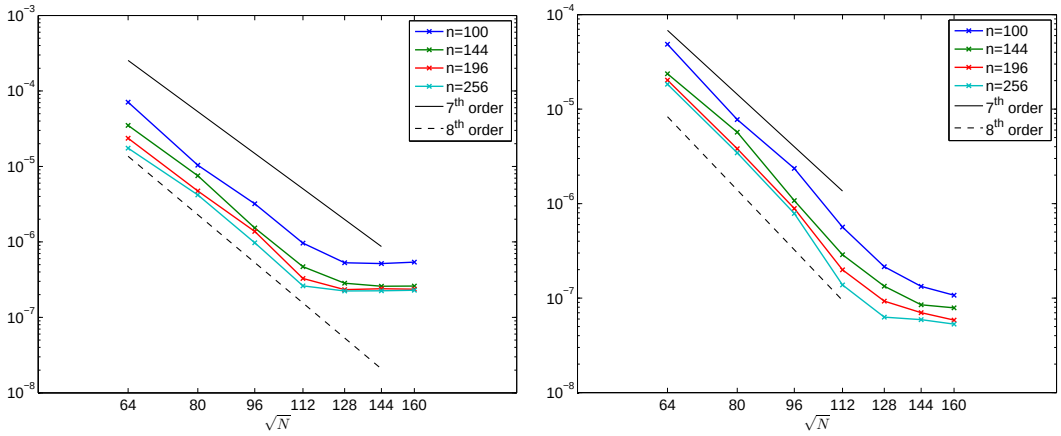
the results indicate the method is giving between 7th and 8th order convergence. As in the previous tests with the Gaussian bell, we do see that saturation errors again show up for increasing N. Increasing the condition number does allow convergence to proceed further with increasing N, but eventually saturation does appear. The results also show that the IMQ kernel is less susceptible to saturation errors than the GA kernel for this test, as the IMQ kernel is able to achieve a relative error at least one order of magnitude lower than the GA kernel. Again even with saturation error, the method is still providing very accurate results compared to other methods that use a similar number of degrees of freedom [27].



(a) Solution at $t = 3$ (b) Solution at $t = 6$ (c) Solution at $t = 9$

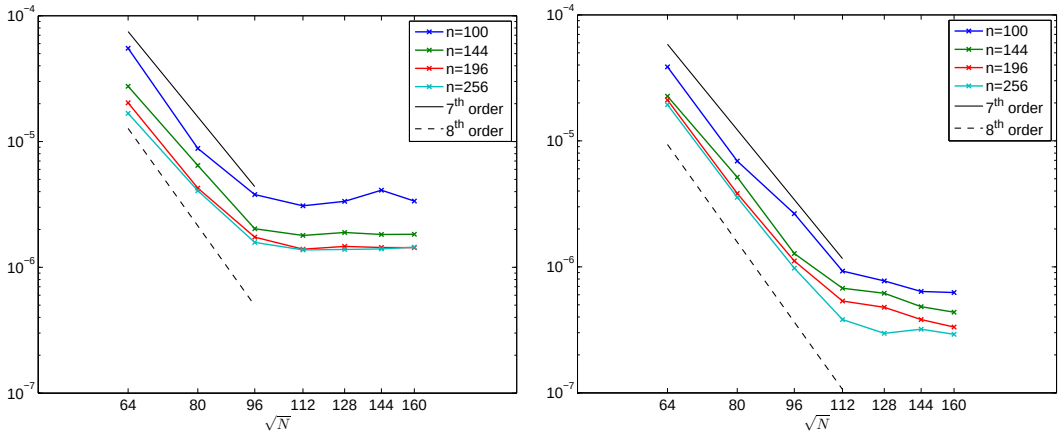
(d) Magnitude of the error at time $t = 3$ (e) Magnitude of the error at time $t = 6$ (f) Magnitude of the error at time $t = 9$

Figure 3.10: Plots of the solution and error of the stationary vortex roll-up test with the GA kernel for $N = 16384$, $n = 100$, $\Delta t = 1/25$, $\mu = 9.8 \times 10^{-12}$,$t_{\text{cond}} = 10^{14}$.



(a) GA Relative l_2 error vs. $\sqrt{N}$(logscale) (b) IMQ Relative l_2 error vs. $\sqrt{N}$(logscale)

Figure 3.11: Convergence plots of stationary vortex roll-up test using the GA and IMQ kernels as a function of N and n for $\Delta t = 1/25$ and $t_{\text{cond}} = 10^{14}$;for the GA kernel test $\mu = 9.8 \times 10^{-12}$, while for the IMQ kernel $\mu = 10^{-11}$.



(a) GA Relative l_2 error vs. $\sqrt{N}$(logscale) (b) IMQ Relative l_2 error vs. $\sqrt{N}$(logscale)

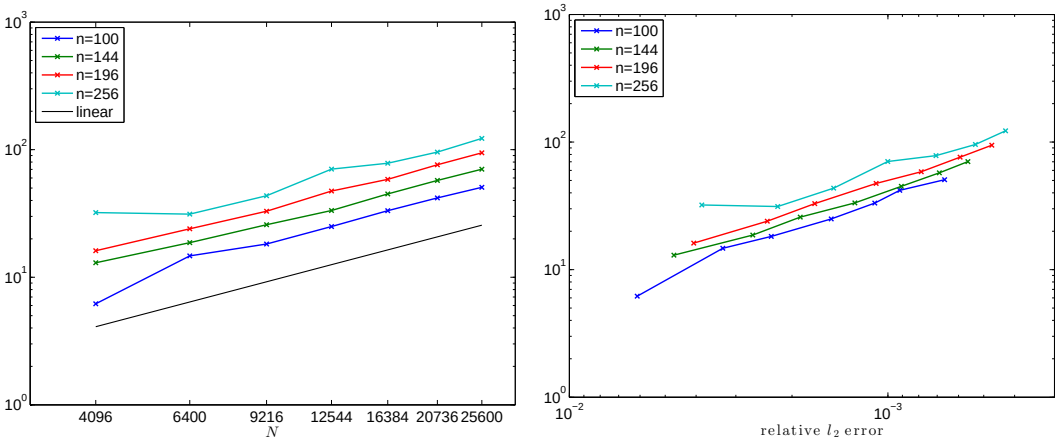
Figure 3.12: Convergence plots of stationary vortex roll-up test using the GA and IMQ kernels as a function of N and n for $\Delta t = 1/25$ and $t_{\text{cond}} = 10^{12}$;for the GA kernel test $\mu = 5 \times 10^{-10}$, while for the IMQ kernel $\mu = 6 \times 10^{-10}$.

3.4 Computational Performance

Here we analyze the computational performance of the RBF-PUM using the wall-clock time in seconds and relative l_2 errors for simulations from the cosine bell test from Section 3.1 and the deformational flow test with the smooth Gaussian bells initial condition from Section 3.2. The machine we used had Intel Xeon processors at 3.10 GHz.

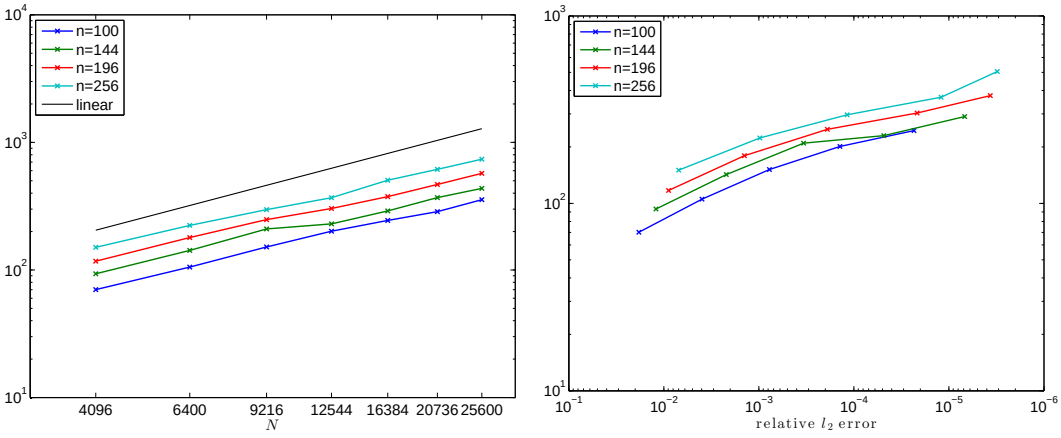
In part (a) of Figures 3.13-3.14, the wall-clock time is plotted against the number of nodes N using a log-log scale. In both figures, we see that the wall-clock time grows linearly with N. From Section 2.5, we have that the the number of non zeros of the RBF-PUM differentiation matrix $nnz = \mathcal{O}(nqN)$, where n is the number of nodes per patch and q is the average number of patches a point belongs to. The dominate computational term for the time integration in our tests is the matrix-vector multiplication with the differential matrix. Thus, we would expect the wall-clock time to grow asymptotically similarly to nnz with respect to N, which in these test it is.

In part (b) of Figures 3.13-3.14, the wall-clock time is plotted against the relative l_2 error using a log-log scale. With these plots, we can determine for a level of error err, which N and n will be the most efficient to evaluate the test so that relative error is at most err. For both tests pairs of N and n with $n = 100$ are the most time efficient for any level err before a saturation is reached. This suggests that if a high level of accuracy is desired, it would be more time efficient to keep n low and raise N.



(a) Wall-clock time (s) vs. N (logscale) (b) Wall-clock time vs. relative l_2 error (logscale)

Figure 3.13: Plots for the cosine bell test for wall-clock time (sec) vs. N and wall-clock time vs. relative l_2 error using the GA kernel with $\Delta t = 2\pi/1600$, and $t_{\text{cond}} = 10^{14}$ and $\mu = 5 \times 10^{-11}$.



(a) Wall-clock time (s) vs. N (logscale) (b) Wall-clock time vs. relative l_2 error (logscale)

Figure 3.14: Plots for the deformational flow test with Gaussian bell initial condition for wall-clock time (sec) vs. N and wall-clock time vs. relative l_2 error using the GA kernel with$\Delta t = 5/2400$ and $t_{\text{cond}} = 10^{14}$ and $\mu = 2.5 \times 10^{-10}$.

CHAPTER 4

FUTURE WORK

In this thesis, we have introduced the radial basis function partition of unity method (RBF-PUM) for solving the transport equation on the surface of the sphere and applied it to several benchmark problems from the literature. The method scales linearly with the number of degrees of freedom and provides high orders of accuracy for sufficiently smooth initial conditions. While our results are promising, more work is needed to realize the full potential of RBF-PUM. In this chapter, we lay down suggestions for future research.

First, methods for choosing the hyperviscosity parameter μ need to be explored. Right now it is chosen through trial and error. Fornberg and Lehto [15] give the following suggestions on how μ should be chosen for the RBF-FD method, which share similarities to RBF-PUM:

- Numerical experiments can be run on low N. For the RBF-FD method, they found scaling $\mu \sim N^{-2}$ worked well.

- Calculate an approximation for the eigenvalue of the differentiation matrix D with the largest real part using an iterative eigenvalue routine for sparse matrices. While these algorithms are efficient, they can occasionally fail to converge (such as Matlab's eigs).

These suggestions can be explored in the context of the RBF-PUM. One benefit of RBF-PUM over the RBF-FD method is that the choice of μ is less sensitive than the RBF-FD method in respect to changing the nodes per patch n and the total nodes N.

In order for the RBF-PUM to be practical, adaptive and static node refinements needs to be developed. Note that with mesh-free methods only the location of the nodes needs to be considered. Starting with the ideas of node refinements for the global RBF method [9], we could explore building this capability into the RBF-PUM. In order for there to be node refinement in the RBF-PUM, a technique would need to be developed to choose that patches such that each patch is relatively the same size. We have already started preliminary work using recursive subdivision of spherical triangles.

As discussed in Section 1.2, when using the global RBF method, smaller ε will lead to more accurate solutions at the cost of a more ill-condition interpolation matrix. Methods such as RBF-QR tackle this issue on the sphere. We do not utilize the flat-limit RBF's in the patches (like the RBF-QR method) as discussed in Section 1.2. Continued research into these methods is necessary to eliminate saturation errors from the RBF-PUM.

A parallel implementation is necessary for either pushing the computational performance further or working with very large N. A popular choice is to utilize GPUs. The RBF-PUM has attractive features for a GPU approach since the method decomposes the problem of approximating spatial derivatives into M global-RBF differentiation matrices of size $\mathcal{O}(n^2)$, where these matrices are small enough to be managed by the cores of the GPU. Building on the GPU implementation of the RBF-FD method in [1], a RBF-PUM GPU implementation can be developed.

In order for a methodology to be widely accepted in climate and weather modeling communities the method needs pass a test suite of dynamical core benchmarks, which include [20, 23, 33]. The RBF-PUM needs to be extended to the full shallow water wave equations to tackle these benchmarks. Once this is done, it needs to be compared to state-of-the-art numerical methods used by global general circulation models.

REFERENCES

[1] E. Bollig, N. Flyer, and G. Erlebacher. A multi-CPU/GPU implementation of RBF-generated finite differences for PDEs on a sphere. *AGU*, IN77, 2011.

[2] E. Bollig, N. Flyer, and G. Erlebacher. Solution to PDEs using radial basis function finite-differences (RBF-FD) on multiple GPUs. *J. Comput. Phys.*, 231:7133–7151, 2012.

[3] M. D. Buhmann. A new class of radial basis functions with compact support. *Math. Comput.*, 70(233):307–318, 1988.

[4] Martin Dietrich Buhmann. *Multivariable Interpolation Using Radial Basis Functions*. PhD thesis, University of Cambridge, Cambridge, England, May 1989.

[5] R. Cavoretto and A. DeRossi. Fast and accurate interpolation of large scattered data sets on the sphere. *J. Comput. Appl. Math.*, 234:1505–1521, 2010.

[6] A.M. Encinas J.M. Gesto E. Bendito, A. Carmona. Estimation of Fekete points. *Journal of Computational Physics*, 225:2354–2376, 2007.

[7] G. E. Fasshauer. *Meshfree Approximation Methods with MATLAB*. Interdisciplinary Mathematical Sciences - Vol. 6. World Scientific Publishers, Singapore, 2007.

[8] G. E. Fasshauer and M. J. McCourt. Stable evaluation of Gaussian radial basis function interpolants. *SIAM J. Sci. Comput.*, 34:A737–A762, 2012.

[9] N. Flyer and E. Lehto. Rotational transport on a sphere: Local node refinement with radial basis functions. *J. Comput. Phys.*, 229:1954–1969, 2010.

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

[11] N. Flyer and G. B. Wright. Transport schemes on a sphere using radial basis functions. *J. Comput. Phys.*, 226:1059–1084, 2007.

[12] N. Flyer and G. B. Wright. A radial basis function method for the shallow water equations on a sphere. *Proc. Roy. Soc. A*, 465:1949–1976, 2009.

[13] B. Fornberg, T. A. Driscoll, G. Wright, and R. Charles. Observations on the behavior of radial basis functions near boundaries. *Comput. Math. Appl.*, 43:473–490, 2002.

[14] B. Fornberg, E. Larsson, and N. Flyer. Stable computations with Gaussian radial basis functions. *SIAM J. Sci. Comput.*, 33(2):869–892, 2011.

[15] B. Fornberg and E. Lehto. Stabilization of RBF-generated finite difference methods for convective PDEs. *J. Comput. Phys.*, 230:2270–2285, 2011.

[16] B. Fornberg and C. Pirét. A stable algorithm for flat radial basis functions on a sphere. *SIAM J. Sci. Comput.*, 30:60–80, 2007.

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

[18] R. Franke. Scattered data interpolation: tests of some methods. *Math. Comput.*, 38:181–200, 1982.

[19] E. J. Fuselier and G. B. Wright. Scattered data interpolation on embedded submanifolds with restricted positive definite kernels: Sobelv error estimates. *SIAM J. Num. Anal.*, 50:17531776, 2012.

[20] J. Galewsky, R. K. Scott, and L. M. Polvani. An initial-value problem for testing numerical models of the global shallow-water equations. *Tellus*, 56A:429–440, 2004.

[21] M. Götz. On the Riesz energy of measures. *Journal of Approximation Theory*, 122:62–78, 2003.

[22] R. L. Hardy. Multiquadric equations of topograpy and other irregular surfaces. *J. Geophy. Res.*, 76:1905–1915, 1971.

[23] Christiane Jablonowski and David L. Williamson. A baroclinic instability test case for atmospheric model dynamical cores. *Quarterly Journal of the Royal Meteorological Society*, 132(621C):2943–2975, 2006.

[24] E. J. Kansa. Multiquadrics – a scattered data approximation scheme with applications to computational fluid-dynamics – ii: Solutions to parabolic, hyperbolic and elliptic partial differential equations. *Comput. Math. Appl.*, 19:147–161, 1990.

[25] Vladimir Maz'ya and Gunther Schmidt. On approximate approximations using gaussian kernels. *Journal of Numerical Analysis*, 16:13–29, 1996.

[26] R. D. Nair, J. Côté, and A. Staniforth. Cascade interpolation for semi-lagrangian advection over the sphere. *Quart. J. Roy. Meteor. Soc.*, 125:1445–1468, 1999.

[27] R. D. Nair, S. J. Thomas, and R. D. Loft. A discontinuous Galerkin transport scheme on the cubed-sphere. *Mon. Wea. Rev.*, 133:814–828, 2005.

[28] Ramachandran D. Nair and Peter H. Lauritzen. A class of deformational flow test cases for linear transport problems on the sphere. *Journal of Computational Physics*, 229:8868–8887, 2010.

[29] R. Schaback. Error estimates and condition numbers for radial basis function interpolants. *Adv. Comput. Math.*, 3:251–264, 1995.

[30] Ian H. Sloan and Robert S. Womersley. Extremal systems of points and numerical integration on the sphere. *Advances in Computational Mathematics*, 192:107–125, 2004.

[31] Yves Talpaert. *Differential Geometry with Applications to Mechanics and Physics*. Marcel Dekker, New York, 2000.

[32] H. Wendland. *Scattered Data Approximation*, volume 17 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2005.

[33] D. L. Williamson, J. B. Drake, J. J. Hack, R. Jakob, and P. N. Swarztrauber. A standard test set for numerical approximations to the shallow water equations in spherical geometry. *J. Comput. Phys.*, 102:211–224, 1992.

[34] R. S. Womersley and I. H. Sloan. How good can polynomial interpolation on the sphere be? *Adv. Comput. Math.*, 23:195–226, 2001.

[35] R. S. Womersley and I. H. Sloan. Interpolation and cubature on the sphere. Website, 2003/2007. http://web.maths.unsw.edu.au/ rsw/Sphere/.

APPENDIX A

CHOOSING NODE SETS

A.1 MD Points

We downloaded our MD node sets from [35]. As described in Section 2.2, the MD method generates the nodes by choosing the point set the maximizes the determent of an interpolating matrix. The method described below is from [34]. Let $\{\mathbf{x}_i\}_{i=1}^n$ be the node set. The matrix depends on this kernel:

$$G_n(\mathbf{x},\mathbf{y}) = \sum_{l=0}^{n} \sum_{k=1}^{N(r,l)} Y_{l,k}^{(r)}(\mathbf{x})\, Y_{l,k}^{(r)}(\mathbf{y}) \tag{A.1}$$

where $Y_{l,k}^{(r)}$ is a spherical harmonic. Let G be the interpolating matrix. Then

$$G_{ij} = G_n(\mathbf{x}_i, \mathbf{x}_j). \tag{A.2}$$

The MD points are found by finding the set $\{\mathbf{x}_i\}_{i=1}^n$ that maximizes $\log(\det(G))$. In [34], MD points were found to be consistently effective as interpolation points.

A.2 ME Points

We generated our own ME node sets from size 11 to 1400. We used the algorithm dictated by [6]. Notice that the ME node set $\{\mathbf{x}_i\}_{i=1}^N$ on $\mathbb{S}^2$ is called the N_{th} order

Fekete points of $\mathbb{S}^2$. The algorithm finds the set of nodes that minimizes

$$\sum_{1 \leq i \leq j \leq N} \|\mathbf{x}_i - \mathbf{x}_j\|^{-2}. \tag{A.3}$$

What is interesting is that the authors of [6] treat the problem mechanically by solving an ODE of particles repelling each other. The problem is considered solved when a steady state is achieved.

APPENDIX B

TABLES FOR N AND Q VALUES

In Section 2.2, we gave formulas (Eq. 2.8 and Eq. 2.10) for choosing the size of the patches and the number of patches with parameters n (number of nodes per patch) and q (average number of patches a point belongs too). The tables below numerically verify that when these formulas are used, n and q correspond to there real associated values.

B.1 Number of Nodes per Patch

Tables B.1, B.2 and B.3 show the average and standard deviation for the number of nodes per patch. Mean values correspond very closely to n, and the standard deviation is quite low. This holds for all q that was tested as well. This highly indicates that the number of nodes per patch corresponds to n and hence validates how the radius is chosen (2.8).

Table B.1: Mean and standard deviation of the number of nodes per patch with q=3

(a) Mean

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	99.99	100.01	100.03	100.04	100.14	100.05	99.94
144	144.15	143.89	144.20	144.07	144.01	143.85	144.09
196	196.25	195.94	195.80	196.21	196.04	195.86	195.90
256	255.81	256.21	255.86	255.91	255.95	256.04	255.93

(b) Standard Deviation

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	1.45	1.67	1.75	1.68	1.60	1.60	1.69
144	1.73	1.67	1.59	1.53	1.68	1.77	1.82
196	2.06	1.83	2.03	2.21	1.84	1.88	2.01
256	2.03	2.28	1.97	1.97	1.91	1.98	2.06

Table B.2: Mean and standard deviation of the number of nodes per patch with q=3.5

(a) Mean

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	99.90	99.93	99.93	99.86	100.03	100.00	100.00
144	143.95	144.06	144.22	143.95	143.90	143.98	144.05
196	195.86	196.15	196.08	195.96	196.17	195.97	196.00
256	255.59	256.13	256.02	256.13	256.00	255.93	256.15

(b) Standard Deviation

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	1.58	1.67	1.53	1.60	1.69	1.63	1.69
144	1.67	1.71	1.66	1.61	1.74	1.61	1.65
196	1.73	2.07	1.92	1.99	1.98	1.87	2.08
256	1.98	1.97	1.86	1.98	1.92	2.03	1.94

Table B.3: Mean and standard deviation of the number of nodes per patch with q=4

(a) Mean

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	99.99	99.99	99.96	99.89	99.84	100.00	100.03
144	143.94	143.96	144.19	143.90	144.01	144.11	144.03
196	195.99	196.08	195.91	196.20	195.96	195.96	196.00
256	255.70	256.37	256.28	255.72	256.05	255.89	256.04

(b) Standard Deviation

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	1.68	1.66	1.67	1.65	1.66	1.55	1.66
144	1.55	1.79	1.68	1.61	1.70	1.72	1.73
196	1.98	1.86	1.92	2.09	2.02	1.97	2.05
256	1.86	1.97	2.06	2.09	2.17	2.09	1.97

B.2 Number of Patches a Node Belongs to

Tables B.4, B.5 and B.6 show the mean and standard deviation of the number of patches a point belongs to. Our claim in Section 2.2 was that q corresponds with this average. Considering that the mean values correspond very closely to the parameter q and the standard deviation is small, there is strong evidence that q is this average. Thus, the way the number of patches is chosen (2.10) is validated.

Table B.4: Mean and standard deviation of the number of patches a node belongs to with q=3

(a) Mean

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	3.00	3.00	3.01	3.01	3.01	3.01	3.00
144	3.03	3.01	3.00	3.01	3.01	3.00	3.01
196	3.02	3.00	3.02	3.00	3.00	3.00	3.00
256	3.00	3.00	3.00	3.00	3.00	3.00	3.00

(b) Standard Deviation

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	0.57	0.58	0.57	0.57	0.57	0.56	0.57
144	0.57	0.57	0.58	0.57	0.57	0.57	0.57
196	0.57	0.56	0.57	0.58	0.57	0.57	0.57
256	0.56	0.57	0.57	0.57	0.58	0.57	0.58

Table B.5: Mean and standard deviation of the number of patches a node belongs to with q=3.5

(a) Mean

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	3.51	3.50	3.50	3.50	3.50	3.50	3.50
144	3.51	3.51	3.51	3.50	3.50	3.50	3.51
196	3.54	3.52	3.51	3.50	3.51	3.51	3.51
256	3.49	3.52	3.50	3.51	3.50	3.51	3.50

(b) Standard Deviation

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	0.54	0.53	0.54	0.54	0.53	0.53	0.53
144	0.53	0.54	0.53	0.54	0.53	0.54	0.54
196	0.55	0.53	0.54	0.53	0.54	0.54	0.53
256	0.54	0.54	0.54	0.54	0.53	0.54	0.54

Table B.6: Mean and standard deviation of the number of patches a node belongs to with q=3.5

(a) Mean

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	4.00	4.00	4.00	4.00	4.00	4.00	4.00
144	4.01	4.00	4.01	4.00	4.01	4.00	4.01
196	4.02	4.01	4.02	4.00	4.01	4.01	4.00
256	4.00	4.01	4.00	4.02	4.00	4.00	4.00

(b) Standard Deviation

$n \diagdown N$	**4096**	**6400**	**9216**	**12544**	**16384**	**20736**	**25600**
100	0.65	0.63	0.64	0.64	0.63	0.64	0.63
144	0.63	0.64	0.64	0.64	0.64	0.63	0.63
196	0.64	0.64	0.64	0.63	0.64	0.64	0.64
256	0.64	0.64	0.64	0.64	0.63	0.64	0.63