A FICTITIOUS POINT METHOD FOR HANDLING BOUNDARY CONDITIONS IN THE RBF-FD METHOD

by

Joseph Lohmeier

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Mathematics

Boise State University

August 2011

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Joseph Lohmeier

Thesis Title: A Fictitious Point Method for Handling Boundary Conditions in the RBF-FD Method

Date of Final Oral Examination: 01 August 2011

The following individuals read and discussed the thesis submitted by student Joseph Lohmeier, 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
Jodi Mead, Ph.D.	Member, Supervisory Committee
Lejo Flores, Ph.D.	Member, Supervisory Committee
Natasha Flyer, Ph.D.	Member, Supervisory Committee

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

dedicated to Jerid Sturman-Camyn

ACKNOWLEDGMENTS

The author wishes to acknowledge some of the contributors to this thesis outside of the committee. First, NSF Grant DMS-0934581, for funding the author's research assistantship. Also, he wishes to thank Dr. Natasha Flyer for the internship opportunity at the National Center for Atmospheric Research (NCAR). During the author's internship at the NCAR, Dr. Bengt Fornberg invited him to discuss the methods at the University of Colorado, Boulder. This was very helpful and the author thanks him for his contributions. Lastly, the author expresses his gratitude to his parents, Sue and Joe Lohmeier, and his roommate, James Hensley, for all of their encouraging words.

ABSTRACT

The main attraction of using radial basis functions (RBFs) for generating finite difference type approximations (RBF-FD) is that they naturally work for unstructured or scattered nodes. Therefore, a geometrically complex domain can be efficiently discretized using scattered nodes and continuous differential operators such as the Laplacian can be effectively approximated locally using RBF-FD formulas on these nodes. This RBF-FD method is becoming more and more popular as an alternative to the finite-element method since it avoids the sometimes complex and expensive step of mesh generation and the RBF-FD method can achieve much higher orders of accuracy. One of the issues with the RBF-FD method is how to properly handle non-Dirichlet boundary conditions. In this thesis, we describe an effective method for handling Neumann conditions in the case of Poisson's equation. The method uses fictitious points and generalized Hermite-Birkhoff interpolation to enforce the boundary conditions and to improve the accuracy of the RBF-FD method near boundaries. We present several numerical experiments using the method and investigate its convergence and accuracy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Interpolation of scattered data is a common problem that arises in many disciplines. For example, one may be given measurements of quantity, such as soil moisture content, at spatially varying locations and want to interpolate the data to a location where the information is unknown. In one dimension, many interpolation methods exist (e.g., polynomial and Fourier methods) and each of them uses the same basic principle. Given a set of data sites $X = \{x_1, \ldots, x_n\}$ and a set of linearly independent functions $\psi_i(x)$, the interpolant to the data, $\{f(x_1), \ldots, f(x_n)\}$, is given by

$$s(x) = \sum_{i=1}^{n} \lambda_i \psi_i(x), \tag{1.1}$$

where λ_i are determined by requiring $s(x_i) = f(x_i)$ for $i = 1, \ldots, n$. These conditions lead to a linear system for the interpolation weights λ_i, which is non-singular if all elements of X are distinct. Unfortunately, this method does not extend to multiple dimensions. In fact, it can be shown that for any set of basis functions independent of the data, $\{\psi_i(\mathbf{x})\}_{i=1}^n$, there exists a set of distinct data sites $\{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^{d>1}$, such that the linear system for interpolation becomes singular [17]. Therefore, for a given scattered node set in $\mathbb{R}^{d>1}$, this method of interpolation may fail to provide a unique interpolant. There is, however, a way around this singularity: the basis $\{\psi_i(\mathbf{x})\}_{i=1}^n$ must be chosen to depend on the data sites. One of the first and most

powerful methods to employ this strategy was the *multiquadric* method developed by Rolland Hardy [12] and the *thin plate spline* method developed by Duchón [2]. The generalization of Hardy and Duchón's methods have now become known as the radial basis function (RBF) method. This technique chooses the basis for interpolation as shifts of a single radially symmetric kernel. In its most simple form, the basis is given by $\{\psi_i(\mathbf{x})\}_{i=1}^n = \{\phi(\|\mathbf{x} - \mathbf{x}_i\|_2)\}_{i=1}^n$, where ϕ is a radially symmetric kernel.

In the early 1990s, there was an explosion of research on both theory and application of RBFs. In terms of applications, one of the most important results was Kansa's work showing the RBF method could be used for solving partial differential equations (PDEs) [14]. Since Kansa's pioneering work, many advances have been made on using RBFs for PDEs (see for example [3, 5, 6, 13, 16, 19]). The method is spectrally accurate for solving PDEs on irregular domains; however, it does have some issues. For example, because of its global nature, it requires solving a large, dense, and possibly ill-conditioned linear system. While some efforts have been made to remedy the problems with the global method ([3,15] and references within), there has been no panacea. In the past 8 years, a new local RBF method has emerged that gives up spectral accuracy of the global RBF method; but, in return, leads to sparse, better-condition linear systems and more flexibility at handling non-linear terms. The method is a generalization of the finite-difference method to scattered nodes. The generalization comes from replacing polynomials (or Taylor series) with RBFs to generate the weights in the finite difference formulas. This method is now referred to as the RBF-FD method and was independently pioneered by Wright and Fornberg [26], Wright [25], Shu et. al. [22], Tolstykh [23], and Cecil et. al. [1].

One issue with the RBF-FD method has been how to properly implement Neumann or Robin boundary conditions so that "one-sided" stencils can be avoided.

In this thesis, we propose a general method for implementing Neumann boundary conditions and study the accuracy of the resulting method in terms of Poisson's equation in 2-D and 3-D.

1.1 RBF Interpolation

1.1.1 Global RBF Interpolation

In this section, we briefly review the RBF method for interpolation since it forms the foundation for the RBF-FD method. In the RBF method, a linear combination of shifts of a radially symmetric kernel ϕ is used to construct an interpolant to a given set of data as follows. Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a set of distinct data sites and $f|_X = \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$ be a set of corresponding function values, then the RBF interpolant to $f|_X$ is given by

$$s(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|), \tag{1.2}$$

where $\{\lambda_1, \ldots, \lambda_n\}$ are determined from the interpolation conditions $s(\mathbf{x}_i) = f(\mathbf{x}_i)$, $i = 1, \ldots, n$. These conditions lead to the following linear system for determining λ_i:

$$\underbrace{\begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_n\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_2 - \mathbf{x}_n\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_n - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_n - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_n - \mathbf{x}_n\|) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix}}_{\lambda} = \underbrace{\begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{bmatrix}}_{\mathbf{f}}.$$

The non-singularity of this system (invertability of A) is dependent on the radial

kernel used. Table 1.1 lists several popular choices for ϕ. For the first 3 of these, the matrix A, is guaranteed to be positive definite, and for the next 2 the matrix is guaranteed to be invertible. For the last 2, there is a possibility the matrix can be singular. However, the interpolant (1.2) can be augmented with side conditions to avoid this issue (see [18] for a full discussion). Some radial kernels feature a shape parameter ϵ, which is used to flatten or sharpen the kernel in order to improve the accuracy of the interpolant. Choosing the right shape parameter is a difficult problem and typically depends on the application [9]. More can be found on the shape parameter in [4, 8, 10, 21, 24].

In our application of the RBF method to generating finite differences, we have found that more accurate results are obtained when the underlying interpolant is modified slightly from (1.2). We add a constant to the interpolant giving

$$s(x) = \sum_{i=1}^{n} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) + \lambda_{n+1}.$$

With a new unknown constant, we need a new constraint for the interpolation system. This is given by

$$\sum_{i=1}^{n} \lambda_i = 0.$$

We choose this because it will yeild a symetric linear system for determining the expansion coefficients.

Table 1.1: Examples of popular choices of radial kernels. The free parameter ϵ is called the shape parameter.

Types of RBF's	$\phi(r)\ r > 0$
Infinitely Smooth RBF's	
Gaussian	$e^{-(\epsilon r)^2}$
Inverse Quadratic	$\frac{1}{1+(\epsilon r)^2}$
Inverse Multiquadric	$\frac{1}{\sqrt{1+(\epsilon r)^2}}$
Multiquadric	$\sqrt{1+(\epsilon r)^2}$
Piecewise Smooth RBF's	
Linear	r
Cubic	r^3
Thin Plate Spline	$r^2 \log r$

$$\underbrace{\left[\begin{array}{cccc|c} \phi(\|\mathbf{x}_1-\mathbf{x}_1\|) & \phi(\|\mathbf{x}_2-\mathbf{x}_1\|) & \cdots & \phi(\|\mathbf{x}_n-\mathbf{x}_1\|) & 1 \\ \phi(\|\mathbf{x}_1-\mathbf{x}_2\|) & \phi(\|\mathbf{x}_2-\mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_n-\mathbf{x}_2\|) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi(\|\mathbf{x}_1-\mathbf{x}_n\|) & \phi(\|\mathbf{x}_2-\mathbf{x}_n\|) & \cdots & \phi(\|\mathbf{x}_n-\mathbf{x}_n\|) & 1 \\ \hline 1 & 1 & \cdots & 1 & 0 \end{array}\right]}_{A} \underbrace{\left[\begin{array}{c} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ \hline \lambda_{n+1} \end{array}\right]}_{\lambda} = \underbrace{\left[\begin{array}{c} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \\ \hline 0 \end{array}\right]}_{\mathbf{f}}. \tag{1.3}$$

1.1.2 Local Interpolation Using RBFs

Creating local scattered data interpolants is similar to that of the global technique. For example, one can construct an interpolant centered about $\mathbf{x}_1$ that only involves the m closest data values to $\mathbf{x}_1$ as follows. First, sort the data set X based on their distance from $\mathbf{x}_1$ and take the m-closest points. This creates a new, smaller, data set, call it $X_1 = \{\mathbf{x}_1 \dots \mathbf{x}_m\}$. Next, we can use the same technique as discussed above for finding the interpolation coefficients λ_i. The resulting system for determining local

interpolation coefficients would look like the following:

$$A^{(1)}\lambda^{(1)} = f|_{X_1}.$$

We can then evaluate this local interpolant at values near x_1 as follows:

$$s^{(1)}(\mathbf{x}) = \sum_{i=1}^{m} \lambda_i^{(1)} \phi(\|\mathbf{x} - \mathbf{x}_i\|) + \lambda_{m+1}^{(1)}$$

$$s^{(1)}(\mathbf{x}) = \begin{bmatrix} \phi(\|\mathbf{x} - \mathbf{x}_1\|) & \dots & \phi(\|\mathbf{x} - \mathbf{x}_m\|) & 1 \end{bmatrix} \lambda^{(1)}$$

$$s^{(1)}(\mathbf{x}) = \begin{bmatrix} \phi(\|\mathbf{x} - \mathbf{x}_1\|) & \dots & \phi(\|\mathbf{x} - \mathbf{x}_m\|) & 1 \end{bmatrix} (A^{(1)})^{-1} f|_{X_1}.$$

1.2 RBF Finite Differences

Similar to how we derived the local interpolation weights in Section 1.1.2, we will calculate finite difference weights for approximating derivatives locally. First, we find $X_1 = \{\mathbf{x}_1 \dots \mathbf{x}_m\}$ as we did in Section 1.1.2. Given the sorted data set, we wish to find a vector D such that $Df|_{X_1} \approx \mathcal{D}_{\mathbf{x}_1} f(\mathbf{x}_1)$, where $\mathcal{D}_{\mathbf{x}_1}$ is any derivative operator taken with respect to $\mathbf{x}_1$. Consider the interpolation weights $\lambda = A^{-1} f|_{X_1}$ from (1.3). Now consider the derivative of the interpolant

$$\mathcal{D}_{\mathbf{x}} s(\mathbf{x})|_{x=x_1} = \sum_{i=1}^{m} \lambda_i \mathcal{D}_{\mathbf{x}} \phi(\|\mathbf{x} - \mathbf{x}_i\|) \approx \mathcal{D}_{\mathbf{x}} f(\mathbf{x}),$$

which we put in vector format

$$\underbrace{\begin{bmatrix} \mathcal{D}_{\mathbf{x}_1}\phi(\|\mathbf{x}_1-\mathbf{x}_1\|) & \mathcal{D}_{\mathbf{x}_1}\phi(\|\mathbf{x}_1-\mathbf{x}_2\|) & \cdots & \mathcal{D}_{\mathbf{x}_1}\phi(\|\mathbf{x}_1-\mathbf{x}_m\|) & 0 \end{bmatrix}}_{B} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \\ \lambda_{n+1} \end{bmatrix} = \mathcal{D}_{\mathbf{x}_1} f(\mathbf{x}_1).$$

Then, $B\lambda = \mathcal{D}_{\mathbf{x}_1} f(\mathbf{x}_1)$ and $BA^{-1} f|_{X_1} = \mathcal{D}_{\mathbf{x}_1} f(\mathbf{x}_1)$. Now we see that $D = BA^{-1}$ [7, 26].

1.3 Outline of Thesis

Through the remainder of this thesis we will introduce Hermite-Birkoff interpolation. We will use this technique to introduce a new fictitious point method using RBFs and how it can be used to implement Neumann boundary conditions. Lastly, we will numerically experiment with this technique and compare it to analytic solutions and solutions for structured data.

CHAPTER 2

FICTITIOUS POINT METHOD FOR ENFORCING BOUNDARY CONDITIONS

Here we develop a fictitious point method for enforcing Neumann-type boundary conditions within the RBF-FD method. Fictitious point methods are standard practice in the regular finite difference methods. In this context, the formulas are derived using symmetry conditions and Taylor series. Since these symmetries cannot be exploited with scattered data, we derive a fictitious point method using the Hermite-Birkhoff method for interpolating scattered data [27].

2.1 Hermite-Birkhoff RBF Interpolation

A Hermite-Birkhoff interpolation problem is one where a mix of function and derivative values are known at a given set of nodes $X = \{x_1, \ldots, x_n\}$. Consider, for example, Figure 2.1 where we wish to interpolate function values at $X_I = \{\mathbf{x}_1, \ldots, \mathbf{x}_4\}$ and derivative values at $X_D = \{\mathbf{x}_5, \mathbf{x}_6\}$. One solution to this problem is known as the symmetric Hermite-Birkhoff RBF interpolant and was first proposed by Wu [27] (see also [20]). This method looks for an interpolant to the data of the form

$$S(\mathbf{x}) = \sum_{i=1}^{4} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) + \sum_{j=5}^{6} \alpha_j \mathcal{D}_{\mathbf{x}_j} \phi(\|\mathbf{x} - \mathbf{x}_j\|) + \beta, \tag{2.1}$$

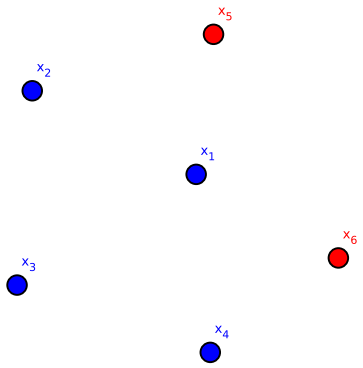


Figure 2.1: An example stencil known function values at $\mathbf{x}_1, \ldots, \mathbf{x}_4$ and derivative values at $\mathbf{x}_5, \mathbf{x}_6$.

where $\mathcal{D}_{\mathbf{x}_j}$ is the derivative operator applied to $\phi(\|\mathbf{x} - \mathbf{x}_j\|)$ with respect to $\mathbf{x}_j$. To determine λ_i, α_j, and β, we impose the constraints that $S(\mathbf{x})$ will fit our known function/derivative values:

$$\begin{aligned} S(\mathbf{x}_i) &= f(\mathbf{x}_i), \qquad \text{for} \quad i = 1, \ldots, 4 \\ \mathcal{D}_{\mathbf{x}_j} S(\mathbf{x}_j) &= \mathcal{D}_{\mathbf{x}_j} f(\mathbf{x}_j), \quad \text{for} \quad j = 5, 6 \\ \textstyle\sum_{i=1}^{4} \lambda_i &= 0. \end{aligned}$$

As in standard RBF interpolation, we impose $\sum_{i=1}^{4} \lambda_i = 0$ in order to keep a symmetric system. We can create a linear system for finding these constraints as follows. First, we define

$$A = \begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_4\|) \\ \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_4 - \mathbf{x}_1\|) & \cdots & \phi(\|\mathbf{x}_4 - \mathbf{x}_4\|) \end{bmatrix}$$

$$D = \begin{bmatrix} \mathcal{D}_{\mathbf{x}_5}\phi(\|\mathbf{x}_1 - \mathbf{x}_5\|) & \mathcal{D}_{\mathbf{x}_6}\phi(\|\mathbf{x}_1 - \mathbf{x}_6\|) \\ \mathcal{D}_{\mathbf{x}_5}\phi(\|\mathbf{x}_2 - \mathbf{x}_5\|) & \mathcal{D}_{\mathbf{x}_6}\phi(\|\mathbf{x}_2 - \mathbf{x}_6\|) \\ \vdots & \vdots \\ \mathcal{D}_{\mathbf{x}_5}\phi(||\mathbf{x}_4 - \mathbf{x}_5||) & \mathcal{D}_{\mathbf{x}_6}\phi(||\mathbf{x}_4 - \mathbf{x}_6||) \end{bmatrix}$$

$$E = \begin{bmatrix} \mathcal{D}_{\mathbf{x}_5}\mathcal{D}_{\mathbf{x}_5}\phi(\|\mathbf{x}_5 - \mathbf{x}_5\|) & \mathcal{D}_{\mathbf{x}_5}\mathcal{D}_{\mathbf{x}_6}\phi(\|\mathbf{x}_5 - \mathbf{x}_6\|) \\ \mathcal{D}_{\mathbf{x}_6}\mathcal{D}_{\mathbf{x}_5}\phi(\|\mathbf{x}_6 - \mathbf{x}_5\|) & \mathcal{D}_{\mathbf{x}_6}\mathcal{D}_{\mathbf{x}_6}\phi(\|\mathbf{x}_6 - \mathbf{x}_6\|) \end{bmatrix}.$$

The linear system for determining the interpolation coefficients is then given by

$$\begin{bmatrix} A & D & \mathbf{1} \\ D^T & E & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathcal{D}\mathbf{f} \\ 0 \end{bmatrix}.$$

Note that this creates a symmetric system that can be solved for the interpolation weights λ and α. This system is guaranteed to be non-singular for the first 4 radial kernels listed in Table 1.1 and is called the symmetric Hermite-RBF interpolant [20, 27]. A non-symmetric Hermite-RBF interpolation can be created by altering the second constraint.

To evaluate at points not in X, say at points $\mathbf{x}_7, \mathbf{x}_8$, we create matrices

$$B = \begin{bmatrix} \phi(||\mathbf{x}_7 - \mathbf{x}_1||) & \phi(||\mathbf{x}_7 - \mathbf{x}_2||) & \cdots & \phi(||\mathbf{x}_7 - \mathbf{x}_4||) \\ \phi(||\mathbf{x}_8 - \mathbf{x}_1||) & \phi(||\mathbf{x}_8 - \mathbf{x}_2||) & \cdots & \phi(||\mathbf{x}_8 - \mathbf{x}_4||) \end{bmatrix}$$

$$C = \begin{bmatrix} \mathcal{D}_{\mathbf{x}_5}\phi(||\mathbf{x}_7 - \mathbf{x}_5||) & \mathcal{D}_{\mathbf{x}_6}\phi(||\mathbf{x}_7 - \mathbf{x}_6||) \\ \mathcal{D}_{\mathbf{x}_6}\phi(||\mathbf{x}_8 - \mathbf{x}_6||) & \mathcal{D}_{\mathbf{x}_6}\phi(||\mathbf{x}_8 - \mathbf{x}_6||) \end{bmatrix}.$$

We can then evaluate (2.1) as follows

$$\begin{bmatrix} B & C & \mathbf{1} \end{bmatrix} \begin{bmatrix} A & D & \mathbf{1} \\ D^T & E & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & 0 \end{bmatrix}^{-1} \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_4) \\ \mathcal{D}_{\mathbf{x}_5} f(\mathbf{x}_6) \\ \mathcal{D}_{\mathbf{x}_7} f(\mathbf{x}_7) \\ 0 \end{bmatrix} \approx \begin{bmatrix} f(\mathbf{x}_7) \\ f(\mathbf{x}_8) \end{bmatrix} \tag{2.2}$$

2.2 General Fictitious Point Method for Discretizing the Laplacian Near Boundaries

We now discuss our fictitious point method for solving Poisson's equation with either Neumann, Dirichlet, or mixed boundary conditions on an irregular domain. Poisson's equation is given by:

$$\nabla^2 u = f(x),$$

with Dirichlet conditions

$$u(x_b) = g(x_b),$$

or Neumann conditions

$$\mathbf{n}\dot{\nabla} u(x_b) = h(x_b),$$

where x_b are points on the boundary. Before we discuss the procedure and the techniques for discretizing the Laplacian, we define the different types of RBF stencils.

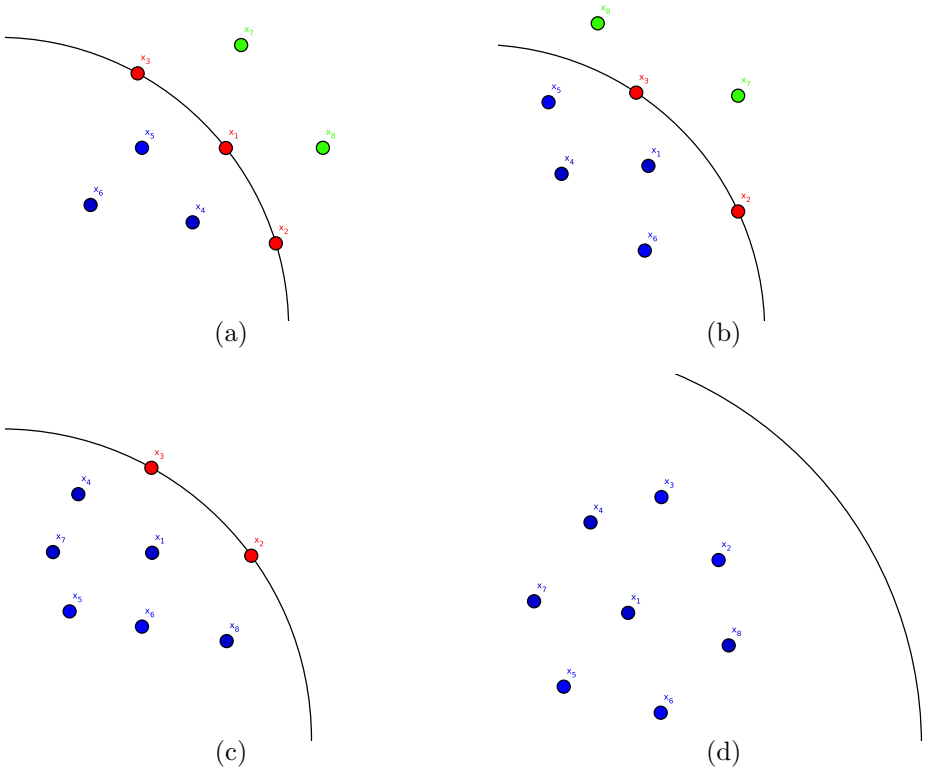


Figure 2.2: Plots in (a)-(d) illustrate 4 types of stencils that can occur. Blue corresponds interior points, red to boundary points, and green to fictitious points. The center of the stencil is always x_1.

2.2.1 Important Definitions

In Figure, 2.2 we illustrate 4 different types of RBF-FD Stencils that can occur in a bounded domain. We define the different points that occur in these stencils as follows.

Definition 1. *Boundary Point : A point in the domain is a boundary point if it is on the boundary of the continuous domain. For example, if the domain is the unit disk, then the point* $\mathbf{x} = (x, y)$ *is on the boundary if* $x^2 + y^2 = 1$. *See Figure 2.2(a).*

Definition 2. *Purely Interior Point : A point is a purely interior point if it has a stencil that contains only interior or boundary points. See Figure 2.2(c) 2.2(d).*

Definition 3. *Edge Point: A point is an edge point if it is not on the boundary, but its corresponding RBF-FD stencil contains at least one fictitious point. See Figure 2.2(b).*

Definition 4. *Fictitious Point: A point is a fictitious point if it is not a boundary, purely interior, or edge point. Fictitious points lie on the outside of the domain. See Figure 2.2(b) 2.2(a) (green points).*

2.2.2 RBF-FD Approximation for a Purely Interior Node

Consider the purely interior stencil as seen in Figure 2.2(c). We apply the standard RBF-FD method to discretize the Laplacian at $\mathbf{x}_1$ ([6, 23]). This gives the following expression for RBF-FD weights:

$$L^T \equiv \begin{bmatrix} \phi(||\mathbf{x}_1-\mathbf{x}_1||) & \phi(||\mathbf{x}_1-\mathbf{x}_2||) & \cdots & \phi(||\mathbf{x}_1-\mathbf{x}_8||) & 1 \\ \phi(||\mathbf{x}_2-\mathbf{x}_1||) & \phi(||\mathbf{x}_2-\mathbf{x}_2||) & \cdots & \phi(||\mathbf{x}_2-\mathbf{x}_8||) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi(||\mathbf{x}_8-\mathbf{x}_1||) & \phi(||\mathbf{x}_8-\mathbf{x}_6||) & \cdots & \phi(||\mathbf{x}_8-\mathbf{x}_8||) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \nabla^2\phi(||\mathbf{x}_1-\mathbf{x}_1||) \\ \nabla^2\phi(||\mathbf{x}_1-\mathbf{x}_2||) \\ \nabla^2\phi(||\mathbf{x}_1-\mathbf{x}_3||) \\ \vdots \\ \nabla^2\phi(||\mathbf{x}_1-\mathbf{x}_8||) \\ 0 \end{bmatrix}$$

The Laplacian of a function at $\mathbf{x}_1$ can then be approximated as

$$\begin{bmatrix} L_1 & L_2 & L_3 & L_4 & L_5 & L_6 & L_7 & L_8 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_8 \end{bmatrix} \approx \nabla^2 u(\mathbf{x}_1). \tag{2.3}$$

Note that this is the method used for computing the RBF-FD weights for all the nodes. However, depending on the node type, some modifications are necessary, as discussed in the following sections.

2.2.3 RBF-FD Approximation for a Boundary Node

In this section, we derive an approximation of the Laplacian for a boundary point. We only consider the case of Neumann boundary conditions as no approximations are needed at the boundaries for Dirichlet boundary conditions. The main idea is to use Hermite-Birkhoff interpolation to fictitious points and enforce the derivative conditions at the boundaries as the Hermite-Birkhoff conditions. Consider the boundary stencil displayed in Figure 2.2(a). Nodes 1-3 are on the boundary, nodes 4-6 are on the interior of the domain, and 7 and 8 are fictitious points. We first discretize the Laplacian about $\mathbf{x}_1$ on this stencil so we have RBF-FD weights L_i as in Equation (2.3) such that, for function values u_i,

$$\begin{bmatrix} L_1 & \cdots & L_8 \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_8 \end{bmatrix} \approx \nabla^2 u(\mathbf{x}_1).$$

Consider splitting this system into a sum of individual components:

$$L_1u_1 + L_2u_2 + \cdots + L_6u_6 + L_7u_7 + L_8u_8 \approx \nabla^2 u(\mathbf{x}_1). \tag{2.4}$$

Now u_7 and u_8 are unknown so we need to replace them with an extrapolated value. This is where we will use the boundary condition. We use Hermite-Birkhoff RBF interpolation to find values for u_7 and u_8, with the Neumann derivative operator as the Hermite-Birkhoff condition. The Hermite-Birkhoff interpolant is given by

$$S(\mathbf{x}) = \sum_{i=1}^{6} \lambda_i \phi(||\mathbf{x} - \mathbf{x}_i||) + \sum_{j=1}^{3} \alpha_j \mathcal{D}_{\mathbf{x}_j} \phi(||\mathbf{x} - \mathbf{x}_j||) + \beta$$

with $\mathcal{D}_{\mathbf{x}_j}$ being the Neumann operator applied to the RBF with respect to $\mathbf{x}_j$. We apply the Hermite-Birkhoff method for interpolation discussed in the first section of this chapter with $X_I = \{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$ and $X_D = \{\mathbf{x}_4, \ldots, \mathbf{x}_6\}$. Consider the first matrix-vector multiplication in (2.2):

$$\begin{bmatrix} B & C & \mathbf{1} \end{bmatrix} \begin{bmatrix} A & D & \mathbf{1} \\ D^T & E & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & 0 \end{bmatrix}^{-1} = \begin{bmatrix} W_{7,1} & W_{7,2} & \cdots & W_{7,6} & \tilde{W}_{7,1} & \cdots & \tilde{W}_{7,3} & T_1 \\ W_{8,1} & W_{8,2} & \cdots & W_{8,6} & \tilde{W}_{8,1} & \cdots & \tilde{W}_{8,3} & T_2 \end{bmatrix} = [\mathbf{W}, \tilde{\mathbf{W}}, \mathbf{T}],$$

where T_1, T_2, and $\mathbf{T}$ are a result of the constant β in the interpolant. Since in (2.2) T_1 and T_2 are multiplied by 0, they will no longer need to be used. We can then evaluate at $\mathbf{x}_7$ and $\mathbf{x}_8$ using

$$\begin{bmatrix} W_{7,1} & W_{7,2} & \cdots & W_{7,6} & \tilde{W}_{7,1} & \cdots & \tilde{W}_{7,3} \\ W_{8,1} & W_{8,2} & \cdots & W_{8,6} & \tilde{W}_{8,1} & \cdots & \tilde{W}_{8,3} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_6 \\ \mathcal{D}_{\mathbf{x}_1} u(\mathbf{x}_1) \\ \mathcal{D}_{\mathbf{x}_2} u(\mathbf{x}_2) \\ \mathcal{D}_{\mathbf{x}_3} u(\mathbf{x}_3) \end{bmatrix} \approx \begin{bmatrix} u_7 \\ u_8 \end{bmatrix}$$

We can replace u_7 and u_8 in (2.4) with the interpolant above; i.e., (2.4) becomes

$$L_1 u_1 + L_2 u_2 + \cdots + L_6 u_6 + L_7 \begin{bmatrix} W_{7,1} \\ W_{7,2} \\ \vdots \\ W_{7,6} \\ \tilde{W}_{7,1} \\ \vdots \\ \tilde{W}_{7,3} \end{bmatrix}^T \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_6 \\ \mathcal{D}_{\mathbf{x}_1} u(\mathbf{x}_1) \\ \mathcal{D}_{\mathbf{x}_2} u(\mathbf{x}_2) \\ \mathcal{D}_{\mathbf{x}_3} u(\mathbf{x}_3) \end{bmatrix} + L_8 \begin{bmatrix} W_{8,1} \\ W_{8,2} \\ \vdots \\ W_{8,6} \\ \tilde{W}_{8,1} \\ \vdots \\ \tilde{W}_{8,3} \end{bmatrix}^T \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_6 \\ \mathcal{D}_{\mathbf{x}_1} u(\mathbf{x}_1) \\ \mathcal{D}_{\mathbf{x}_2} u(\mathbf{x}_2) \\ \mathcal{D}_{\mathbf{x}_3} u(\mathbf{x}_3) \end{bmatrix}.$$

Expanding this expression gives

$$L_1 u_1 + L_2 u_2 + \cdots + L_6 u_6 + L_7 (W_{7,1} u_1 + W_{7,2} u_2 + \cdots + W_{7,6} u_6) + L_7 (\tilde{W}_{7,1} \mathcal{D}_{\mathbf{x}_1} u_1 + \cdots + \tilde{W}_{7,3} \mathcal{D}_{\mathbf{x}_1} u_3)$$

$$+ \ldots + L_8 (W_{8,1} u_1 + W_{8,2} u_2 + \cdots + W_{8,6} u_6) + L_8 (\tilde{W}_{8,1} \mathcal{D}_{\mathbf{x}_1} u_1 + \cdots + \tilde{W}_{8,3} \mathcal{D}_{\mathbf{x}_3} u_3) \approx \nabla^2 u(\mathbf{x}_1).$$

We can then write (2.4) as:

$$\begin{bmatrix} (L_1 + L_7W_{7,1} + L_8W_{8,1}) & (L_2 + L_7W_{7,2} + L_8W_{8,2}) & \cdots & (L_6 + L_7W_{7,6} + L_8W_{8,6}) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_6 \end{bmatrix} +$$

$$\begin{bmatrix} (L_7\tilde{W}_{7,1} + L_8\tilde{W}_{8,1}) & (L_7\tilde{W}_{7,2} + L_8\tilde{W}_{8,2}) & (L_7\tilde{W}_{7,3} + L_8\tilde{W}_{8,3}) \end{bmatrix} \begin{bmatrix} \mathcal{D}_{\mathbf{x}_1}u(\mathbf{x}_1) \\ \mathcal{D}_{\mathbf{x}_2}u(\mathbf{x}_2) \\ \mathcal{D}_{\mathbf{x}_3}u(\mathbf{x}_3) \end{bmatrix} \approx \nabla^2 u(\mathbf{x}_1).$$

Once we compute $L_1, \ldots, Ł_8, W_{7,1}, \ldots, W_{7,6}, W_{8,1}, \ldots, W_{8,6}, \tilde{W}_{7,1}, \ldots, \tilde{W}_{7,3}, \tilde{W}_{8,1}, \ldots, \tilde{W}_{8,3}$, we can approximate $\nabla^2 u(\mathbf{x}_1)$ for any values of $u_1, \ldots, u_6$ and $\mathcal{D}_{\mathbf{x}_1}u(\mathbf{x}_1), \ldots, \mathcal{D}_{\mathbf{x}_3}u(\mathbf{x}_3)$.

2.2.4 Edge Point

We consider both Dirichlet and Neumann boundary conditions when developing an approximation for an edge point. Here, both Dirichlet and Neumann conditions require approximation (since we are not on the boundary). Let us start with Dirichlet conditions.

Dirichlet Boundary Conditions

For Dirichlet conditions, we do not need to enforce any extra derivatives on the boundary. However, we still have a choice to make. Since our stencil contains fictitious points, we could use the Hermite-Birkhoff method (as done with boundary points) and use the Laplacian as the Hermite condition since $\nabla^2 u = f$ has to hold on the boundary. The other option is to replace the fictitious points in the stencil with non-fictitious points, which turns the point into a purely interior point. The disadvantage

of the latter is that it results in a one-sided stencil. Regardless, we now only need to consider the first option since the second is discussed in Section (2.2.2).

To this end, consider the edge point stencil in Figure (2.2(b)). Nodes 2-3 are on the boundary, nodes 1 and 4-6 are on the interior of the domain, and 7 and 8 are fictitious points. We first discretize the Laplacian on this stencil. So, we have Laplacian weights L_i and function values u_i

$$\begin{bmatrix} L_1 & \cdots & L_8 \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_8 \end{bmatrix} \approx \nabla^2 u(\mathbf{x}_1) \tag{2.5}$$

Now, we split this system into a sum of individual components as in Equation (2.4). However, u_7 and u_8 are unknown, so we need to replace them with an extrapolated value. We will use Hermite-Birkhoff RBF method to find values for u_7 and u_8. The interpolant is given by

$$S(\mathbf{x}) = \sum_{i=1}^{6} \lambda_i \phi(||\mathbf{x} - \mathbf{x}_i||) + \sum_{j=2}^{3} \alpha_j \mathcal{D}_{\mathbf{x}_j} \phi(||\mathbf{x} - \mathbf{x}_j||) + \beta \tag{2.6}$$

Note that since $\mathcal{D}_{\mathbf{x}_j} = \nabla^2_{\mathbf{x}_j}$, it does not matter if it is applied to $\mathbf{x}$ or $\mathbf{x}_j$ since both the Laplacian and ϕ are radially symmetric. Thus, we can say $\mathcal{D}_{\mathbf{x}_j} = \nabla^2$. In this case, $X_I = \{\mathbf{x}_1, \mathbf{x}_4, \ldots, \mathbf{x}_6\}$ and $X_D = \{\mathbf{x}_4, \ldots, \mathbf{x}_6\}$ and we find the Hermite-Birkhoff interpolation weights $[\mathbf{W}, \tilde{\mathbf{W}}]$ as in Section (2.2.3), but with the proper Hermite-Birkhoff constraints ($\mathcal{D}_{\mathbf{x}_j} = \nabla^2$). If we now go back to the weights for the Laplacian (2.5), we can replace u_7 and u_8 with the interpolant (2.6).

$$L_1u_1+L_2u_2+\cdots+L_6u_6+L_7\begin{bmatrix} W_{7,1} \\ W_{7,2} \\ \vdots \\ W_{7,6} \\ \tilde{W}_{7,2} \\ \tilde{W}_{7,3} \end{bmatrix}^T \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_6 \\ \nabla^2 u(\mathbf{x}_2) \\ \nabla^2 u(\mathbf{x}_3) \end{bmatrix} + L_8 \begin{bmatrix} W_{8,1} \\ W_{8,2} \\ \vdots \\ W_{8,6} \\ \tilde{W}_{8,2} \\ \tilde{W}_{8,3} \end{bmatrix}^T \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_6 \\ \nabla^2 u(\mathbf{x}_2) \\ \nabla^2 u(\mathbf{x}_3) \end{bmatrix} \approx \nabla^2 u(\mathbf{x}_1)$$

Expanding this expression gives

$$L_1u_1+L_2u_2+\cdots+Ł_6u_6+L_7(W_{7,1}u_1+W_{7,2}u_2+\cdots+W_{7,6}u_6)+L_7(\tilde{W}_{7,2}\nabla^2u(\mathbf{x}_2)+\tilde{W}_{7,3}\nabla^2u(\mathbf{x}_3))$$

$$+\ldots+L_8(W_{8,1}u_1+W_{8,2}u_2+\cdots+W_{8,6}u_6)+L_8(\tilde{W}_{8,2}\nabla^2u(\mathbf{x}_2)+\tilde{W}_{8,3}\nabla^2u(\mathbf{x}_3)) \approx \nabla^2u(\mathbf{x}_1),$$

and we can write (2.5) as

$$\begin{bmatrix} (L_1 + L_7W_{7,1} + L_8W_{8,1}) & (L_2 + L_7W_{7,2} + L_8W_{8,2}) & \cdots & (L_6 + L_7W_{7,6} + L_8W_{8,6}) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_6 \end{bmatrix} +$$

$$\begin{bmatrix} (L_7\tilde{W}_{7,2} + L_8\tilde{W}_{8,2}) & (L_7\tilde{W}_{7,3} + L_8\tilde{W}_{8,3}) \end{bmatrix} \begin{bmatrix} \nabla^2 u(\mathbf{x}_2) \\ \nabla^2 u(\mathbf{x}_3) \end{bmatrix} \approx \nabla^2 u(\mathbf{x}_1).$$

Now that we have obtained weights $L_1, \ldots, Ł_8, W_{7,1}, \ldots, W_{7,6}, W_{8,1}, \ldots, W_{8,6}, \tilde{W}_{7,2}, \tilde{W}_{7,3}, \tilde{W}_{8,2}, \tilde{W}_{8,}$ we can approximate $\nabla^2 u(\mathbf{x}_1)$ for any values of $u_1, \ldots, u_6$ and $\mathcal{D}_{\mathbf{x}_1}u(\mathbf{x}_2), \mathcal{D}_{\mathbf{x}_3}u(\mathbf{x}_3)$.

Neumann Boundary Conditions

Here we will use the same stencil as in the Dirichlet case above. So consider the edge point stencil in Figure 2.2(b). We could change the stencil to make the point a purely interior point, however we wish to enforce the boundary condition wherever possible. So let us consider the Hermite-Birkhoff extrapolation technique again. Only now, we use the Neumann operator as the Hermite condition (denoted below as $\mathcal{D}$). The only difference between this scenario and the approximation for a boundary point is the center of the stencil. In this case, the center is *not* on the boundary. Instead, nodes 2-3 are on the boundary, nodes 1 and 4-6 are on the interior of the domain, and 7 and 8 are fictitious points. We again discretize the Laplacian on this stencil as in (2.4). However, now u_7 and u_8 are unknown, so we need to replace them with an extrapolated value. Using Hermite-Birkhoff RBF interpolation, the interpolant is given by:

$$S(\mathbf{x}) = \sum_{i=1}^{6} \lambda_i \phi(||\mathbf{x} - \mathbf{x}_i||) + \sum_{j=2}^{3} \alpha_j \mathcal{D}_{\mathbf{x}_j} \phi(||\mathbf{x} - \mathbf{x}_j||) + \beta$$

Once more, $X_i = \{\mathbf{x}_1, \mathbf{x}_4, \ldots, \mathbf{x}_6\}$ and $X_b = \{\mathbf{x}_2, \mathbf{x}_3\}$ and we find $[\mathbf{W}, \tilde{\mathbf{W}}]$ and replace the unknown function values in (2.4). We replace u_7 and u_8 with the Hermite-Birkhoff interpolant,

$$L_1u_1+L_2u_2+\cdots+L_6u_6+L_7 \begin{bmatrix} W_{7,1} \\ W_{7,2} \\ \vdots \\ W_{7,6} \\ \tilde{W}_{7,2} \\ \tilde{W}_{7,3} \end{bmatrix}^T \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_6 \\ \mathcal{D}_{\mathbf{x}_2} u(\mathbf{x}_2) \\ \mathcal{D}_{\mathbf{x}_3} u(\mathbf{x}_3) \end{bmatrix} + L_8 \begin{bmatrix} W_{8,1} \\ W_{8,2} \\ \vdots \\ W_{8,6} \\ \tilde{W}_{8,2} \\ \tilde{W}_{8,3} \end{bmatrix}^T \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_6 \\ \mathcal{D}_{\mathbf{x}_2} u(\mathbf{x}_2) \\ \mathcal{D}_{\mathbf{x}_3} u(\mathbf{x}_3) \end{bmatrix} \approx \nabla^2 u(\mathbf{x}_1).$$

Expanding this expression gives

$$L_1u_1+L_2u_2+\cdots+Ł_6u_6+L_7(W_{7,1}u_1+W_{7,2}u_2+\cdots+W_{7,6}u_6)+L_7(\tilde{W}_{7,2}\mathcal{D}_{\mathbf{x}_2}u(\mathbf{x}_2)+\tilde{W}_{7,3}\mathcal{D}_{\mathbf{x}_3}u(\mathbf{x}_3))+$$

$$\ldots L_8(W_{8,1}u_1+W_{8,2}u_2+\cdots+W_{8,6}u_6)+L_8(\tilde{W}_{8,2}\mathcal{D}_{\mathbf{x}_2}u(\mathbf{x}_2)+\tilde{W}_{8,3}\mathcal{D}_{\mathbf{x}_3}u(\mathbf{x}_3))\approx\nabla^2u(\mathbf{x}_1),$$

and we can write (2.4) as

$$\begin{bmatrix} (L_1+L_7W_{7,1}+L_8W_{8,1}) & (L_2+L_7W_{7,2}+L_8W_{8,2}) & \cdots & (L_6+L_7W_{7,6}+L_8W_{8,6}) \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_6 \end{bmatrix}+$$

$$\begin{bmatrix} (L_7\tilde{W}_{7,1}+L_8\tilde{W}_{8,1}) & (L_7\tilde{W}_{7,2}+L_8\tilde{W}_{8,2}) & (L_7\tilde{W}_{7,3}+L_8\tilde{W}_{8,3}) \end{bmatrix}\begin{bmatrix} \mathcal{D}_{\mathbf{x}_2}u(\mathbf{x}_2) \\ \mathcal{D}_{\mathbf{x}_3}u(\mathbf{x}_3) \end{bmatrix}\approx\nabla^2u(\mathbf{x}_1).$$

We have now computed weights for approximating $\nabla^2u(\mathbf{x}_1)$ for any values of $u_1,\ldots,u_6$ and $\mathcal{D}_{\mathbf{x}_1}u(\mathbf{x}_2),\mathcal{D}_{\mathbf{x}_3}u(\mathbf{x}_3)$.

2.3 Method for Solving Poisson's Equation

We have seen several scenarios where we discretize the Laplacian for different points in the domain. Notice that each section above has some row vector of weights that we multiply by function values to get an approximation. We go through each point in the domain and find these weights, indexing and ordering them in a square sparse matrix L. However, in the scenarios that had Hermite-Birkhoff interpolation used, we had two row vectors. We must index and order a second sparse matrix ($\tilde{L}$) for

multiplication by the Hermite conditions. This results in 2 matrices, a square matrix L and non-square matrix $\tilde{L}$, such that

$$\left[\begin{array}{c} L \end{array}\right]\left[\begin{array}{c} u \end{array}\right] + \left[\begin{array}{c} \tilde{L} \end{array}\right]\left[\begin{array}{c} \mathcal{D}_{\mathbf{x}_b} u(\mathbf{x}_b) \end{array}\right] \approx \nabla^2 u \tag{2.7}$$

where u is the solution to Poisson's equation and $\mathcal{D}_{\mathbf{x}_b} u(\mathbf{x}_b)$ is a vector of points where the Hermite condition is being enforced. If we apply this discretization to the Laplacian, we get

$$\left[\begin{array}{c} L \end{array}\right]\left[\begin{array}{c} u(\mathbf{x}_b) \\ u(\mathbf{x}_i) \end{array}\right] + \left[\begin{array}{c} \tilde{L} \end{array}\right]\left[\begin{array}{c} \mathcal{D}u(\mathbf{x}_b) \end{array}\right] = \left[\begin{array}{c} \nabla^2 u \end{array}\right] \tag{2.8}$$

Since $\left[\begin{array}{c} \tilde{L} \end{array}\right]\left[\begin{array}{c} \mathcal{D}u(\mathbf{x}_b) \end{array}\right]$ is known, we can move it to the right-hand side.

$$\left[\begin{array}{c} L \end{array}\right]\left[\begin{array}{c} u(\mathbf{x}_b) \\ u_i \end{array}\right] = \left[\begin{array}{c} \nabla^2 u \end{array}\right] - \left[\begin{array}{c} \tilde{L} \end{array}\right]\left[\begin{array}{c} \mathcal{D}u(\mathbf{x}_b) \end{array}\right]$$

Thus, to determine the approximate RBF-FD solution to Poisson's equation, we need to solve the above *sparse* linear system. We call this method the RBF-FD fictitious point method.

CHAPTER 3

NUMERICAL RESULTS

As mentioned in Chapter 2, we are solving Poisson's equation on an irregular domain Ω. This is given by

$$\nabla^2 u = f(\mathbf{x}) \tag{3.1}$$

and is subjected to the Dirichlet boundary condition

$$u(\mathbf{x}_b) = g(\mathbf{x}_b), \quad \mathbf{x}_b \in \partial\Omega \tag{3.2}$$

or Neumann boundary conditions

$$\mathbf{n} \cdot \nabla u = h(\mathbf{x}_b), \quad \mathbf{x}_b \in \partial\Omega \tag{3.3}$$

where $\mathbf{n}$ is the unit normal vector.

Since the RBF-FD method does not require a grid or mesh, it is possible to discretize the domain in an "optimal" way. To do this, we developed a method based on a minimum energy algorithm and we discuss this algorithm in Appendix A.

In the following experiments, we will use the Gaussian radial kernel from Table 1.1. Additionally, since we will be applying the RBF-FD fictitious point method to solve Poisson's equation, we need analytical formulas for the Laplacian applied to the Gaussian. These are given in [26]:

$$\nabla^2\phi = (d-1) * \frac{\partial\phi(r)}{\partial r} + \frac{\partial^2\phi(r)}{\partial r^2}$$

where d is the dimension of the domain. Our applications include several different domains, an annulus, an ellipse, and a spherical shell. As seen in Section 2.3, we require a matrix inversion for the solution. Here we will be using the MATLAB built in linear system solver \or mldivide together with MATLAB's sparse matrix library.

3.1 Solutions on an Annulus

In this section, we will solve Poisson's equation on an annulus using the RBF-FD fictitious point method. Then, we will compare the results to both the analytical solution and to a standard second order finite difference solution applied on a polar grid. Let j be the third zero of the Bessel function $J_1(r)$ and consider the following Poisson's equation, written in polar form.

$$\nabla^2 u = \frac{1}{r^2} - 8\sin(4\theta)J_0(jr) - j^2\sin(4\theta)J_0(jr)/2 \tag{3.4}$$

defined on the domain $\Omega = \{(x,y)|\frac{1}{2} \leq x^2 + y^2 \leq 1\}$, where $x = r\cos\theta$ and $y = r\sin\theta$. We consider the following Neumann conditions on the exterior boundary $(\partial\Omega : x^2 + y^2 = 1)$

$$\frac{\partial u}{\partial r}(r = 1, \theta) = -j\cos(2\theta)\sin(2\theta)J_1(jr) = 0$$

and Dirichlet conditions on the interior boundary

$$u\left(r = \frac{1}{2}, \theta\right) = \cos(2\theta)\sin(2\theta)J_0(jr) + 1$$

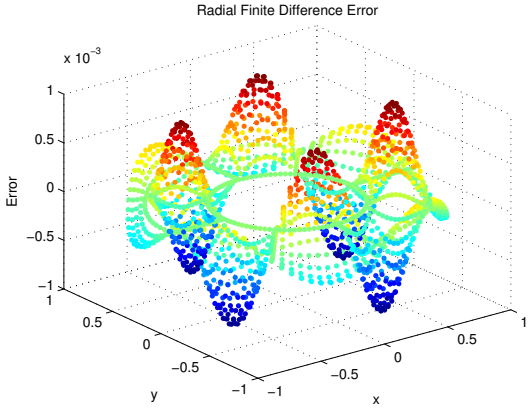


Figure 3.1: The error in the standard second or FD method based on a polar grid

The analytic solution to this equation is

$$\mathbf{n} \cdot \nabla u(r, \theta) = \cos(2\theta) \sin(2\theta) J_0(\frac{1}{2} j) + 1.$$

Recall that in Section 2.2, when given Dirichlet boundary conditions, we had to make a choice. We would either use the fictitious point method with the Laplacian as the Hermite-Birkhoff condition, or use one-sided stencils with the RBF-FD method. In this section, we will test both cases.

3.1.1 Standard Finite Differences in Polar Coordinates

Using equally-spaced polar grid and the standard second order 5-point finite difference formula in polar coordinates, we compute the solution to (3.4). With 2000 nodes, the standard finite difference approximation had a maximum error of $9.1528 * 10^{-4}$. The error for this solution is plotted in Figure 3.1.

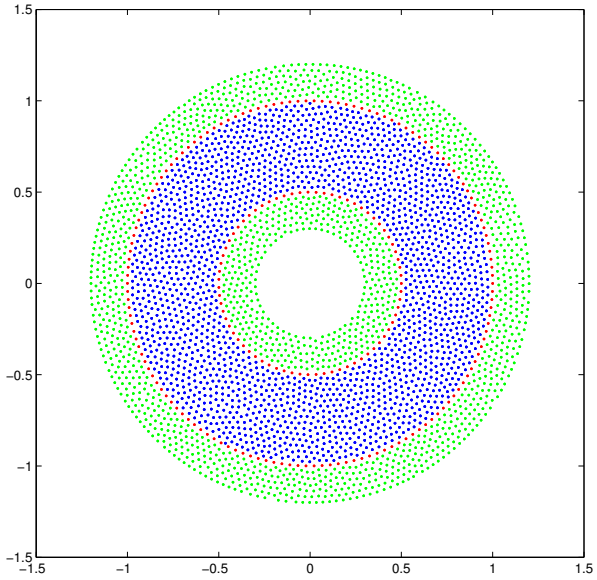
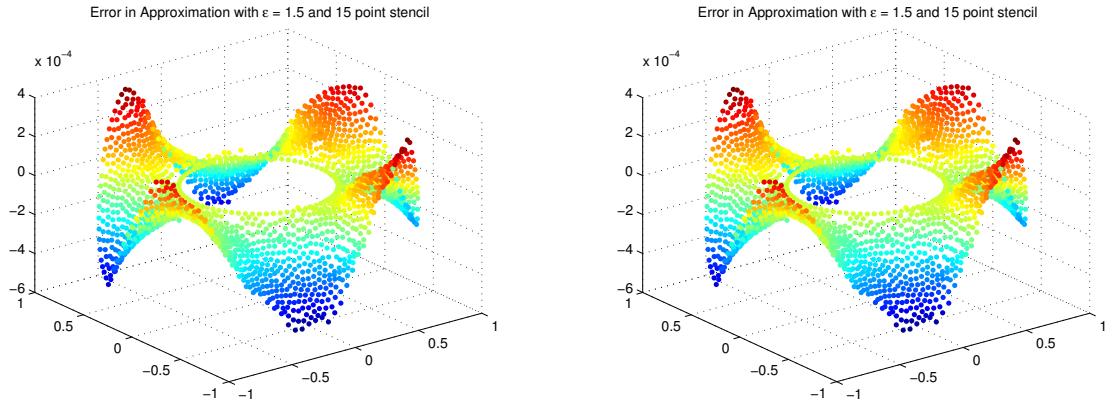


Figure 3.2: The scattered data domain with red points as boundary nodes, blue points as purely interior nodes, and green points as fictitious nodes.



(a) (b)

Figure 3.3: Left - Error using one-sided stencils to Dirichlet points. Right - Error using Hermite interpolation on Dirichlet points

Table 3.1: Errors for variable node set density with one-sided interior boundary

Node Density	Error $\|\|\|_\infty$	Optimal ϵ	Error $\|\|\|_2$	Optimal ϵ
500	0.0014	3.3	6.09E-004	3.2
1000	5.37E-004	3.1	2.03E-004	3.3
2000	1.31E-004	3.3	5.83E-005	3.3
4000	5.33E-005	3.0	1.88E-005	3.2
8000	1.46E-005	3.2	4.08E-006	3.2

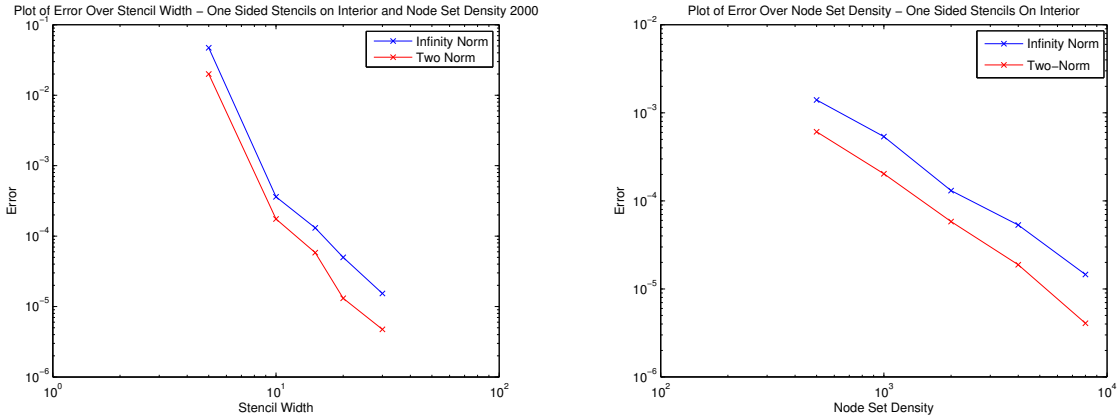
3.1.2 RBF-FD with One Sided-Stencils on Dirichlet Boundary

Now we consider the RBF-FD approximations on the point set seen in Figure 3.2. First, we use the technique described in Section 2.2.4 and choose the one-sided stencils on Dirichlet boundary points. In Figure 3.3(a), we plot the spread of error over the domain. Here our shape parameter was $\epsilon = 1.5$ with a stencil size of 15.

We consider two types of convergence plots for this technique. First, in Figure 3.4(a), we do a convergence plot based on variable stencil size (with fixed node set density of 2000 nodes and fixed $\epsilon = 3.3$). Then, in Figure 3.4(b), we plot the convergence based on variable node set density (with fixed stencil size of 15). This is to show, as in standard finite differences, that as the stencil size increases a better solution is obtained. In the second case, we loop through a vector of possible ϵ and used the error from the optimal shape parameter. That is, of all possible $\epsilon = 0.1, 0.2, \ldots, 5$, we use the result that has the smallest error. We plot the maximum error of each case over the possible ϵ in Figure 3.5. We also consider two types of error; relative error based off the infinity norm, and that of the two norm (both are plotted and labeled in the figures). Note that all of these are plotted on a log scale in order to estimate the order of accuracy. Since order of accuracy can be measured by the slope of a log scale error plot, we find the method to be approximately 2nd order. The data used to create these plots are seen in Table 3.1 and Table 3.2.

Table 3.2: Errors for varying stencil width with one-sided interior boundary

Stencil Size	Error $\|\|\|\cdot\|\|\|_\infty$	Error $\|\|\|\cdot\|\|\|_2$
5	4.72E-002	1.99E-002
10	3.61E-004	1.75E-004
15	1.3E-004	5.82E-005
20	4.99E-005	1.30E-005
30	1.53E-005	4.74E-006



(a) (b)

Figure 3.4: Convergence of the RBF-FD solution with one-sided stencils: (a) Relative two-norm errors as a function of increasing stencil sizes with fixed node set size of 2000 points. (b) Relative two-norm errors as a function of increasing node sets with fixed stencil size of 15. Note that the axes from both plots are on a logarithmic scale.

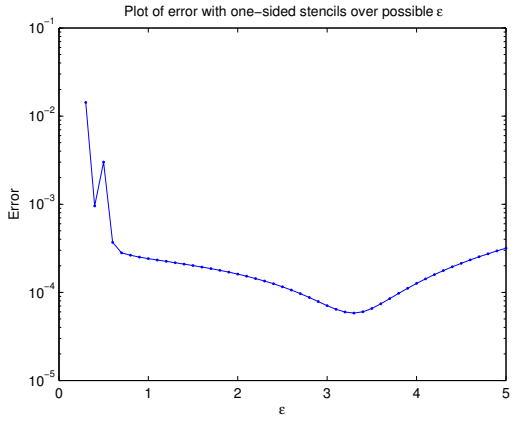


Figure 3.5: Relative two-norm of the error over possible ϵ.

Table 3.3: Errors for Node Set Density with Hermite Interior Boundary

Node Density	Error $\|\|\|_\infty$	Optimal ϵ	Error $\|\|\|_2$	Optimal ϵ
500	0.0013	3.3	6.77E-004	3.2
1000	5.98E-004	3.2	2.23E-004	3.2
2000	1.25E-004	3.3	6.13E-005	3.3
4000	5.08E-005	3	1.71E-005	3.2
8000	1.45E-005	3.2	4.21E-006	3.2

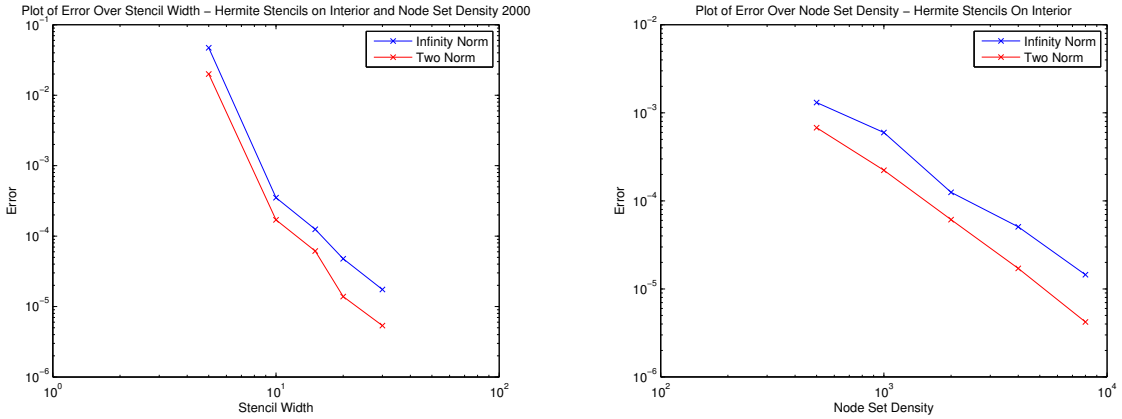
3.1.3 RBF-FD with Hermite Interpolation on Dirichlet Boundary

Lastly, we will use the other technique described in Section 2.2.4, which uses Hermite-Birkhoff interpolation to fictitious points. With the same ϵ and stencil size as in Section 3.1.2, we plot the spread of error (seen in Figure 3.3(b)). Notice that these results in comparison to Figure 3.3(a) are indistinguishable. However, by subtracting the two approximations, we saw that they differ by very little. This is expected since the techniques for finding solutions are the same everywhere but the interior boundary.

Further experimentation was done with the Hermite interpolation technique to investigate convergence with respect to stencil size. Figure 3.6(a) was the resulting convergence plot. As in the previous example, this was done to show the effectiveness of increasing the stencil size. Lastly, convergence plots were done with respect to node set density and we can see that this is approximately a second order method (seen in Figure 3.6(b)). The values that created these plots can also be found in Table 3.3. If you compare the data seen in Table 3.1 with 3.3, you see the Hermite technique did improve the accuracy of the solution, but by very little. As in the last case, we found these values by solving over a set of possible ϵ. We also show a plot of error over possible ϵ in Figure 3.7.

Table 3.4: Errors for Varying Stencil Width With Hermite Interior Boundary

Stencil Size	Error $\|\|\|\cdot\|\|\|_\infty$	Error $\|\|\|\cdot\|\|\|_2$
5	4.71E-002	1.99E-002
10	3.51E-004	1.70E-004
15	1.25E-004	6.13E-005
20	4.76E-005	1.38E-005
30	1.74E-005	5.35E-006



(a) (b)

Figure 3.6: Convergence of the RBF-FD solution with Hermite stencils: (a) Relative two-norm errors as a function of increasing stencil sizes with fixed node set size of 2000 points. (b) Relative two-norm errors as a function of increasing node sets with fixed stencil size of 15. Note the axes from both plots are on a logarithmic scale.

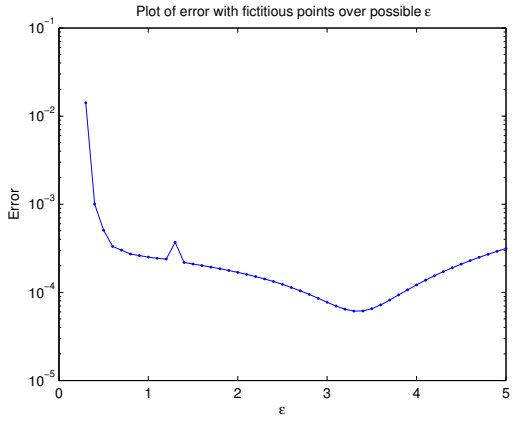


Figure 3.7: Relative two-norm of the error over possible ϵ.

3.2 Elliptical Domain with a Hole

Here we will solve a problem where the geometry is more complex. In this complex domain, an unreasonable mesh refinement would be required near the boundary, so standard finite differences cannot be applied. We will solve Poisson's equation on an ellipse punctured by a smaller ellipse with Neumann boundary conditions on the exterior and Dirichlet conditions on the interior ellipse. This domain can be described as $\Omega = \{(x,y)|\frac{x^2}{4} + y^2 = 0\} - \{(x,y)|16x^2 + 4y^2\}$. We use the following function to generate the right-hand side of Poisson's equation along with the Neumann boundary condition.

$$u(x,y) = \frac{1}{5}\left[(x-1)^2 y(16(x+1)^2 + 4y^2 - 1)\left(\frac{x^2}{4} + y^2 - 1\right)\right] + 1$$

The Dirichlet condition is given by evaluating the function on the interior boundary, which gives $f(\mathbf{x}_{interior}) = 1$.

This will be solved using the domain seen in Figure 3.8 with fictitious points used to enforce the exterior boundary condition. This domain was created with boundaries $\frac{x^2}{4} + y^2 = 0$ (outer ellipse) and $16(x+1)^2 + 4y^2 = 0$ (inner ellipse). We then run the RBF-FD fictitious points solution method and attain the errors in the domain (Figure 3.9(a)). For this experiment, we use $\epsilon = 1.8$ and $\epsilon = 1.5$, the stencil size was 15. We see that this method can handle irregular domains with a reasonable amount of accuracy.

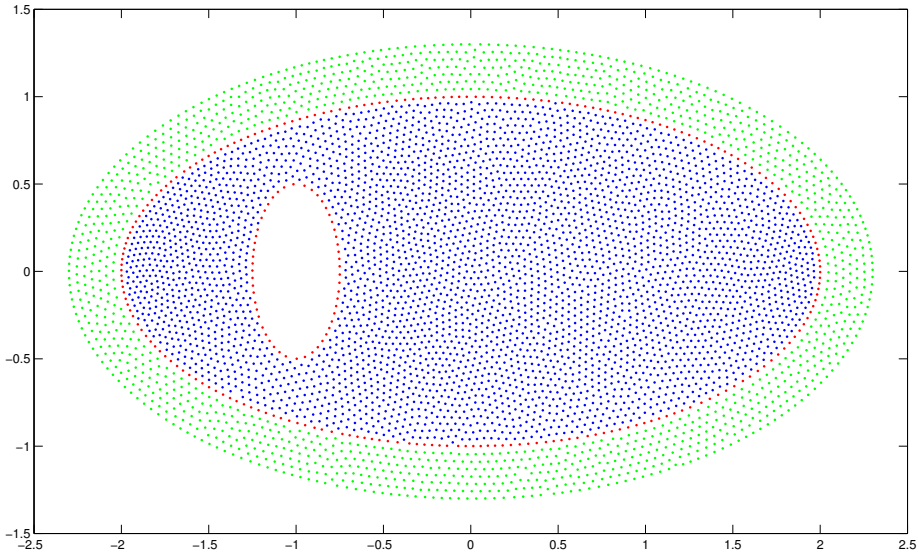
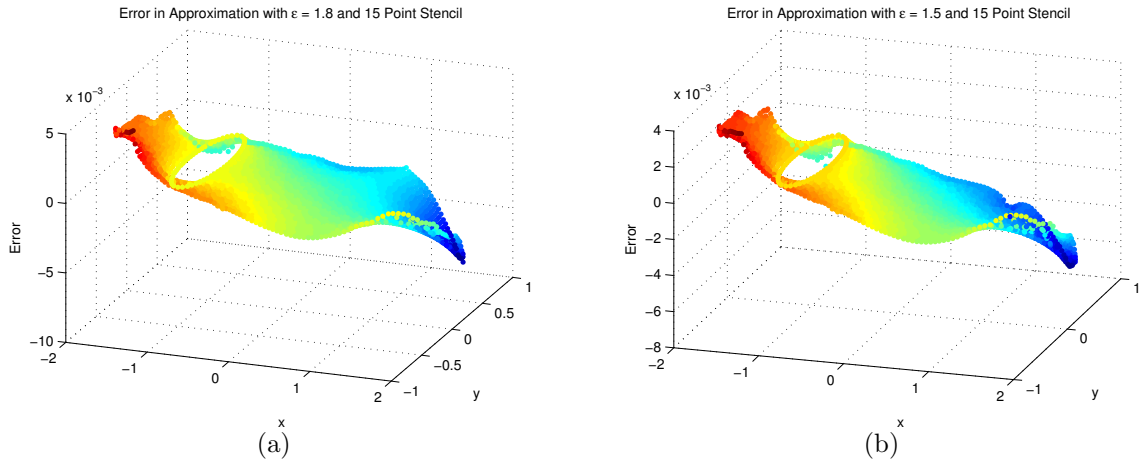


Figure 3.8: Ellipse Domain: 3700 interior points, 270 boundary points, 354 fictitious points



(a) (b)

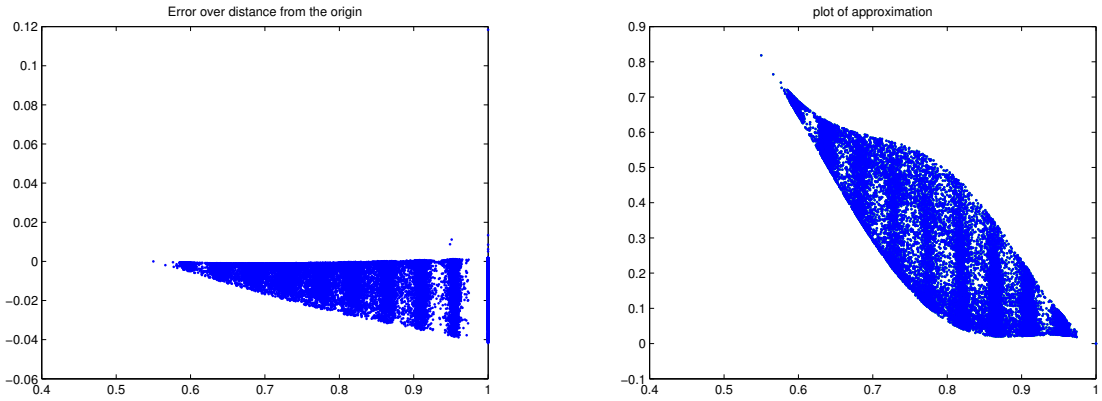
Figure 3.9: Errors for the elliptical domain. Left: Error with $\epsilon = 1.8$; Right: Error with $\epsilon = 1.5$

3.3 Spherical Shell

Here we test the method on a 3-dimensional problem. We solve Poisson's equation on a spherical shell with interior radius 0.55 and exterior radius 1 (which gives an aspect ratio similar to that of the Earth's mantle). We will create the problem based on the solution

$$u(x,y) = \frac{1-r}{r} + \frac{1}{2}\left(\frac{15}{16}\sqrt{\frac{5}{7\pi}}(x^4 - 6x^2y^2 + y^4) + \frac{3(3-30z^2+35z^4)}{16\sqrt{\pi}}\right)\sin^2\left(\frac{20}{9}\pi\left(\frac{-11}{20}+r\right)\right)$$

with $r = \sqrt{(x^2+y^2+z^2)}$. This was constructed such that $\mathbf{n}\cdot\nabla u = 0$ on the exterior boundary and the interior Dirichlet condition $f(\mathbf{x}_i) = 1$. We find a solution using the RBF-FD method and compare it to the analytic solution. Again, we experimented with different values of ϵ and stencil sizes. We chose to use a stencil size of 35 and $\epsilon = 0.9$. Since it is difficult to visualize the error in 3D, we plot the error in 2D with the x-axis as the radial distance from the origin and the y-axis as the error at the given nodes. Note the large gap between the closest interior points and boundary points. This limits the amount of edge points in the domain and therefore the extent to which the derivative-based boundary conditions can be enforced. The method was still robust enough to yield a reasonable solution. We use the same 2D technique to plot the true solution in Figure 3.10(b) to show the solution is not radially symmetric.



(a) (b)

Figure 3.10: On the right we see the spread of error in the approximation, and, on the left, the true solutions values.

CHAPTER 4

CONCLUDING REMARKS

In this thesis, we have introduced the first fictitious point method for solving Poisson's equation on irregular domains with Neumann boundary conditions using "scattered nodes" and the RBF-FD method. We summarize the main results of our findings as follows:

- The method uses fictitious points to implement Neumann boundary conditions, which limits the use of one-sided stencils.
- We tested the method on irregular domains in 2 and 3 dimensions:
  - The method exhibited smaller errors than the standard FD method.
  - The approximate solution converges to the true solution as the node density increases with a fixed stencil size.
  - For a fixed node set, the accuracy of the method can be improved by increasing the stencil size.
- The method should be extendable to other types of boundary conditions, such as Robin.

In a future study, we will explore the applications of the RBF-FD fictitious point method to time-dependent problems of both parabolic and hyperbolic types.

APPENDIX A

NODE PLACEMENT AND NODE SET GENERATING

Before we can use these methods, we first have to generate a set of distinct scattered nodes. We wish for the data to be unstructured and seemingly random. Here we develop an algorithm for generating such a data set. The idea behind this node set generator comes from Coulomb's law of repulsion in spring mass systems. Suppose we have a discrete domain Ω, such as a grid. We first need to break Ω into a series of overlapping sub domains $\{\Omega_1, \Omega_2, \ldots, \Omega_n\}$, with each node being the center of a sub domain. For instance, let x_1 be the center of sub domain Ω_1 and let Ω_1 be composed of $\mathbf{x}_1$ and the k closest nodes to $\mathbf{x}_1$. Thus, $\Omega_1 = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$. This will represent the set of all nodes that has an effective repulsion force acting on $\mathbf{x}_1$.

We then sum the forces acting on $\mathbf{x}_1$ (assume all forces have magnitude 1) and we will use this to find a direction of the overall force by normalizing it. In 2D, this will look like

$$F = \sum_{i=2}^{k} [(x_1, y_1) - (x_i, y_i)]$$

F is a vector composed of the forces acting on $\mathbf{x}_1$; however, we wish to normalize, so let

$$F_N = \frac{F}{\|F\|^p}$$

Here we introduced the p-term in Coulomb's law of repulsion [11]. Now, we repeat this process for each point in Ω, thus finding the direction of force for each point.

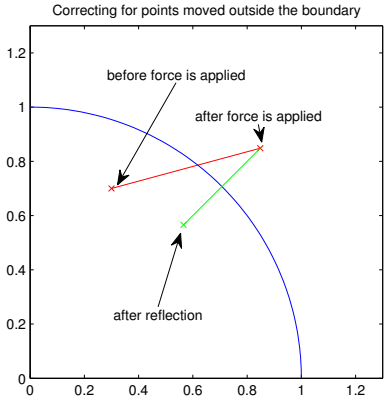


Figure A.1: Reflecting Point Moved Outside the Boundary

Next, we must understand that the magnitude of these forces may be too large for the size of an arbitrary domain. This is why we normalized the forces and only use them for direction. However, we are looking to relocate the point in the direction of this force, and we are looking for an equilibrium of the spring mass system. So, we propose moving each node an equal magnitude (call it δ) in the direction F_N. We then decrement δ and repeat this process until some desired equilibrium is found (pseudo code will follow).

However, another problem will arise in this. Points near the boundary will not have forces from an outer node and thus they will be moved through the boundary. To correct for nodes moved outside the boundary, we propose reflecting them back in. First, find all nodes moved outside of the boundary of the domain. Then, reflect them back in the domain on a path normal to the boundary at a distance equal to the nodes current distance from the boundary (seen in Figure A). If the reflection moves the point through the domain and back outside the boundary, the initial δ of the algorithm is too large.

Pseudo Code:

```
While equilibrium tolerance is not met do
    For length of Omega do
        Find Forces acting on each point in Omega
        Normalize Forces
    end
    Move points in direction of normalized force by magnitude delta
    Find points moved outside of boundary
    Reflect these points back in Omega as instructed above
    decrement delta
end
```

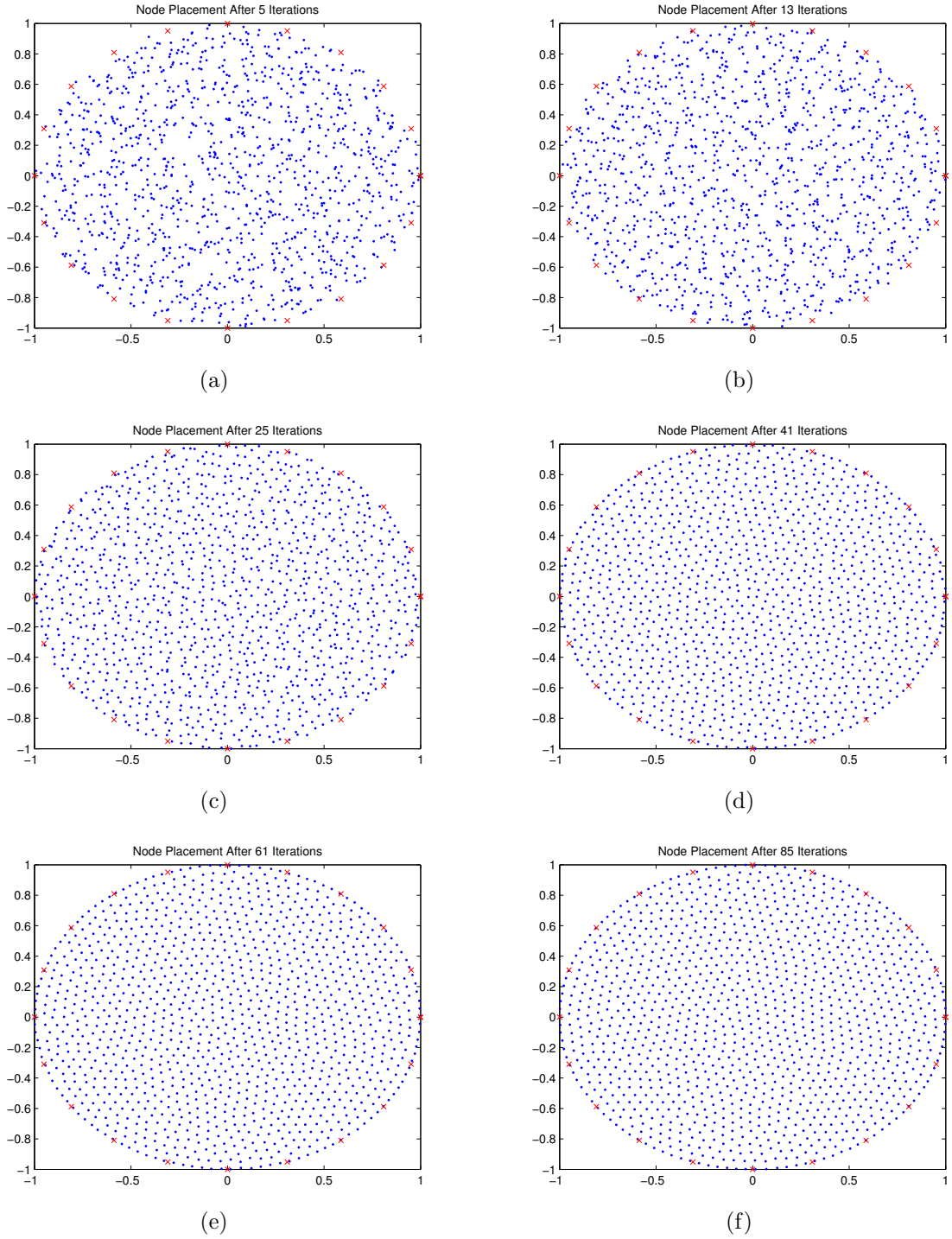
There are many options for experimentation with this method. For instance, how often do you decrement δ? What is the equilibrium tolerance? We found that it is best to do a max number of iterations for equilibrium tolerance and to decrement δ on a scale that is linear to the current iteration number. Essentially, this means we add another loop; our pseudo code then becomes

```
for j = 1:max
    for i = 1:4*j
        for length of Omega
            Find Forces acting on each point in Omega
            Normalize Forces
        end
        Move points in direction of normalized force by magnitude delta
        Find points moved outside of boundary
        Reflect these points back in Omega as instructed above
```

```
        Decrement delta
    end
end
```

Figure A.2 shows how this algorithm behaves through several iterations. Notice the similarity from A.3(e) to A.3(f). This is due to the fact that the spring mass system is reaching equilibrium.

Figure A.2: Several Iterations of the Node Generating Algorithm



REFERENCES

[1] T. Cecil, J. Qian, and S. Osher. Numerical methods for high dimensional hamilton-jacobi equations using radial basis functions. J. Comput. Phys., 196:327–347, 2004.

[2] J. Duchon. Splines mimimizing rotation-invariant semi-norms in sobolev space. Constructive Theory of Functions of Several Variables, Springer Lecture Notes in Math, 21:85–100, 1977.

[3] G. E. Fasshauer. Solving partial differential equations with radial basis functions: multilevel methods and smoothing. Adv. Comput. Math., pages 139–159, 1999.

[4] G. E. Fasshauer and J. G. Zhang. On choosing "optimal" shape parameters for rbf approximation. Numerical Algorithms, To appear (2007).

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

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

[7] B. Fornberg and E. Lehto. Stabilization of rbf-generated finite difference methods for convective pdes. Jornal of Comput. Phys., 230 Issue 6:2271–2285, 2011.

[8] B. Fornberg and C. Piret. On choosing a radial basis function and a shape parameter when solving a convective PDE on a sphere. J. Comp. Phys., 227:2758–2780, 2008.

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

[10] B. Fornberg and J. Zuev. The Runge phenomenon and spatially variable shape parameters in RBF interpolation. Comput. Math. Appl., 54:379–398, 2007.

[11] D. J. Griffiths. Introduction to Electrodynamics (3rd ed.). Prentice Hall, 1998.

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

[13] Y. C. Hon and X. Z. Mao. An efficient numerical scheme for Burgers' equation. Appl. Math. Comput., 95:37–50, 1998.

[14] 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.

[15] E. J. Kansa and Y. C. Hon. Circumventing the ill-conditioning problem with multiquadric radial basis functions: Applications to elliptic partial differential equations. Comput. Math. Appl., 2001.

[16] E. Larsson and B. Fornberg. A numerical study of some radial basis function based solution methods for elliptic pdes. Comput. Math. Appl., 46:891–902, 2003.

[17] J. C. Mairhuber. On Haar's theorem concerning Chebychev approximation problems having unique solutions. Proc. Amer. Math. Soc., 7:609–615, 1956.

[18] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. Constr. Approx., 2:11–22, 1986.

[19] D. Y. N. Flyer, G.B. Wright. A hybrid radial basis function - pseudospectral method for thermal convection in a 3d spherical shell. Geochem. Geophys. Geosyst., 2010.

[20] F. J. Narcowich and J. D. Ward. Generalized Hermite interpolation via matrix-valued conditionally positive definite functions. Math. Comp., 63:661–687, 1994.

[21] S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. Adv. Comput. Math., 11:193–210, 1999.

[22] C. Shu, H. Ding, and K. S. Yeo. Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible navier-stokes equations. Comput. Methods Appl. Mech. Engrg., 192:941–954, 2003.

[23] A. I. Tolstykh, M. V. Lipavskii, and D. A. Shirobokov. High-accuracy discretization methods for solid mechanics. Arch. Mech., 55:531–553, 2003.

[24] J. Wertz, E. J. Kansa, and L. Ling. The role of multiquadric shape parameters in solving elliptic partial differential equations. Comput. Math. Appl., 51:1335–1348, 2006.

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

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

[27] Z. Wu. Hermite-birkoff interpolation of scattered data by radial basis functions. Approx. Theory Appl., 8(2):1–10, 1992.