2016

Approximating Nonlocal Diffusion Problems Using Quadrature Rules Generated by Radial Basis Functions

Isaac Ron Lyngaas

FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCE

APPROXIMATING NONLOCAL DIFFUSION PROBLEMS USING QUADRATURE RULES GENERATED BY RADIAL BASIS FUNCTIONS

By

ISAAC LYNGAAS

A Thesis submitted to the
Department of Scientific Computing
in partial fulfillment of the
requirements for the degree of
Master of Science

2016

Isaac Lyngaas defended this thesis on September 9, 2016.
The members of the supervisory committee were:

Janet Peterson

Professor Directing Thesis

Max Gunzburger

Committee Member

John Burkardt

Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the thesis has been approved in accordance with university requirements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Nonlocal models differ from traditional partial differential equation (PDE) models because they contain no spatial derivatives; instead an appropriate integral is used. Nonlocal models are especially useful in the case where there are issues calculating the spatial derivatives of a PDE model. In many applications (e.g., biological systems, flow through porous media) the observed rate of diffusion is not accurately modeled by the standard diffusion differential operator but rather exhibits so-called anomalous diffusion. Anomalous diffusion can be represented in a PDE model by using a fractional Laplacian operator in space whereas the nonlocal approach only needs to slightly modify its integral formulation to model anomalous diffusion. Anomalous diffusion is one such case where approximating the spatial derivative operator is a difficult problem.

In this work, an approach for approximating standard and anomalous nonlocal diffusion problems using a new technique that utilizes radial basis functions (RBFs) is introduced and numerically tested. The typical approach for approximating nonlocal diffusion problems is to use a Galerkin formulation. However, the Galerkin formulation for nonlocal diffusion problems can often be difficult to compute efficiently and accurately especially for problems in multiple dimensions. Thus, we investigate the alternate approach of using quadrature rules generated by RBFs to approximate the nonlocal diffusion problem.

This work will be split into three major parts. The first will introduce RBFs and give some examples of how they are used. This part will motivate our approach for using RBFs on the nonlocal diffusion problem. In the second part, we will derive RBF-generated quadrature rules in one dimension and show they can be used to approximate nonlocal diffusion problems. The final part will address how the RBF quadrature approach can be extended to higher dimensional problems. Numerical test cases are shown for both the standard and anomalous nonlocal diffusion problems and compared with standard finite element approximations. Preliminary results show that the method introduced is viable for approximating nonlocal diffusion problems and that highly accurate approximations are possible using this approach.

CHAPTER 1

INTRODUCTION

In recent years, nonlocal models have seen increased interest due to their ability to describe physical phenomena which are not modeled well by standard PDE models. Nonlocal models differ from standard PDE models in that they are free of spatial derivatives. Instead of a spatial derivative, nonlocal models choose to use an integral formulation that features the interactions which occur between spatial points separated by a finite distance. The applications that are of interest in this work are nonlocal diffusion and anomalous diffusion that correspond to the fractional Laplacian operator. Anomalous diffusion refers to the case where the spatial spread of the diffusing quantity is not proportional to the square root of time as predicted by the heat equation.

Nonlocal diffusion problems have been approximated using several approaches. For example, in [7, 8, 27] standard Finite Element methods are used to discretize the equation when the solution is continuous and discontinuous Galerkin methods are used for problems involving discontinuities. However, using Finite Element methods to approximate the nonlocal diffusion equation causes difficulties because the nonlocal diffusion equation needs to be integrated to find the weak formulation. This means that in one dimension a double integral needs to be approximated and in two dimensions a triple integral needs to be approximated, etc. Therefore, to formulate the Finite Element problem for a nonlocal diffusion problem more complicated quadrature rules must be used than when the Finite Element Method is used to approximate standard PDE models. Another example, in [3] uses a Galerkin-RBF method to approximate the nonlocal diffusion problems. An RBF is a real-valued function whose value only depends on the distance from a fixed point so it is readily extendable to higher dimensions. In this Galerkin-RBF approach, RBFs are used as the approximating functions instead of piecewise polynomials in the FEM approach. The problem with this approach is that the formulation does not alleviate the need to accurately approximate the multivariable integral for the nonlocal term.

RBFs have seen widespread use since their introduction by Hardy [17] in 1971 for the interpolation of scattered data sets. The advantages of using RBFs include the possibility of yielding

spectrally convergent results and the ease at which they are extended to higher dimensions. There are many different types of RBFs which are classified by the support of the RBF (compact or global) and the degree of differentiability of the RBF. Often these RBFs will include a shape parameter which is a free parameter whose utility is to help tune an RBF approximation. To use RBFs for interpolation, a particular RBF is chosen and then the interpolant is formed as a linear combination of translates of the RBF at each data point. As usual, the weights for the RBF interpolant are determined by enforcing the interpolation conditions at each point. Theory has been developed that shows RBF interpolation is an invertible problem when the RBF is positive definite.

In recent years, RBFs have been used to approximate the solution of PDEs using a collocation approach [18] or a Galerkin-RBF approach [25]. In the collocation approach, the differentiability of RBFs is taken advantage of to form an approximation in a similar manner to the RBF interpolation problem. In the Galerkin-RBF approach, RBFs are used to form a basis in place of more commonly used bases. In addition to being used for PDEs, RBFs have also been used to approximate integral equations using a similar collocation approach that was used for the PDE approach [2] and nonlocal problems using a Galerkin-RBF approach as mentioned before.

Another approach that has been taken is to use RBFs in order to derive finite difference stencils on a scattered grid that approximate a derivative operator. In this approach, finite difference stencils are derived by choosing the weights so that the rule is exact for translates of a given RBF instead of the usual approach of making the rule exact for monomials. These so-called RBF-generated finite difference stencils have been implemented successfully for various applications including geophysical problems on the sphere [13], the incompressible Navier-Stokes equations [20], and the Hamilton-Jacobi equations [6].

In this work, we first consider approximating the solution of the one-dimensional nonlocal diffusion equation

$$\frac{du}{dt} + \frac{2(1-s)}{\delta^{2-2s}} \int_{x-\delta}^{x+\delta} \frac{u(x) - u(z)}{|x-z|^{1+2s}} dz.$$

Here $s \in [0, 1)$ governs the diffusion rate and $\delta > 0$ defines the horizon length for the nonlocal integral. The horizon determines the finite distance with which spatial points interact in the nonlocal integral. Our strategy is to use RBFs to derive a quadrature formula for the nonlocal integral term and then use standard BDFs to approximate the time derivative. This approach

alleviates the need to approximate a double integral in one dimension which arises when a Galerkin formulation is used. The quadrature points for the RBF-generated quadrature rule are chosen to be grid points (either uniformly spaced or scattered) in the interval $[x-\delta, x+\delta]$. To determine the quadrature weights, we enforce the rule to be exact for translates of a given RBF. It is important to note that we are deriving a quadrature rule for the specific integrand of the nonlocal integral and not for a generic integrand as one typically does in numerical quadrature. In addition, we will also discuss the difficulties which arise when attempting to accurately obtain quadrature weights due to the possible ill-conditioning of the matrix that results from this method and address how this can be overcome by altering the shape parameter of the RBF. After describing the derivation of the RBF-generated quadrature rules, we apply these rules to approximate the nonlocal integral and obtain relative errors and numerical rates of convergence. Once the quadrature rules for the nonlocal integral have been found, the complete nonlocal diffusion problem in one dimension is approximated and the results are compared with Finite Element results from [27] for various rates of diffusion. The RBF-generated quadrature rule approach greatly improves on the Finite Element results. The tradeoff is that the bandwidth of the RBF approach increases as more quadrature points are used whereas the Finite Element matrix has a fixed bandwidth. The size of the matrices are identical when the same grid is used, however the computational cost of approximating the entries in the Finite Element matrix and right-hand side are much higher than the RBF approach.

After describing how the RBF-generated quadrature rule is derived in one dimension, we turn to higher dimensional problems. In particular, we consider the two-dimensional case but generalizations to higher dimensions are obvious. In this situation, we have the option of deriving a quadrature in a manner analogous to the one-dimensional case or using tensor product of one-dimensional rules. We demonstrate results for both approaches. Moreover, for the non-tensor product approach we consider both a square and a circular horizon where the quadrature points reside. We discuss the pros and cons of each approach for approximating the integral and then apply these rules to solve a nonlocal diffusion problem in two dimensions.

In Chapter 2, a brief overview of RBFs is given and several methods for solving PDEs using RBFs are described. In Chapter 3, a method for deriving RBF-generated quadrature rules for the nonlocal integral in one dimension is described and tested. In addition, an approach for using these quadrature rules to approximate nonlocal diffusion problems is described and tested for various

diffusion rates. In Chapter 4, two methods for finding RBF-generated quadrature rules for the nonlocal integral in two dimensions are described and tested using several different approaches for choosing quadrature points. These quadrature rules are then used to approximate a two-dimensional nonlocal diffusion problem. Finally in Chapter 5, we summarize some conclusions about our approach for approximating the nonlocal diffusion problem and discuss future work.

CHAPTER 2

A BRIEF OVERVIEW OF RBFS & THEIR APPLICATIONS

Radial basis functions (RBFs) are a class of functions that depend only on the distance between two points which makes them easily extendable to higher dimensions. The origins of RBFs can be tracked back to Hardy [17] who initially developed RBFs in order to interpolate geographical surfaces given some scattered data sites. In his work, Hardy was attempting to find a method for approximating two dimensional surfaces since traditional methods like polynomial and Fourier series interpolation did not provide adequate accuracy. Hardy found that choosing a particular RBF and forming an approximation using a linear combination of translates of the RBF at given data sites did a particularly good job at interpolating multivariable functions.

Interpolation using RBFs is normally done using a global collocation approach where an approximation is formed using all given data sites as values that should be interpolated. In a survey of methods for scattered data approximation of multivariable functions, Franke [16] found that this global collocation approach using RBFs is one of the most accurate methods available for function interpolation in multiple dimensions. This finding of Franke ultimately led to interest in using RBFs in order to approximate PDEs.

RBFs have been used in various ways to approximate solutions to PDEs. In [18], a method known as Kansa's method that uses a global collocation approach with RBFs is derived in order to approximate PDEs. In [25], a Galerkin approach using RBFs is taken in order to approximate PDEs. The main advantage of using these global RBF methods for solving PDEs and interpolating functions is their possible spectral accuracy, however the drawback is the large, dense, possibly ill-conditioned linear system that results from these methods. Another approach that has recently been taken by several authors [6, 20, 28] is known as the RBF-generated finite difference (RBF-FD) stencil method. This method is considered a local method as it uses only nearby data sites to devise a finite difference approximation to a differential operator using RBFs. Using RBFs allows for a finite difference stencil to be found on non-uniform grids that maintain the same accuracy

as standard finite difference approximations on uniform grids. This method sacrifices the possible spectral accuracy from global methods for algebraic convergence in order to result in a sparse better-conditioned system.

In addition to solving PDEs with RBFs, interest has been shown for using RBFs to approximate integral equations. In [2], a global collocation approach with RBFs is used to succesfully approximate integral equations where the unknown is sought as a linear combination of translations of RBFs and the integral is approximated by standard numerical quadrature. RBFs have also been successfully used in a Galerkin approach to solve a nonlocal diffusion problem; see [3]. However, this approach is analogous to the Galerkin-RBF approach used in [25] to solve PDEs and it is not the approach we take in this work. Thus, RBF methods have been shown to not only work on differential equations but integral equations as well.

Methods using RBFs are often called *mesh-free* methods as compared with mesh-based methods such as finite difference and finite element methods. Rather than discretizing space, in *mesh-free* methods solutions are approximated by a set of basis functions which typically have much more smoothness than standard finite element basis functions. An advantage of these *mesh-free* methods is that no complicated meshes need to be formulated for them to be used.

In §2.1, common RBFs are defined and some of their important properties are presented. In §2.2, the global collocation approach for function interpolation is derived. Global approaches for approximating PDEs are outlined in §2.3. Finally, §2.4 will detail the RBF-FD approach. An overview of these methods for function interpolation and PDE approximation will be useful for formulating a method that uses RBFs to approximate the integral in the nonlocal problem of interest.

2.1 RBFs

RBFs consist of radial functions of the following form

$$\phi(r(\mathbf{x}-\mathbf{x}_c),\varepsilon),\ \mathbf{x}\in\mathbb{R}^d \tag{2.1}$$

where $\mathbf{x}_c$ is some fixed point in $\mathbb{R}^d$ which is often called a center,

$$r(\mathbf{x}) = \|\mathbf{x}-\mathbf{x}_c\|,\ \mathbf{x}\in\mathbb{R}^d$$

is some measure of distance, usually the standard Euclidean distance which we use in this work, and ε is a free parameter often called the shape parameter. These functions are known as radial functions as they are radially symmetric about the center $\mathbf{x}_c$ and depend on the distance from $\mathbf{x}$ to the given center. The utility of the shape parameter ε is discussed in §2.2.1.

Table 2.1: Commonly used RBFs.

Type of Basis Function	Radial Function $\phi(r, \varepsilon)$
Gaussian	$e^{-(\varepsilon r)^2}$
Multiquadric	$\sqrt{1+(\varepsilon r)^2}$
Inverse Multiquadric	$1/\sqrt{1+(\varepsilon r)^2}$
Inverse Quadratic	$1/(1+(\varepsilon r)^2)$

Several different choices of radial functions are available for use in RBF methods; some common choices are shown in Table 2.1. The radial functions listed in Table 2.1 have two major properties in common: infinite differentiability and global support. Radial functions with these properties are considered throughout this work as they can attain spectral convergence rates in all RBF methods. Spectral convergence in this case means that the size of the approximation error decreases at an exponential rate dependent on a parameter α being decreased (i.e., of the general form $\mathcal{O}(e^{-\alpha})$). Radial functions that are not infinitely differentiable (such as the radial cubic function $\phi(r) = r^3$) are not considered here as they result in algebraic convergence due to jumps in the derivative. There are compactly supported radial functions (i.e., ones that are zero after a certain distance from the center) that can be used in RBF methods, however they have inferior convergence rates to globally supported radial functions [24]. Compactly supported radial functions are mainly used in RBF methods to avoid the dense matrices that result from global RBF methods that are computationally expensive to solve [12]. See [23, 5] for examples of compactly supported radial functions that are suitable for RBF methods. An infinitely differentiable, globally supported radial function from Table 2.1 is considered throughout the remainder of this work because we are interested in finding approximations with higher possible accuracy and convergence rates for our application.

2.2 RBF Interpolation

In order to interpolate a function using RBFs, a finite set of scattered data points $\{f_j\}_{j=1}^N$ at distinct spatial locations $\{\mathbf{x}_j\}_{j=1}^N, \mathbf{x}_j \in \Omega \subset \mathbb{R}^d$ is considered. Choosing an RBF from Table 2.1

and taking translates $\phi(\|\mathbf{x}-\mathbf{x}_j\|,\varepsilon)$ at each location $\mathbf{x}_j$, an interpolant of the form

$$I^N(x) = \sum_{j=1}^{N} c_j \phi(\|\mathbf{x}-\mathbf{x}_j\|,\varepsilon) \tag{2.2}$$

is sought where $\{c_j\}_{j=1}^N$ are expansion coefficients. These expansion coefficients are found by enforcing the interpolation conditions $I^N(\mathbf{x}_i) = f_i,\ i = 1,\ldots,N$ which results in a symmetric linear system of the form

$$\begin{pmatrix} \phi(\|\mathbf{x}_1-\mathbf{x}_1\|,\varepsilon) & \phi(\|\mathbf{x}_1-\mathbf{x}_2\|,\varepsilon) & \cdots & \phi(\|\mathbf{x}_1-\mathbf{x}_N\|,\varepsilon) \\ \phi(\|\mathbf{x}_2-\mathbf{x}_1\|,\varepsilon) & \phi(\|\mathbf{x}_2-\mathbf{x}_2\|,\varepsilon) & \cdots & \phi(\|\mathbf{x}_2-\mathbf{x}_N\|,\varepsilon) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_N-\mathbf{x}_1\|,\varepsilon) & \phi(\|\mathbf{x}_N-\mathbf{x}_2\|,\varepsilon) & \cdots & \phi(\|\mathbf{x}_N-\mathbf{x}_N\|,\varepsilon) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix} \tag{2.3}$$

or in an alternate form

$$\mathbf{A}\mathbf{c} = \mathbf{f} \tag{2.4}$$

where $\mathbf{A} = (A(\mathbf{x}_i,\mathbf{x}_j)_{i,j=1}^N)$ with $A(\mathbf{x}_i,\mathbf{x}_j) = \phi(\|\mathbf{x}_i-\mathbf{x}_j\|,\varepsilon)$, $\mathbf{c} = (c_1,\ldots,c_N)^T$, and $\mathbf{f} = (f_1,\ldots,f_N)^T$. The matrix $\mathbf{A}$ is often called the RBF interpolation matrix. Solving this system of equations gives the expansion coefficients $\{c_j\}_{j=1}^N$ for the interpolant $I^N(x)$. The interpolant is constructed in this manner as it guarantees that the RBF interpolation matrix $\mathbf{A}$ generated in (2.4) is nonsingular for choices of radial functions $\phi(r,\varepsilon)$ that are positive definite.

Theorem 1. *A real-valued continuous function ϕ is positive definite on $\mathbb{R}^d$ if and only if it is even and*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j \phi(\boldsymbol{x}_i - \boldsymbol{x}_j) \geq 0$$

for any N pairwise different points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N \in \mathbb{R}^d$ and $\boldsymbol{c} = (c_1,\ldots,c_N)^T \in \mathbb{R}^N$ [11].

Clearly all choices of radial functions $\phi(r,\varepsilon)$ from Table 2.1 are even functions as they are radially symmetric functions based on distance. If $\phi(r,\varepsilon)$ is a positive definite function, it can be proved that the interpolation matrix $\mathbf{A}$ is a symmetric positive definite matrix for any distinct points $\mathbf{x}_1,\ldots,\mathbf{x}_N$ making it nonsingular. It has been shown that Gaussian, inverse multiquadric,

and inverse quadratic radial functions are positive definite functions [19]. Other radial functions that are often used in RBF methods such as the multiquadric function on the other hand have only been show to be conditionally positive definite.

Definition 1. *A real-valued continuous even function ϕ is called conditionally positive definite of order m on $\mathbb{R}^d$ if*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j \phi(\boldsymbol{x}_i - \boldsymbol{x}_j) \geq 0$$

for any N points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^d$, and $\boldsymbol{c} = (c_1, \ldots, c_N)^T \in \mathbb{R}^N$ satisfying

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

for any real valued polynomial p of degree at most m-1 [11].

A conditionally positive definite function of order 0 is then considered a positive definite function. The extra polynomial constraint conditions for conditionally positive definite RBFs means that the interpolant should reproduce polynomials. When using radial basis functions that are conditionally positive definite, it is common practice to add in these necessary polynomial constraints from Definition 1 to the linear system from (2.3). For example using the multiqaudric function which is conditionally positive definite of order 1, the constraint $\sum_{j=1}^{N} c_j = 0$ needs to be added to the linear system. Thus, the system that now needs to be solved for the expansion coefficients is

$$\begin{pmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|, \varepsilon) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|, \varepsilon) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_N\|, \varepsilon) & 1 \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|, \varepsilon) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|, \varepsilon) & \cdots & \phi(\|\mathbf{x}_2 - \mathbf{x}_N\|, \varepsilon) & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 \\ \phi(\|\mathbf{x}_N - \mathbf{x}_1\|, \varepsilon) & \phi(\|\mathbf{x}_N - \mathbf{x}_2\|, \varepsilon) & \cdots & \phi(\|\mathbf{x}_N - \mathbf{x}_N\|, \varepsilon) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \\ c_{N+1} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \\ 0 \end{pmatrix}.$$

Now ignoring c_{N+1}, the expansion coefficients $\{c_j\}_{j=1}^{N}$ are used to form an interpolant like (2.2).

In order to improve accuracy of the RBF interpolant there are two strategies that can be taken. The first strategy is to add more data points to the interpolant at other distinct spatial points in

Ω. In [26], it is shown that error bounds of RBF interpolants using globally supported, positive definite, radial functions have spectral convergence dependent on the fill distance of the space Ω

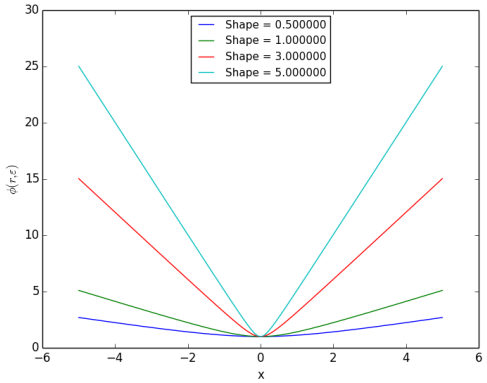
$$h_N = \sup_{\mathbf{x}\in\Omega} \min_{j=1,\ldots,N} \|\mathbf{x} - \mathbf{x}_j\|$$

or the largest "data site free" ball in Ω. It is not necessarily true that adding more points will decrease h_N, however it is simple to devise strategies for picking data points that decrease the fill distance. The second strategy for improving accuracy is to lower the shape parameter ε. Driscoll and Fornberg [9] have observed that in the case where $\varepsilon \to 0$ RBF interpolation with globally supported RBFs reproduces pseudospectral methods (i.e., Fourier methods and Chebyshev methods). This means that accuracy of the interpolation should improve as ε is lowered towards zero. The problem with both of these strategies for increasing accuracy is that the interpolation matrix becomes increasingly ill-conditioned which can lead to numerical instabilities.

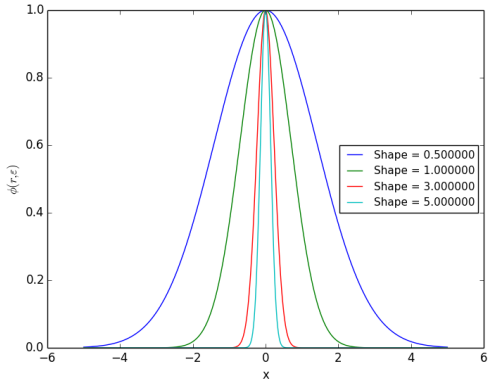
2.2.1 Ill-Conditioning and the Shape Parameter

To get a better idea of why this conditioning problem arises using RBFs, a more thorough look at the interpolation matrix with regards to the shape parameter and the spacing of points is needed. Figure 2.1 shows in one dimension how the radial functions from Table 2.1 change as the shape parameter varies. These functions tend to flatten out as $\varepsilon \to 0$ which means that as more accurate RBF interpolants are sought the entries in the interpolation matrix in (2.3) tend toward values that are closer and closer together. As ε is decreased, numerical conditioning problems occur when the point is reached where these numbers cannot be distinguished from one another if they cannot be represented on a computer with more precision. Now, consider what happens as the accuracy is increased by decreasing the fill distance h_N. By decreasing h_N, the spatial points in the RBF interpolant become closer and closer together. Once two points become too close together, a problem occurs where rows in the interpolation matrix (2.3) become almost identical yielding an ill-conditioned system.

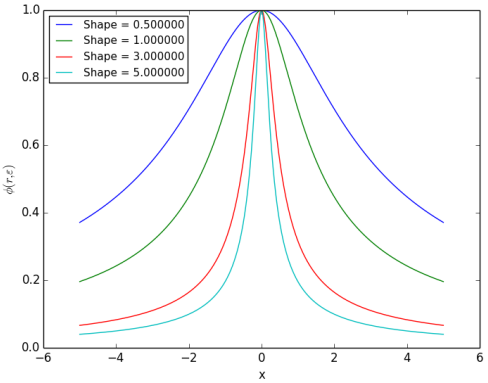
Both of those conditioning issues are a hinderance to how far the accuracy of RBF interpolants can be pushed. A simple way to combat these conditioning issues is to use higher precision arithmetic, however using higher precision arithmetic can become very computationally expensive and may not always be available. Also, there are several algorithms such as the Contour-Pade algorithm



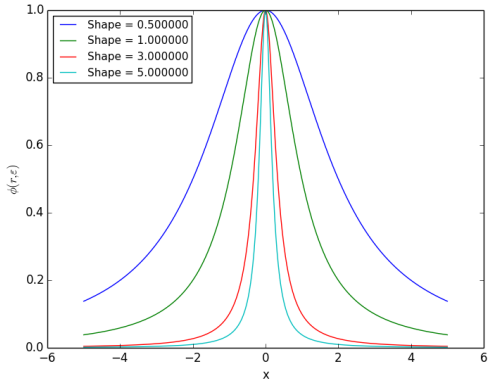
(a) Multiquadric



(b) Gaussian



(c) Inverse Multiquadric



(d) Inverse Quadratic

Figure 2.1: RBFs from Table 2.1 with a varying shape parameter.

[15] and the RBF-QR algorithm [14] that can bypass the numerical conditioning problem as $\varepsilon \to 0$, however these algorithms will not be discussed as they are not in the scope of this work.

2.2.2 Numerical Examples

We now want to numerically test the RBF interpolation method to show how approximation error changes as more points are added and the shape parameter changes. For a numerical test in one dimension, we are interested in approximating a function $f(x)$ on a domain $\Omega = [a, b]$. In order to test the RBF interpolation, we need to come up with a scheme for choosing points for the method. The scheme we use chooses $N + 1$ points

$$x_1 = a, x_2 = a + h, \ldots, x_N = a + (N - 1)h, x_{N+1} = b \tag{2.5}$$

with uniform spacing $h = \frac{b-a}{N}$ over the domain Ω. Choosing points in this manner guarantees that the fill distance of the points on the domain decreases as N grows.

The one-dimensional test problem we consider uses the test function $f(x) = \sin(\pi x)$ on the domain $\Omega = [0, 1]$. In Table 2.2, results are shown for the method as the number of uniform points for the RBF interpolation method increases using Gaussian RBFs with a shape parameter $\varepsilon = 5.0$. From these results, we see spectral-like convergence rates as the number of uniform points used in the method increase. However, we also see that as the number of points increase the condition number of the system that needs to be solved for the method increases as well. The large condition number of the system becomes an issue when $N = 64$ resulting in a fall off in the convergence rates of the error. The error does not decrease because we only use double precision arithmetic for these calculations. In Table 2.3, similar test results to Table 2.2 are shown except that the shape parameter $\varepsilon = 2.0$ is now used. We see that the $\varepsilon = 2.0$ case also has spectral-like convergence rates. Lower errors are observed in the $\varepsilon = 2.0$ cases for all values of N, however the condition number of the system becomes too large faster causing convergence rates to drop off sooner.

2.3 Standard Approaches for Using RBFs to Approximate PDEs

In this section some common approaches for numerically solving PDEs using RBFs are discussed. The first method uses a collocation approach similar to the one used for RBF interpolation in §2.2. The second method uses a variational approach using RBFs for approximating PDEs.

Table 2.2: RBF interpolation of the function $f(x) = \sin(\pi x)$ on $\Omega = [0, 1]$ as the number of uniform points N increases using Gaussian RBFs with a shape parameter of $\varepsilon = 5.0$.

N	L2 Error	Conv. Rate	Cond. #
4	9.80898e-03	-	2.1
8	3.74012e-03	1.39	1.3e+02
16	5.31249e-06	9.46	1.8e+08
32	1.03863e-09	12.32	5.6e+17
64	1.3398e-09	-0.37	3.8e+19

Table 2.3: RBF interpolation of the function $f(x) = \sin(\pi x)$ on $\Omega = [0, 1]$ as the number of uniform points N increases using Gaussian RBFs with a shape parameter of $\varepsilon = 2.0$.

N	L2 Error	Conv. Rate	Cond. #
4	0.00268994	-	2.2e+02
8	6.72786e-06	8.6	3.7e+07
16	4.65698e-10	13.82	1.8e+18
32	1.68855e-10	1.46361	2.8e+18

2.3.1 Kansa's Method

In [18], Kansa introduces a non-symmetric method for numerically solving PDEs using RBFs which is now known as Kansa's Method. Kansa's Method uses techniques very similar to the global collocation approach, however its goal is to use RBFs in order to solve PDEs. In order to demonstrate Kansa's Method, the simple time independent PDE in (2.6) is considered where Ω is some bounded domain of $\mathbb{R}^d$, $\partial\Omega$ is the boundary of Ω, L is a linear partial differentiation operator, f is the forcing function for the PDE, and g is the given boundary data.

$$\begin{cases} Lu(\mathbf{x}) = f(\mathbf{x}), \ \mathbf{x} \in \Omega \\ \\ u(\mathbf{x}) = g(\mathbf{x}), \ \mathbf{x} \in \partial\Omega \end{cases} \tag{2.6}$$

The process begins by choosing a set of N distinct spatial locations $X = \{\mathbf{x}_i\}_{i=1}^N \in \bar{\Omega}$ where $\bar{\Omega} = \Omega \cup \partial\Omega$. Similar to RBF interpolation, an RBF from Table 2.1 is chosen and translates of that RBF $\phi(\|\mathbf{x} - \boldsymbol{\xi}_j\|, \varepsilon)$ are used to form an approximation $\hat{u}$ to u in the form

$$\hat{u}(x) = \sum_{j=1}^{N} c_j \phi(\|\mathbf{x} - \boldsymbol{\xi}_j\|, \varepsilon) \tag{2.7}$$

where typically we set $\boldsymbol{\xi}_j = \mathbf{x}_j,\ j = 1, \ldots, N$. In this case, we will consider only positive definite RBFs in order to simplify notation so that the extra polynomial conditions from conditionally positive definite RBFs are not needed. In order to find the expansion coefficients $\{c_j\}_{j=1}^N$ for (2.7) the collocation conditions

$$L\hat{u}(x_i) = f(\mathbf{x}_i),\ \mathbf{x}_i \in \Omega$$
$$\hat{u}(x_i) = g(\mathbf{x}_i),\ \mathbf{x}_i \in \partial\Omega$$

are used. These collocation conditions are used to form the following linear system

$$\begin{bmatrix} L\boldsymbol{\phi} \\ \boldsymbol{\phi} \end{bmatrix} \begin{bmatrix} \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \tag{2.8}$$

where $L\boldsymbol{\phi}$ and $\boldsymbol{\phi}$ are blocks that are generated as follows

$$L\boldsymbol{\phi}_{ij} = L\phi(\|\mathbf{x}_i - \xi_j\|, \varepsilon),\ \mathbf{x}_i \in \Omega,\ \xi_j \in X$$
$$\boldsymbol{\phi}_{ij} = \phi(\|\mathbf{x}_i - \xi_j\|, \varepsilon),\ \mathbf{x}_i \in \partial\Omega,\ \xi_j \in X,$$

where $\mathbf{f}$ is a vector such that $\mathbf{f}_i = f(\mathbf{x}_i)$ for $\mathbf{x}_i \in \Omega$, $\mathbf{g}$ is a vector such that $\mathbf{g}_i = g(\mathbf{x}_i)$ for $\mathbf{x}_i \in \partial\Omega$, and $\mathbf{c}$ is a vector of the expansion coefficients $\{c_j\}_{j=1}^N$. Since the radial functions being used are infinitely differentiable, the $L\boldsymbol{\phi}_{ij}$ terms can be found exactly. Solving the system (2.8) results in the expansion coefficients $\{c_j\}_{j=1}^N$ for the approximation (2.7). Although Dirichlet boundary conditions are given in (2.6), this method can easily be extended to a problem with Neumann boundary conditions. To do this, $\boldsymbol{\phi}$ from (2.8) is replaced with $G\boldsymbol{\phi}$ which is of the form

$$G\boldsymbol{\phi}_{ij} = G\phi(\|\mathbf{x}_i - \xi_j\|, \varepsilon),\ \mathbf{x}_i \in \partial\Omega,\ \xi_j \in X$$

where G is the operator on the boundary and $\mathbf{g}$ is replaced according to the new boundary data.

Kansa's method is called non-symmetric as it results in a non-symmetric matrix of the form (2.8) unlike the symmetric interpolation matrix that was used for RBF interpolation in §2.2. Also, this resultant matrix is composed of terms $L\phi(r, \varepsilon)$ which may not be positive definite. Since Kansa's method results in a matrix of a different form, uniqueness results from RBF interpolation do not carry over to this method. Even though there is no guarantee of a unique solution, Kansa [18] shows that this method often works and is very effective at approximating various PDEs. There does exist an alternate symmetric collocation method for PDEs based on a Hermite interpolation approach

that guarantees a unique collocation matrix [22], however we will not delve into the method in this work. Like RBF interpolation methods, collocation methods for PDEs suffer from the same conditioning problems mentioned in §2.2.1.

2.3.2 Galerkin-RBF Method

In [25], Wendland introduces a Galerkin approach using RBFs as a finite dimensional basis in order to solve PDEs. To demonstrate this method, we consider the second order elliptic boundary value problem

$$\begin{cases} -\Delta u(\mathbf{x}) + u(\mathbf{x}) = f(\mathbf{x}), \ \mathbf{x} \in \Omega \\ \\ \dfrac{\partial u}{\partial \nu}(\mathbf{x}) = 0, \ \mathbf{x} \in \partial\Omega \end{cases}$$

where Ω is some bounded domain in $\mathbb{R}^d$, $\partial\Omega$ is the boundary of Ω, Δ is the Laplacian operator $\Delta = \sum_{i=1}^{d} \dfrac{\partial^2}{\partial x_i^2}$, ν denotes the unit normal vector to the boundary $\partial\Omega$, and f is a given forcing function. The method is then derived using the usual approach. Multiplying the equation

$$-\Delta u(\mathbf{x}) + u(\mathbf{x}) = f(\mathbf{x})$$

by a test function

$$v \in V = \{v \in L^2(\Omega) : \frac{\partial v}{\partial x_i} \in L^2(\Omega)\}$$

where $L^2(\Omega) = \{f : \Omega \to \mathbb{R} \mid \int_\Omega |f|^2 < \infty\}$, and integrating over the domain Ω, we get

$$-\int_\Omega \Delta u(\mathbf{x}) v(\mathbf{x}) d\mathbf{x} + \int_\Omega u(\mathbf{x}) v(\mathbf{x}) d\mathbf{x} = \int_\Omega f(\mathbf{x}) v(\mathbf{x}) d\mathbf{x}. \tag{2.9}$$

Using Green's Theorem

$$\int_\Omega \Delta u(\mathbf{x}) v(\mathbf{x}) d\mathbf{x} + \int_\Omega \nabla u(\mathbf{x}) \cdot \nabla v(\mathbf{x}) d\mathbf{x} = \int_{\partial\Omega} \Big(\frac{\partial u}{\partial \nu}(x)\Big) v(\mathbf{x}) d\mathbf{s}$$

we can now reformulate (2.9) as

$$\int_\Omega \nabla u(\mathbf{x}) \cdot \nabla v(\mathbf{x}) d\mathbf{x} - \int_{\partial\Omega} \Big(\frac{\partial u}{\partial \nu}(\mathbf{x})\Big) v(\mathbf{x}) d\mathbf{s} + \int_\Omega u(\mathbf{x}) v(\mathbf{x}) d\mathbf{x} = \int_\Omega f(\mathbf{x}) v(\mathbf{x});$$

however the term $\int_{\partial\Omega} \Big(\dfrac{\partial u}{\partial \nu}(\mathbf{x})\Big) v(\mathbf{x}) d\mathbf{s}$ vanishes due to the boundary condition, leaving the weak formulation

$$\int_\Omega \nabla u(\mathbf{x}) \cdot \nabla v(\mathbf{x}) d\mathbf{x} + \int_\Omega u(\mathbf{x}) v(\mathbf{x}) d\mathbf{x} = \int_\Omega f(\mathbf{x}) v(\mathbf{x}) d\mathbf{x}, \ \forall v \in V. \tag{2.10}$$

To make this problem computationally feasible, instead of the functions u and v in an infinite dimensional space V, we consider functions u_N and v_N in a finite dimensional subspace $V_N \subset V$. Then (2.10) can be restated as seeking $u_N \in V_N$ such that

$$\int_\Omega \nabla u_N(\mathbf{x}) \cdot \nabla v_N(\mathbf{x}) d\mathbf{x} + \int_\Omega u_N(\mathbf{x}) v_N(\mathbf{x}) d\mathbf{x} = \int_\Omega f(\mathbf{x}) v_N(\mathbf{x}) d\mathbf{x}, \ \forall v_N \in V_N. \tag{2.11}$$

Galerkin-RBF methods choose this finite dimensional subspace to be a linear combination RBFs. The subspace considered is

$$V_N = \text{span}\{\phi(\|\mathbf{x} - \mathbf{x}_1\|, \varepsilon), \ldots, \phi(\|\mathbf{x} - \mathbf{x}_N\|, \varepsilon)\}$$

where ϕ is chosen to be a positive definite RBF so that the polynomial reproduction terms are not needed to form the basis and $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is a set of distinct points in Ω. Since u_N is in the space V_N it can be represented as

$$u_N(x) = \sum_{j=1}^{N} c_j \phi(\|\mathbf{x} - \mathbf{x}_j\|, \varepsilon), \tag{2.12}$$

thus we have

$$\int_\Omega \sum_{j=1}^{N} c_j \nabla \phi(\|\mathbf{x} - \mathbf{x}_j\|, \varepsilon) \cdot \nabla v_N(\mathbf{x}) d\mathbf{x} + \int_\Omega \sum_{j=1}^{N} c_j \phi(\|\mathbf{x} - \mathbf{x}_j\|, \varepsilon) v_N(\mathbf{x}) d\mathbf{x} = \int_\Omega f(\mathbf{x}) v_N(\mathbf{x}), \ \forall v_N \in V_N$$

which can be rewritten as the following N linear equations

$$\sum_{j=1}^{N} c_j \int_\Omega \nabla \phi(\|\mathbf{x} - \mathbf{x}_j\|, \varepsilon) \cdot \nabla \phi(\|\mathbf{x} - \mathbf{x}_i\|, \varepsilon) d\mathbf{x} + \sum_{j=1}^{N} c_j \int_\Omega \phi(\|\mathbf{x} - \mathbf{x}_j\|, \varepsilon) \phi(\|\mathbf{x} - \mathbf{x}_i\|, \varepsilon) d\mathbf{x}$$
$$= \int_\Omega f(\mathbf{x}) \phi(\|\mathbf{x} - \mathbf{x}_i\|, \varepsilon) d\mathbf{x}, \ i = 1, \ldots, N$$

since testing against each $v_N \in V_N$ is equivalent to testing against each element of the basis. An $N \times N$ linear system of equations can then be constructed and solved to find the unknown expansion coefficients $c_1, \ldots, c_N$ for the approximation (2.12).

The linear system for this Galerkin-RBF approach results in a dense matrix if the RBF that is being used is globally supported like those in Table 2.1. Usually Galerkin approaches shy away from these computationally expensive problems; one way to do this with RBFs would be to use compactly supported RBFs. However, using compactly supported RBFs would lose some of the convergence properties that globally supported RBFs would contain. For more details on Galerkin approaches for PDEs used in the finite element setting refer to [4]. For details on the convergence properties of Galerkin-RBF methods refer to [25].

2.4 RBF-generated Finite Difference Stencils

The RBF methods that have been introduced so far have been global methods meaning that the approximations are formed using points throughout the target domain. Since we have been considering globally supported RBFs, these global RBF methods require a dense linear system to be solved in order to come up with an approximation. These dense linear system solves can become computationally expensive when a large amount of points are used. The idea behind RBF-FD stencils is to avoid these large dense linear systems by finding approximations using only nearby or local points. Approximation with RBFs using only local points sacrifices possible spectral for high order algebraic convergence, but computational efficiency is gained especially since these methods are more capable of utilizing multicore computing architectures [13]. These RBF-FD stencil methods have been shown to work well in various applications. Some applications for which these methods have been tested are geophysical problems on the sphere [13], the incompressible Navier-Stokes equations [20], and the Hamilton-Jacobi equations [6].

Finite difference stencil rules seek to find an approximation to a partial differential operator L acting on an unknown function u in terms of the function u at nearby points. These stencil rules take the form

$$Lu(\mathbf{x}_c) \approx \sum_{i=1}^{n} c_i u(\mathbf{x}_i) \tag{2.13}$$

where $\mathbf{x}_c \in \mathbb{R}^d$ is the point at which the operator is approximated, $c_1, \ldots, c_n$ are stencil weights for a given rule and the terms $u(\mathbf{x}_1), \ldots, u(\mathbf{x}_n)$ are unknowns at spatial points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ that are nearby and often include the point $\mathbf{x}_c$. For example, consider the standard five point stencil rule for the Laplacian operator in two dimensions on a uniform grid with spacing h

$$\Delta u(x,y) \approx \frac{u(x-h,y) + u(x+h,y) + u(x,y-h) + u(x,y+h) - 4u(x,y)}{h^2}.$$

This stencil rule gives a second order approximation (i.e., the error term is $\mathcal{O}(h^2)$) to the Laplacian operator acting on the function u at the point (x,y) as h decreases. Typically stencil weights for standard finite difference stencil rules are derived by manipulating Taylor series expansions to find a form of $Lu(\mathbf{x})$ that has a desired error term. However, finding a high order finite difference formula with a desired error term is often difficult to find unless the points being used are uniformly spaced. Standard finite difference rules have the relationship that a rule of order n should be exact for all

polynomials $P_n(\mathbf{x})$ of degree $\leq n$. An alternate approach for finding stencil weights for a standard finite difference rule of order n would be to guarantee (2.13) is exact for polynomials belonging to a finite dimensional basis set of the polynomials $P_n(\mathbf{x})$. If these stencil weights satisfy all functions of a basis set of $P_n(\mathbf{x})$ then they should satisfy all of $P_n(\mathbf{x})$. As an example, we consider finding the stencil weights that are exact for all $P_2(x)$ with the first derivative operator acting at a point x_c. A basis set of functions for $P_2(x)$ is $\{1, x, x^2\}$, so we want the stencil weights to satisfy the following equations

$$\sum_{i=1}^{3} c_i = \frac{d}{dx}(1)|_{x=x_c}$$
$$\sum_{i=1}^{3} c_i x_i = \frac{d}{dx}(x)|_{x=x_c}$$
$$\sum_{i=1}^{3} c_i x_i^2 = \frac{d}{dx}(x^2)|_{x=x_c}$$

which can be formed into the linear system

$$\begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 2x_c \end{pmatrix}.$$

The stencil weights can then be solved for to find a finite difference rule using the points $\{x_i\}_{i=1}^3$. Notice that we choose our set points $\{x_i\}_{i=1}^3$ to consist of only three points, this is so the system is neither underdetermined or overdetermined. This means that a three point finite difference rule can be used to satisfy all polynomials of degree ≤ 2. The matrix in the linear system that results from this approach is the Vandermonde matrix which is known to be ill-conditioned as the number of terms grow so it is not an ideal way to find stencil weights. It is known that as these points x_i get closer to the point x_c the error in the finite difference stencil decreases.

The goal of RBF-generated finite difference stencils is to find a finite difference stencil approximation like (2.13), however the stencil weights for the rule are found so that they are exact for RBFs rather than polynomials. To find the stencil weights $c_1, \ldots, c_n$, we require (2.13) to be exact when $u(\mathbf{x})$ is replaced by RBF translates $\phi(\|\mathbf{x} - \mathbf{x}_j\|, \varepsilon),\ j = 1, \ldots, n$. Choosing an RBF ϕ from Table 2.1 and replacing u with RBF translates to the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, we then have n linear

equations

$$\sum_{i=1}^{n} c_i \phi(\|\mathbf{x}_i - \mathbf{x}_j\|, \varepsilon) = L\phi(\|\mathbf{x}_c - \mathbf{x}_j\|, \varepsilon),\ j = 1, \ldots, n.$$

We can then form a linear system of equations

$$\begin{pmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|, \varepsilon) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|, \varepsilon) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_n\|, \varepsilon) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|, \varepsilon) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|, \varepsilon) & \cdots & \phi(\|\mathbf{x}_2 - \mathbf{x}_n\|, \varepsilon) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_n - \mathbf{x}_1\|, \varepsilon) & \phi(\|\mathbf{x}_n - \mathbf{x}_2\|, \varepsilon) & \cdots & \phi(\|\mathbf{x}_n - \mathbf{x}_n\|, \varepsilon) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} L\phi(\|\mathbf{x}_c - \mathbf{x}_1\|, \varepsilon) \\ L\phi(\|\mathbf{x}_c - \mathbf{x}_2\|, \varepsilon) \\ \vdots \\ L\phi(\|\mathbf{x}_c - \mathbf{x}_n\|, \varepsilon) \end{pmatrix} \tag{2.14}$$

that is solved to find stencil weights for the RBF generated finite difference stencil. Notice that the matrix in (2.14) is the same as the RBF interpolation matrix in (2.3); this means that the matrix is nonsingular for choices of positive definite RBFs. Polynomial reconstruction constraints can be added to the system so that matrix is nonsingular in the case of a conditionally positive definite RBF. Since the RBF ϕ being used is infinitely differentiable the exact value in the right hand side of the system can be found exactly.

Now we want to use these RBF generated finite difference stencil rules in order to approximate a PDE. We consider the time independent PDE

$$\begin{cases} Lu(\mathbf{x}) = f(\mathbf{x}),\ \mathbf{x} \in \Omega \\ \\ u(x) = g(\mathbf{x}),\ \mathbf{x} \in \partial\Omega \end{cases}$$

where Ω is some bounded domain of $\mathbb{R}^d$, $\partial\Omega$ is the boundary of Ω, L is a linear partial differential equation operator, f is the forcing function for the PDE, and g is the given boundary data. The first step is to discretize Ω into a set of N locations $X = \{\mathbf{x}_k\}_{k=1}^N$. Then for each point $\mathbf{x}_k$, stencil weights $\mathbf{c}_k = \{c_{k1}, \ldots, c_{kn}\}$ are found for an RBF-generated finite difference rule using $n < N$ nearby points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in X$. If the points in X are on a uniform grid, the stencil calculation only needs to be done once as all stencils weights would be the same. Otherwise on a nonuniform grid, stencil weights need to be found for each $\mathbf{x}_k$, however this can be done as a preprocessing step which can save a lot of computation especially if a time dependent problem were to be considered.

Using the stencil weights $\mathbf{c}_k$, a system of linear equations can then be formed with the stencil rules that can be solved for the unknowns $u(\mathbf{x}_k)$ for points $\mathbf{x}_k \in \Omega$ as u is known on $\partial\Omega$. If the

stencil uses a point on the boundary, we can substitute into the stencil the known value from our boundary data and move it to the right hand side of the system. On a uniform grid, the system results in a banded matrix like that of standard finite difference methods, e.g. the five point rule for the Laplacian in two dimensions. On a nonuniform grid, a banded matrix can also be formed with a judicious numbering of the points $\mathbf{x}_k$.

2.4.1 Numerical Examples

For a numerical example using RBF-FD stencil rules, we will look at approximating the Laplacian operator Δ acting on a one-dimensional function u at a point x. The RBF-FD stencil rule we compute uses stencil points corresponding to those used in the standard second order centered finite difference formula in one dimension. The standard second order centered finite difference formula in one dimension takes the form

$$\Delta u(x) = u''(x) \approx \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}$$

where $h > 0$ is some fixed constant. The error for this centered difference formulas is known to be $\mathcal{O}(h^2)$. Since we know how the error in the centered difference formula acts as h changes, we are interested in comparing the error between the centered difference formula and RBF-FD stencil rules when the same stencil points are used.

To numerically test the RBF-FD stencil rule, we use the function $u(x) = \sin(\pi x)$ where the exact value of the Laplacian is $\Delta u(x) = -\pi^2 \sin(\pi x)$. In our tests, we use Gaussian RBFs with a shape parameter $\varepsilon = 1.0$ in order to solve problem (2.14) for the stencil weights of the RBF-FD stencil rule. Table 2.4 gives numerical results from approximating the test problem at an arbitrarily chosen point $x = .5$ for both the second order centered finite difference stencil rule and the RBF-FD stencil rule as h is decreased. From these results, we see that the error of the RBF-FD stencil rules is similar to those of the second centered difference formula and decreases at a similar second order rate as h is decreased. Interestingly if we look at the weights for both stencil rules in Table 2.5, we observe that the weights are similar but slightly different for all values of h. It turns out that the RBF-FD stencil weights approach the centered difference weights as $h \to 0$ and $\varepsilon \to 0$. Although our numerical example only tests the RBF stencil rule on a uniform grid, Flyer et al. [13] have successfully used this approach on a scattered grid.

Table 2.4: RBF-FD and second order centered finite difference stencil rules applied where $u(x) = \sin(\pi x)$.

	RBF-FD Stencil		Centered Finite Difference Stencil	
h	Relative Error	Conv. Rate	Relative Error	Conv. Rate
.5	3.46602e-03	-	1.89430e-01	-
.25	3.37804e-03	0.04	5.03588e-02	1.91
.125	1.10480e-03	1.61	1.27852e-02	1.98
0.0625	2.92336e-04	1.92	3.20864e-03	1.99
0.03125	7.40904e-05	1.98	8.02932e-04	2.00

Table 2.5: RBF-FD and second order centered finite difference stencil weights.

	RBF-FD Stencil Points / Weights			Centered Finite Difference Stencil Points / Weights		
h	$x-h$	x	$x+h$	$x-h$	x	$x+h$
.5	5.03042	-9.83540	5.03042	4.0	-8.0	4.0
.25	17.00975	-33.95837	17.00975	16.0	-32.0	16.0
.125	65.00256	-129.98958	65.00256	64.0	-128.0	64.0
.0625	257.00065	-513.99740	257.00065	256.0	-512.0	256.0
0.03125	1025.000163	-2049.99935	1025.000163	1024.0	-2048.0	1024.0

CHAPTER 3

USING RBFS TO APPROXIMATE THE 1-D NONLOCAL DIFFUSION PROBLEM

Now that an approach for determining RBF-FD stencils has been introduced, we want to extend this concept to formulate RBF-generated quadrature rules for a nonlocal integral and apply these rules to approximate a nonlocal problem. However, the method introduced in Chapter 2 was used to approximate derivatives not integrals. The goal of this chapter is to introduce an approach similar to the one in §2.4 to obtain an RBF-generated quadrature formula for a nonlocal integral. This quadrature rule is then used to approximate the nonlocal problem of interest.

This chapter begins by deriving a method for approximating a one-dimensional nonlocal integral. This method will then be used to numerically approximate nonlocal integrals for some test problems. A strategy will then be devised to approximate a complete nonlocal problem. Finally, numerical results for a nonlocal problem will be given and compared with those obtained by using a standard finite element approximation.

3.1 RBF-generated Quadrature Rules

The one-dimensional nonlocal integral we are interested in approximating has the following form

$$\int_{x-\delta}^{x+\delta} \frac{u(x)-u(z)}{|x-z|^{1+2s}} dz \tag{3.1}$$

where $\delta > 0$ is the horizon length and $s \in [0, 1)$. For nonlocal integrals, the horizon is defined by all of the points within a δ radius of a point, i.e. in one dimension the horizon is the interval $[x-\delta, x+\delta]$. The goal is to devise a quadrature rule that can approximate (3.1) using a finite set of N quadrature points $\{x_1, x_2, \ldots, x_N\} \in [x-\delta, x+\delta]\backslash\{x\}$. The point x is excluded from the quadrature points due to the discontinuity in the integrand at that point. In order to produce quadrature weights corresponding to our quadrature points, a strategy similar to the RBF-generated FD stencil discussed in §2.4 is used.

Like the RBF-FD stencil, we want a method for approximating an operator locally around a point of interest, however instead of a differential operator the integral operator in (3.1) is considered. This integral operator is approximated at a point x_c by a linear combination of the function values at N quadrature points $\{x_1, \ldots, x_N\}$ in the horizon of x_c. An approximation to the one-dimensional nonlocal integral then takes the following form

$$\int_{x_c-\delta}^{x_c+\delta} \frac{u(x_c)-u(z)}{|x_c-z|^{1+2s}}dz \approx \sum_{i=1}^{N} c_i \frac{u(x_c)-u(x_i)}{|x_c-x_i|^{1+2s}} \tag{3.2}$$

where $\{c_1, \ldots, c_N\}$ are the quadrature weights corresponding to the quadrature points $\{x_1, \ldots, x_N\}$. It is important to realize that we are not deriving a quadrature rule as one normally does for $\int_a^b f(z)dz$, but rather when the integrand has the specific form of (3.1).

To determine the quadrature weights, we require (3.2) to be exact when $u(x)$ is replaced by RBF translates $\phi(|x-x_j|,\varepsilon),\ j=1,\ldots,N$. This gives N equations

$$\int_{x_c-\delta}^{x_c+\delta} \frac{\phi(|x_c-x_j|,\varepsilon)-\phi(|z-x_j|,\varepsilon)}{|x_c-z|^{1+2s}}dz = \sum_{i=1}^{N} c_i \frac{\phi(|x_c-x_j|,\varepsilon)-\phi(|x_i-x_j|,\varepsilon)}{|x_c-x_i|^{1+2s}},\ j=1,\ldots,N.$$

This system of equations then yields the following problem in matrix form

$$\begin{pmatrix} \frac{\phi(|x_c-x_1|,\varepsilon)-\phi(|x_1-x_1|,\varepsilon)}{|x_c-x_1|^{1+2s}} & \frac{\phi(|x_c-x_1|,\varepsilon)-\phi(|x_1-x_2|,\varepsilon)}{|x_c-x_2|^{1+2s}} & \cdots & \frac{\phi(|x_c-x_1|,\varepsilon)-\phi(|x_1-x_N|,\varepsilon)}{|x_c-x_N|^{1+2s}} \\ \frac{\phi(|x_c-x_2|,\varepsilon)-\phi(|x_2-x_1|,\varepsilon)}{|x_c-x_1|^{1+2s}} & \frac{\phi(|x_c-x_2|,\varepsilon)-\phi(|x_2-x_2|,\varepsilon)}{|x_c-x_2|^{1+2s}} & \cdots & \frac{\phi(|x_c-x_2|,\varepsilon)-\phi(|x_2-x_N|,\varepsilon)}{|x_c-x_N|^{1+2s}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\phi(|x_c-x_N|,\varepsilon)-\phi(|x_N-x_1|,\varepsilon)}{|x_c-x_1|^{1+2s}} & \frac{\phi(|x_c-x_N|,\varepsilon)-\phi(|x_N-x_2|,\varepsilon)}{|x_c-x_2|^{1+2s}} & \cdots & \frac{\phi(|x_c-x_N|,\varepsilon)-\phi(|x_N-x_N|,\varepsilon)}{|x_c-x_N|^{1+2s}} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} \tag{3.3}$$

$$= \begin{pmatrix} \int_{x_c-\delta}^{x_c+\delta} \frac{\phi(|x_c-x_1|,\varepsilon)-\phi(|z-x_1|,\varepsilon)}{|x_c-z|^{1+2s}}dz \\ \int_{x_c-\delta}^{x_c+\delta} \frac{\phi(|x_c-x_2|,\varepsilon)-\phi(|z-x_2|,\varepsilon)}{|x_c-z|^{1+2s}}dz \\ \vdots \\ \int_{x_c-\delta}^{x_c+\delta} \frac{\phi(|x_c-x_N|,\varepsilon)-\phi(|z-x_N|,\varepsilon)}{|x_c-z|^{1+2s}}dz \end{pmatrix}$$

which can be solved to find the quadrature weights $c_1, \ldots, c_N$ that are used in (3.2) to approximate the nonlocal integral at x_c with a horizon defined by δ.

In §2.4, it was shown that the problem that needs to be solved in order to come up with the stencil weights for RBF-FD stencil rules involves using the RBF interpolation matrix from (2.3). This matrix guarantees that the problem is nonsingular when the RBF is chosen to be a positive definite function. Finding quadrature weights for the RBF-generated quadrature rule

involves solving the problem (3.3) whose matrix is different from than that of the RBF interpolation matrix. In fact, with scattered points the matrix in (3.3) is not even symmetric. Since the matrix in (3.3) is of a different form, we cannot use previous results to prove that it is invertible. However, numerical evidence from tests throughout this chapter indicate that this problem is invertible.

Another difference between the RBF-FD stencil rule and RBF-generated quadrature rule is the right-hand side terms in the linear system that are needed to solve for quadrature weights. The RBF-FD stencil rule has a right-hand side built of RBF derivatives which can be found exactly while the RBF-generated quadrature has a right-hand side requiring integration, i.e. evaluating

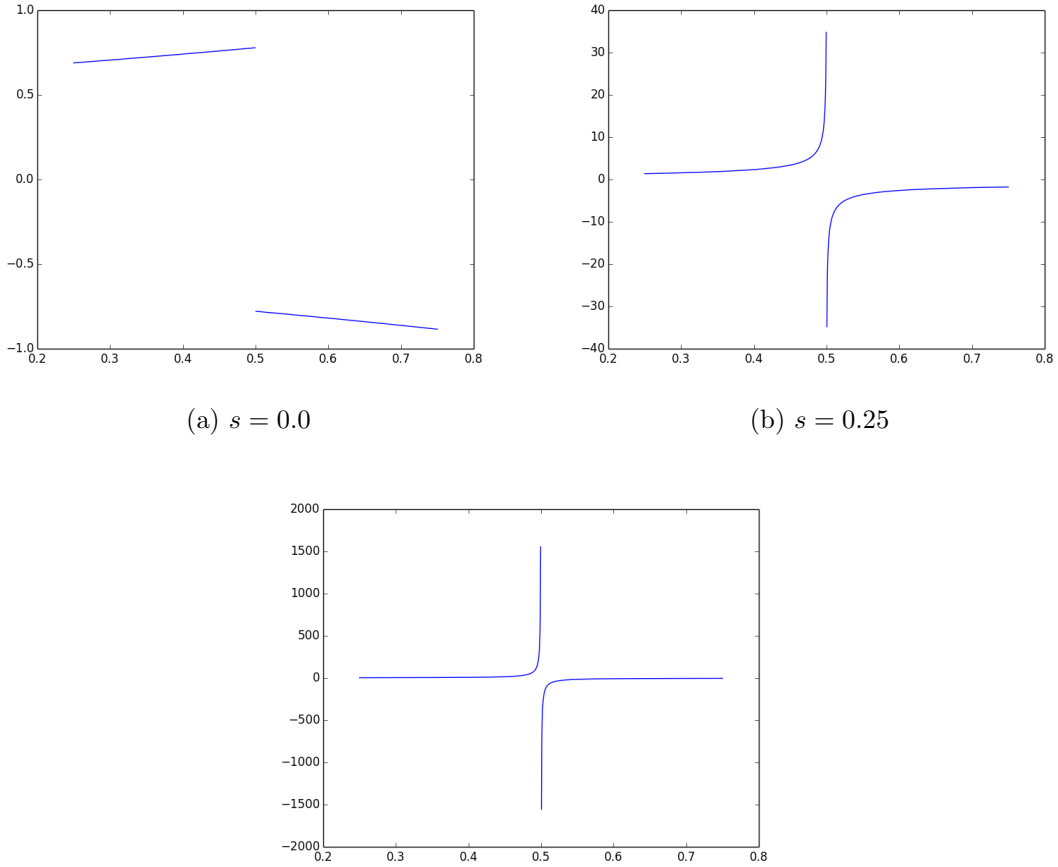
$$\int_{x_c-\delta}^{x_c+\delta} \frac{\phi(|x_c - x_i|, \varepsilon) - \phi(|z - x_i|, \varepsilon)}{|x_c - z|^{1+2s}} dz, \;\; i = 1, \ldots, N, \tag{3.4}$$

which is often difficult to find exactly. In Figure 3.1, some examples of the integrands for these right-hand side integrals are shown for different values of s. We observe that these functions are discontinuous and this discontinuity gets worse as s grows, so these right-hand side terms become more difficult to find as s increases. In §3.2, we will discuss our strategy for finding these right-hand side integral terms.

3.2 Numerical Results for Nonlocal Integrals

To demonstrate the ability of our RBF quadrature method to approximate an integral, we look at some test problems where $u(x)$ is chosen in such a way that the exact integral is known. We are interested in observing how the approximation error behaves as $\delta \to 0$ for a fixed number of quadrature points and how it behaves as δ is fixed and the number of quadrature points is increased. Also, we are interested in how the approximation performs for different values of s.

The formation of problem (3.3) for finding quadrature weights now needs to be addressed. Since it was specified that the quadrature points satisfy $x_i \neq x_c, \;\; i = 1, \ldots, N$ there are no problems calculating the terms of the matrix in (3.3). The integrals terms (3.4) in the right-hand side of (3.3) are not so simple to find. For smaller values of s these terms can be found exactly using a symbolic algebra package such as Mathematica™. However in practice, we numerically approximate the right-hand side terms using a composite Gauss quadrature rule that avoids calculations at the known point of discontinuity x_c. We found that this method for finding the right-hand side terms is adequate for most cases of s and δ, however there is reason to believe that there can be some



(a) $s = 0.0$

(b) $s = 0.25$

(c) $s = 0.5$

Figure 3.1: Discontinuous integrand from (3.4) using a Gaussian RBF.

issues using this approach. The first issue to point out is that as δ is decreased, more of the region that is to be integrated is dominated by a discontinuous region. Also, as seen in Figure 3.1, the discontinuity in the integrand of the nonlocal integral becomes worse as s increases. Thus, a good numerical approximation to these terms can become very difficult to find due to the discontinuity. Now consider trying to solve (3.3) with a matrix that is ill-conditioned, which occurs frequently in this problem, and a right-hand side that does not have a very good approximation to the exact value. In this case, the solution is not going to give the correct quadrature weights. Thus for this method to work well, it is imperative to get a very good approximation to these right-hand side terms. In the test problems to come, we will see a few cases where the quadrature weights from

our RBF quadrature method are not correct due to this situation.

3.2.1 Test Problems

Before any test cases are considered, first we must generate a function $u(x)$ where the nonlocal integral (3.1) can be computed exactly. This is typically done using a Taylor series expansion [7]. The initial nonlocal integral is split into two terms

$$\int_{x-\delta}^{x+\delta} \frac{u(x)-u(z)}{|x-z|^{1+2s}}dz = \int_{x}^{x+\delta} \frac{u(x)-u(z)}{|x-z|^{1+2s}}dz + \int_{x-\delta}^{x} \frac{u(x)-u(z)}{|z-x|^{1+2s}}dz.$$

Defining a change of variable for z so that $z = x + \Delta x$ and $dz = d\Delta x$ for $-\delta \leq \Delta x \leq \delta$, we can make the integral dependent on Δx so that the integral is now

$$\int_{-\delta}^{0} \frac{u(x)-u(x+\Delta x)}{|x-(x+\Delta x)|^{1+2s}}d\Delta x + \int_{0}^{\delta} \frac{u(x)-u(x+\Delta x)}{|(x+\Delta x)-x|^{1+2s}}d\Delta x. \tag{3.5}$$

Upon simplifying (3.5) becomes

$$\int_{-\delta}^{0} \frac{u(x)-u(x+\Delta x)}{|-\Delta x|^{1+2s}}d\Delta x + \int_{0}^{\delta} \frac{u(x)-u(x+\Delta x)}{|\Delta x|^{1+2s}}d\Delta x. \tag{3.6}$$

Writing the term $u(x+\Delta x)$ as the Taylor series

$$u(x+\Delta x) = u(x) + \Delta x u'(x) + \Delta x^2 \frac{u''(x)}{2!} + \cdots \tag{3.7}$$

and inserting this term into (3.6), we now have an integral that is computable. In particular, we have that

$$\int_{x-\delta}^{x+\delta} \frac{u(x)-u(z)}{|x-z|^{1+2s}}dz = -u'(x)\int_{-\delta}^{\delta} \frac{\Delta x}{|\Delta x|^{1+2s}}d\Delta x - \frac{u''(x)}{2}\int_{-\delta}^{\delta} \frac{\Delta x^2}{|\Delta x|^{1+2s}}d\Delta x + \cdots. \tag{3.8}$$

Now, if the integral terms on the right of (3.8) have integrands that are odd functions then the term evaluates to zero due to symmetry leaving only the integral terms with integrands that are even functions. For example when $s = 0$, we have that

$$\int_{x-\delta}^{x+\delta} \frac{u(x)-u(z)}{|x-z|}dz = -\frac{u''(x)}{2}\int_{-\delta}^{\delta} \frac{\Delta x^2}{|\Delta x|}d\Delta x - \frac{u^{(4)}(x)}{24}\int_{-\delta}^{\delta} \frac{\Delta x^4}{|\Delta x|}d\Delta x + \cdots.$$

If $u(x)$ is chosen to be a polynomial then the value of the nonlocal integral can be found exactly. If δ is kept small (i.e., $\delta < 1$), the nonlocal integral with most functions $u(x)$ can be represented well with only a few terms of this Taylor expansion as the higher power terms of Δx become insignificant. A good representation of the nonlocal integral can be found even with a function $u(x)$ whose higher derivatives do not vanish. This method will be used repeatedly throughout this section to find exact solutions to nonlocal integrals and in §3.4 to find forcing functions to be used when solving the complete nonlocal problem.

3.2.2 Results for the $s = 0$ Case on a Uniform Grid

Initially, we consider symmetric RBF-generated quadrature rules on a uniform grid. Let N represent an even number of quadrature points chosen over the horizon where $\frac{N}{2}$ quadrature points come from points in $[x_c - \delta, x_c)$ and $\frac{N}{2}$ quadrature points are in $(x_c, x_c + \delta]$. Using the points $x_c - \delta$ and $x_c + \delta$ as quadrature points insures that the approximation is done over the entire horizon, but is not necessary. Table 3.1 shows how our quadrature points are chosen for several values of N on a uniform grid. Although RBF schemes allow the use of any set of quadrature points, uniform points were initially chosen as they allow for a systematic way of looking at our approximation as the number of quadrature points and the size of the horizon changes.

Table 3.1: Uniform quadrature point choices for several cases of N.

N	Quadrature Points in $[x_c - \delta, x_c)$	Quadrature Points in $(x_c, x_c + \delta]$
2	$x_c - \delta$	$x_c + \delta$
4	$x_c - \delta, x_c - \frac{\delta}{2}$	$x_c + \frac{\delta}{2}, x_c + \delta$
6	$x_c - \delta, x_c - \frac{2\delta}{3}, x_c - \frac{\delta}{3}$	$x_c + \frac{\delta}{3}, x_c + \frac{2\delta}{3}, x_c + \delta$
8	$x_c - \delta, x_c - \frac{3\delta}{4}, x_c - \frac{\delta}{2}, x_c - \frac{\delta}{4}$	$x_c + \frac{\delta}{4}, x_c + \frac{\delta}{2}, x_c + \frac{3\delta}{4}, x_c + \delta$

To see how well the RBF quadrature method works, the nonlocal integral is approximated using the following test functions $u(x) = x^4 - x^2$ and $u(x) = \sin(x)$. Approximations are formed by choosing quadrature points from Table 3.1 and solving (3.3) to find quadrature weights (using a composite Gauss quadrature rule to find the right-hand side of (3.3)) and evaluating the right side of (3.2) with the given quadrature points and weights. These approximate values are then compared with exact values calculated using the Taylor series method mentioned earlier. We begin by looking at approximations to the nonlocal integral for the case where $s = 0$.

Standard quadrature rules derived via polynomial interpolation such as the trapezoid rule and Simpson's Rule have convergence rates that are dependent on the length of the integration interval. Thus for our RBF quadrature rule on a nonlocal integral which is derived via RBF interpolation, we are interested to see if similar convergence rates exists as the horizon decreases. Using the Gaussian RBF with $\varepsilon = 1.0$ and an arbitrarily chosen point $x_c = .5$, the results of the RBF quadrature method using two and four uniform quadrature points with decreasing horizon are shown for $u(x) = x^4 - x^2$

in Table 3.2 and Table 3.3 respectfully, and for $u(x) = \sin(x)$ in Table 3.4 and Table 3.5. From Table 3.2 and Table 3.4, we see that error reduces at a second order rates for both test functions using an RBF quadrature rule with two uniform quadrature points. Likewise in Table 3.3 and Table 3.5, error reduces at fourth order rate using RBF quadrature with four uniform quadrature points for both test functions. Overall these convergence rates are not surprising since normal quadrature methods like the two quadrature point trapezoid rule and three quadrature point Simpson's rule have second and third order convergence rates as the integration region is decreased.

Table 3.2: RBF quadrature as δ decreases using $N = 2$ uniform quadrature points for $s = 0$ and $u(x) = x^4 - x^2$. Only one quadrature weight is shown as the weights are symmetric due to points being uniformly spaced.

δ	Quad. Weight	Relative Error	Conv. Rate
.25	1.26690e-01	7.31508e-02	-
.125	6.27365e-02	1.92276e-02	1.93
.0625	3.12803e-02	4.86392e-03	1.98
.03125	1.56288e-02	1.21952e-03	1.99

Table 3.3: RBF quadrature as δ decreases using $N = 4$ uniform quadrature points for $s = 0$ and $u(x) = x^4 - x^2$. Only two quadrature weights are shown as the weights are symmetric due to points being uniformly spaced.

δ	Quad. Weights		Relative Error	Conv. Rate
.25	4.19472e-02	1.6611e-01	2.22570e-04	-
.125	2.08692e-02	8.32618e-02	1.49057e-05	3.90
.0625	1.04212e-02	4.16576e-02	9.47271e-07	3.97
.03125	5.20890e-03	2.08322e-02	7.84456e-08	3.59

Table 3.4: RBF quadrature as δ decreases using $N = 2$ uniform quadrature points for $s = 0$ and $u(x) = \sin(x)$. Only one quadrature weight is shown as the weights are symmetric due to points being uniformly spaced.

δ	Quad. Weight	Relative Error	Conv. Rate
.25	1.2669e-01	1.08961e-02	-
.125	6.27365e-02	3.13149e-03	1.80
.0625	3.12803e-02	8.06185e-04	1.96
.03125	1.56288e-02	2.02977e-04	1.99

Table 3.5: RBF quadrature as δ decreases using $N = 4$ uniform quadrature points for $s = 0$ and $u(x) = \sin(x)$. Only two quadrature weights are shown as the weights are symmetric due to points being uniformly spaced.

δ	Quad. Weights		Relative Error	Conv. Rate
.25	4.19472e-02	1.6611e-01	2.06343e-05	-
.125	2.08692e-02	8.32618e-02	1.36001e-06	3.92
.0625	1.04212e-02	4.16576e-02	8.73807e-08	3.96
.03125	5.20890e-03	2.08322e-02	5.29062e-09	3.77

The other thing we want to look at is how the approximation improves for a fixed horizon as the number of uniform quadrature points is increased. Using the Gaussian RBF and an arbitrarily chosen point $x_c = .5$, results for the RBF quadrature method are shown using a fixed horizon $\delta = .125$ and increasing number of stencil points with two different choices of the shape parameter $\varepsilon = 2.0$ and $\varepsilon = 10.0$ for $u(x) = x^4 - x^2$ in Table 3.6 and for $u(x) = \sin(x)$ in Table 3.7. The approximations with two different shape parameters are shown to demonstrate what happens as it is changed. For both test cases, spectral-like convergence rates are seen as the number of uniform quadrature points increase. For the lower value of the shape parameter, errors are lower and convergence rates are higher as expected. However, it is observed that for the lower shape parameter the condition of the linear system for solving for quadrature weights is much higher. This high condition number becomes an issue for both test cases when $N = 16$ quadrature points for the $\varepsilon = 2.0$ case. Since double precision arithmetic for these calculations are used, as the condition number goes past 10^{16} digits of accuracy for the quadrature weights are lost due to this ill-conditioned problem causing errors not to improve.

Table 3.6: RBF quadrature on a fixed horizon $\delta = .125$ with increasing number of uniform quadrature points for $s = 0$ and $u(x) = x^4 - x^2$. Condition denotes the condition number of the linear system used for finding quadrature weights.

	$\varepsilon = 2.0$			$\varepsilon = 10.0$		
N	Relative Error	Conv. Rate	Condition	Relative Error	Conv. Rate	Condition #
2	2.91241e-02	-	2.2e+00	6.56219e-01	-	1.7e+00
4	7.68881e-05	8.56	4.3e+03	9.22783e-03	6.15	1.2e+01
8	2.47053e-08	11.60	7.9e+12	7.37968e-05	6.97	2.6e+04
16	1.00798e-06	-5.35	2.8e+16	9.67794e-08	9.57	9.1e+14

Table 3.7: RBF quadrature on a fixed horizon $\delta = .125$ with increasing number of uniform quadrature points for $s = 0$ and $u(x) = \sin x$. Condition denotes the condition number of the linear system used for finding quadrature weights.

	$\varepsilon = 2.0$			$\varepsilon = 10.0$		
N	Relative Error	Conv. Rate	Condition #	Relative Error	Conv. Rate	Condition #
2	1.28715e-02	-	2.2e+00	0.630063	-	1.7e+00
4	2.31372e-05	9.12	4.3e+03	8.47793e-03	6.22	1.2e+01
8	2.65601e-08	9.77	7.9e+12	6.05205e-05	7.13	2.6e+04
16	9.62016e-07	-5.18	2.8e+16	5.58869e-07	6.76	9.1e+14

Another interesting thing to look at is how RBF quadrature compares to existing quadrature rules that are of a similar form. In order to do a comparison, we consider the RBF quadrature rule using two uniform points and the trapezoid rule which uses the same quadrature points, the end points of the domain of integration. We choose to solve a nonlocal integral using the test function $u(x) = x^4 - x^2$. Again, we are interested in observing the error as the horizon is decreased with δ. The results for the comparisons between these quadrature approximation are shown in Table 3.8. Unsurprisingly, the trapezoid rule does not converge due to the singularity in the integrand. This is expected since the trapezoid rule normally is only expected to work on continuous functions.

Table 3.8: RBF quadrature and the trapezoid rule using $N = 2$ uniform quadrature points for $s = 0$ and $u(x) = x^4 - x^2$. Only one quadrature weight is shown as the weights are symmetric due to points being uniformly spaced.

	RBF Quadrature			Trapezoid Rule		
δ	Quad. Weight	Relative Error	Conv. Rate	Quad. Weight	Relative Error	Conv. Rate
.25	1.2669e-01	7.31508e-02	-	0.25	1.11765	-
.125	6.27365e-02	1.92276e-02	1.93	0.125	1.03077	0.11
.0625	3.12803e-02	4.86392e-03	1.98	6.25e-02	1.00778	0.03
.03125	1.56288e-02	1.21952e-03	1.99	3.125e-02	1.00195	0.01

At this point, one might wonder why we do not use a composite Gaussian quadrature rule to evaluate the nonlocal integral since that is what we are using to approximate the right-hand side in (3.3). The reason is that our ultimate goal is to approximate the nonlocal diffusion equation and every quadrature point that is used to approximate the nonlocal integral is a point where the temporal derivative in the nonlocal diffusion problem must be approximated. Consequently the number of quadrature points used to approximate the nonlocal integral governs the size of

our discrete nonlocal problem. The process of approximating a nonlocal diffusion problem will be illustrated in §3.3.

3.2.3 Results for the $s = 0$ Case on a Nonuniform Grid

So far we have only considered using RBF quadrature with uniform quadrature points, however one of the nice properties of the RBF quadrature method is that it can work on nonuniform sets of quadrature points. We now want to take a look at a set of nonuniform quadrature points to see how they affect RBF quadrature approximation. The nonuniform quadrature points that are considered here are derived by specifying $x_c = 0$ and choosing the points $x_c - \frac{\delta}{2}$ and $x_c + \frac{\delta}{2}$ and tranforming these points with $\sin(\frac{\pi x}{2})$ so that they are within the horizon defined by δ. The points that result from this transformation are shown in Table 3.9 for several values of δ. These quadrature points are of interest not only because they are nonuniform but also they do not include the endpoints. Approximating the nonlocal integral for the case $u(x) = x^4 - x^2$ with the two nonuniform quadrature points from Table 3.9 gives the results in Table 3.10. Like the $N = 2$ case for uniform quadrature points, we observe second order convergence for these nonuniform quadrature points. This shows that the quadrature points do not need to be uniform or contain the end points for the RBF quadrature method to work. However, this does not guarantee that RBF-generated quadrature works for any set of points.

Table 3.9: Nonuniform quadrature point choices for several cases of δ.

δ	Horizon $(-\delta, \delta)$		Quadrature Points	
$\frac{1}{6}$	-0.166666	0.166666	-0.130526	0.130526
$\frac{1}{12}$	-0.083333	0.083333	-0.065403	0.065403
$\frac{1}{24}$	-0.041666	0.041666	-0.032719	0.032719
$\frac{1}{48}$	-0.020833	0.020833	-0.016362	0.016362

3.2.4 Results for the $s > 0$ Case on a Uniform Grid

Now that we have shown various test cases where $s = 0$, we want to look at cases with other values of s. In this section, numerical tests are performed using uniform quadrature points chosen from Table 3.1. The nonlocal integral approximated in this section will use the test function

Table 3.10: RBF quadrature using nonuniform quadrature points from Table 3.9 for $s = 0$ and $u(x) = x^4 - x^2$.

δ	Quad. Weights		Relative Error	Conv. Rate
$\frac{1}{6}$	1.0657e-01	1.0657e-01	1.67087e-03	-
$\frac{1}{12}$	5.31108e-02	5.31108e-02	4.08785e-04	2.03
$\frac{1}{24}$	2.65332e-02	2.65332e-02	1.01624e-04	2.01
$\frac{1}{48}$	1.32638e-02	1.32638e-02	2.53701e-05	2.0

$u(x) = x^4 - x^2$. Numerical results are found using the Gaussian RBF and the nonlocal integral centered at the point $x_c = .5$.

Initially, we will look at the case where the horizon of the nonlocal integral is decreasing. Results of the RBF quadrature method with a shape parameter $\varepsilon = 1.0$ using two and four uniform quadrature points on a decreasing horizon for the case where $s = .25$ are shown in Table 3.11 and Table 3.12 respectfully, and for the case where $s = .5$ in Table 3.13 and Table 3.14. From the results of Table 3.11 and Table 3.13, we observe that convergence rates of the error are second order for both the $s = 0.25$ and $s = 0.5$ cases when two uniform quadrature points are used. These results are similar to when two quadrature points are used in the $s = 0.0$ case. From the results of Table 3.12 and Table 3.14, we see that in the four quadrature point case that convergence rates are near fourth order initially but drop off as the horizon decreases with δ. The convergence rates begin to degrade at a higher value of δ when s is larger. We suspect that this drop off happens due to the inaccuracy of solving (3.3) for the quadrature weights of the RBF-generated quadrature rule. The problem being that we cannot approximate the right-hand side integral terms accurately enough as δ becomes too small and as s increases. Additionally, as δ gets smaller (3.3) becomes more ill-conditioned because the quadrature points are closer together. If we raise the shape parameter to $\varepsilon = 10.0$ for the $N = 4$ uniform quadrature point cases for both $s = .25$ and $s = .5$ as is done in Table 3.15 and Table 3.16, we find that the convergence rates do not drop off with a larger shape parameter. However, as expected we do see significantly larger errors as the shape parameter ε is increased. We can conclude from these results that a combination of the condition of the system (3.3) and inaccuracies calculating the right-hand side integral terms in (3.3) cause the convergence rates to degrade in the $\varepsilon = 1.0$ test cases.

Table 3.11: RBF quadrature as δ decreases using $N = 2$ uniform quadrature points for $s = .25$ and $u(x) = x^4 - x^2$. Only one quadrature weight is shown as the weights are symmetric due to points being uniformly spaced.

δ	Quad. Weight	Relative Error	Conv. Rate
.25	1.69248e-01	8.43272e-02	-
.125	8.36939e-01	2.20242e-02	1.94
.0625	4.17128e-02	5.56199e-03	1.90
.03125	2.08391e-02	1.39396e-03	2.00

Table 3.12: RBF quadrature as δ decreases using $N = 4$ uniform quadrature points for $s = .25$ and $u(x) = x^4 - x^2$. Only two quadrature weights are shown as the weights are symmetric due to points being uniformly spaced.

δ	Quad. Weights		Relative Error	Conv. Rate
.25	3.97948e-02	1.79426e-01	6.72912e-05	-
.125	1.98560e-02	8.97705e-02	4.63939e-06	3.86
.0625	9.92255e-03	4.48930e-02	2.32646e-07	4.32
.03125	4.96007e-03	2.24478e-02	1.76668e-06	-2.93

Table 3.13: RBF quadrature as δ decreases using $N = 2$ uniform quadrature points for $s = .5$ and $u(x) = x^4 - x^2$. Only one quadrature weight is shown as the weights are symmetric due to points being uniformly spaced.

δ	Quad. Weight	Relative Error	Conv. Rate
.25	0.25452	9.95311e-02	-
.125	0.12563	2.57724e-02	1.94
.0625	6.25808e-02	6.49496e-03	1.98
.03125	3.12602e-02	1.62414e-03	2.00

Table 3.14: RBF quadrature as δ decreases using $N = 4$ uniform quadrature points for $s = .5$ and $u(x) = x^4 - x^2$. Only two quadrature weights are shown as the weights are symmetric due to points being uniformly spaced.

δ	Quad. Weights		Relative Error	Conv. Rate
.25	2.73110e-02	2.2268e-01	1.89143e-04	-
.125	1.38308e-02	1.1116e-01	1.23582e-05	3.94
.0625	6.93796e-03	5.55622e-02	9.77902e-06	0.34
.03125	3.46501e-02	2.77907e-02	1.60529e-04	-4.04

We also look at how the error changes for the $s = 0.25$ and $s = .5$ cases as the number of uniform quadrature points is increased for the nonlocal integral with a fixed horizon. Numerical

Table 3.15: RBF quadrature with $\varepsilon = 10.0$ as δ decreases using $N = 4$ uniform quadrature points for $s = .25$ and $u(x) = x^4 - x^2$. Only two quadrature weights are shown as the weights are symmetric due to points being uniformly spaced.

δ	Quad. Weights		Relative Error	Conv. Rate
.25	3.94071e-02	1.88545e-01	3.54547e-02	-
.125	1.98937e-02	8.94341e-02	2.37450e-03	3.90
.0625	1.00413e-02	4.47341e-02	1.71046e-04	3.80
.03125	4.98140e-03	2.24185e-02	1.85694e-05	3.20

Table 3.16: RBF quadrature with $\varepsilon = 10.0$ as δ decreases using $N = 4$ uniform quadrature points for $s = .5$ and $u(x) = x^4 - x^2$. Only two quadrature weights are shown as the weights are symmetric due to points being uniformly spaced.

δ	Quad. Weights		Relative Error	Conv. Rate
.25	-2.60216e-02	2.67930e-01	5.14123e-02	-
.125	7.05843e-03	1.153526e-01	2.19257e-02	1.23
.0625	6.21494e-03	5.62386e-02	8.10027e-04	4.76
.03125	3.38076e-03	2.78676e-02	5.81402e-05	3.80

results for approximating the nonlocal integral fixed with $\delta = .125$ and increasing number of uniform quadrature points using two different shape parameters $\varepsilon = 2.0$ and $\varepsilon = 10.0$ are shown for the $s = .25$ case in Table 3.17 and for the $s = .5$ case in Table 3.18. From these results, we see that for the case where $\varepsilon = 2.0$ that convergence rates are high initially but degrade as the system for finding quadrature weights becomes ill-conditioned. Compared to the $s = 0$ case, we see a drop off in convergence when fewer quadrature points are used even though the condition numbers are at a point that we would not expect this. Presumably this is due to the right-hand side of the system not being approximated closely enough to the exact value. The drop off in convergence is worse in the $s = .5$ case due to the right-hand side terms being more difficult to approximate accurately than the $s = .25$ case. Setting $\varepsilon = 10.0$, as expected we see that errors initially are higher and convergence rates are lower than that of the $\varepsilon = 2.0$ case. However, increasing the shape parameter gives a better conditioned system for finding weights which allows for better results to be found when more quadrature points are used.

Table 3.17: RBF quadrature on a fixed horizon $\delta = .125$ with increasing number of uniform quadrature points for $s = .25$ and $u(x) = x^4 - x^2$. Condition denotes the condition number of the linear system used for finding quadrature weights.

	$\varepsilon = 2.0$			$\varepsilon = 10.0$		
N	Relative Error	Conv. Rate	Condition	Relative Error	Conv. Rate	Condition #
2	3.33753e-02	-	2.2e+00	7.65087e-01	-	1.7e+00
4	2.33677e-05	10.48	4.2e+03	2.37423e-03	6.15	1.3e+01
8	2.94926e-06	2.99	7.4e+12	4.80517e-05	6.97	3.0e+04
16	1.57768e-05	-2.42	2.3e+16	1.49256e-07	9.57	1.1e+15

Table 3.18: RBF quadrature on a fixed horizon $\delta = .125$ with increasing number of uniform quadrature points for $s = .5$ and $u(x) = x^4 - x^2$. Condition denotes the condition number of the linear system used for finding quadrature weights.

	$\varepsilon = 2.0$			$\varepsilon = 10.0$		
N	Relative Error	Conv. Rate	Condition	Relative Error	Conv. Rate	Condition #
2	3.90758e-02	-	2.2e+00	9.14086e-01	-	1.7e+00
4	6.4567e-05	9.24126	4.4e+03	2.19257e-02	5.38	1.6e+01
8	1.26176e-04	-0.966569	9.4e+12	4.48552e-06	12.25	4.4e+04
16	7.73787e-04	-2.6165	4.8e+16	8.79239e-06	-0.97	1.6e+15

3.2.5 Results Using Different RBFs

So far the only RBF that has been considered for the RBF-generated quadrature method is the Gaussian RBF. We use the Gaussian RBF because it is positive definite; moreover very little difference is found between the results of RBF quadrature using Gaussians and other RBFs in Table 2.1. To show this we will give numerical results from a test case using different RBFs and compare them to results using the Gaussian RBF. The test problem we will use to do these comparisons is the nonlocal integral with the test function $u(x) = x^4 - x^2$ centered at the point $x_c = .5$. Results with two uniform quadrature points where the horizon is decreasing are shown for the case of the multiquadric, inverse quadratic, and Gaussian RBFs with $\varepsilon = 1.0$ and $s = 0.0$ in Table 3.19. Similarly Table 3.20 shows results using four uniform quadrature points with a decreasing horizon for multiquadric, inverse quadratic and Gaussian RBFs with $\varepsilon = 1.0$ and $s = 0.5$. From these tables we see that results are not exactly the same, however very similar error and convergence patterns emerge no matter what RBF is used. Due to these similarities in these results we will continue to use only the Gaussian RBF to produce numerical results.

Table 3.19: RBF quadrature with three different RBFs as δ decreases using $N = 2$ uniform quadrature points for $s = 0$ and $u(x) = x^4 - x^2$.

	Gaussian		Multiquadric		Inverse Quadratic	
δ	Relative Error	Conv. Rate	Relative Error	Conv. Rate	Relative Error	Conv. Rate
.25	7.31508e-02	-	6.42963e-02	-	7.38219e-02	-
.0125	1.92276e-02	1.93	1.71777e-02	1.90	2.2183e-02	1.73
.0625	4.86392e-03	1.98	4.36909e-03	1.98	5.78066e-03	1.94
.03125	1.21952e-03	2.0	1.09704e-03	1.99	1.45992e-03	1.98

Table 3.20: RBF quadrature with three different RBFs as δ decreases using $N = 4$ uniform quadrature points for $s = .5$ and $u(x) = x^4 - x^2$.

	Gaussian		Multiquadric		Inverse Quadratic	
δ	Relative Error	Conv. Rate	Relative Error	Conv. Rate	Relative Error	Conv. Rate
.25	1.89143e-04	-	3.1454e-04	-	6.83432e-04	-
.0125	1.23582e-05	3.94	2.23807e-05	3.81	4.93997e-05	3.79
.0625	9.77902e-06	0.34	1.86696e-05	0.26	1.48928e-06	5.05
.03125	1.60529e-04	-4.04	4.64898e-05	-1.32	5.02642e-05	-5.08

3.3 Approximating a 1-D Nonlocal Diffusion Problem Using RBF-generated Quadrature Rules

Now that RBF quadrature methods have been shown to do a satisfactory job approximating nonlocal integrals, a solution to a complete nonlocal diffusion problem is now sought. We consider the one dimensional problem

$$\begin{cases} \dfrac{\partial u(x,t)}{\partial t} + \dfrac{2(1-s)}{\delta^{2-2s}} \displaystyle\int_{x-\delta}^{x+\delta} \frac{u(x,t) - u(z,t)}{|x-z|^{1+2s}} dz = f(x,t), \ \forall (x,t) \in \Omega \times [0,T) \\ \\ u(x,t) = g(x,t), \ \forall (x,t) \in \Gamma \times [0,T) \\ \\ u(x,0) = u_0(x), \ \forall x \in \Omega \end{cases} \tag{3.9}$$

where $\Omega = (\alpha, \beta) \subset \mathbb{R}$ is a bounded domain where the solution u is sought, $\Gamma = [\alpha - \delta, \alpha] \cup [\beta, \beta + \delta]$ is the domain where a volume constraint $g(x,t)$ is applied, u_0 is a given initial condition, T is a given time, $f(x,t)$ is a given source term, $\delta > 0$ is the given horizon, and $s \in [0,1)$. In order to fully detail how the nonlocal problem will be solved, the domain $\Omega' = [\alpha - \delta, \beta + \delta]$ is also defined. The spatial domain then can be visualized in Figure 3.2.

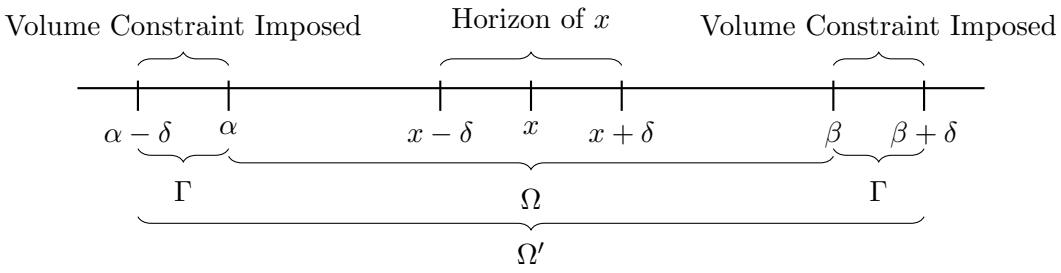


Figure 3.2: The spatial domains defined for the 1-D nonlocal diffusion problem.

The nonlocal diffusion problem in (3.9) relates to the standard heat equation in that it takes on a similar form except for the spatial operator. For the nonlocal diffusion problem, the Laplacian spatial operator $-\Delta u(x,t)$ from the standard heat equation is replaced by the nonlocal integral

$$\frac{2(1-s)}{\delta^{2-2s}} \int_{x-\delta}^{x+\delta} \frac{u(x,t) - u(z,t)}{|x-z|^{1+2s}} dz \tag{3.10}$$

which is a representation of the fractional Laplacian operator $-(\Delta)^s u(x,t)$ as $\delta \to \infty$ for $s > 0$ [8]. For details on this nonlocal integral formulation for the fractional Laplacian refer to [10]. Due to the use of this nonlocal integral operator, the handling of the boundary conditions for the nonlocal diffusion problem needs to change from those one might see for standard local methods on a bounded domain.

An important feature of the nonlocal problem (3.9) that must be discussed further is the volume constraint $g(x,t)$. For nonlocal problems, the volume constraint acts as a natural extension of boundary conditions for differential equations on bounded domains [10]. Like boundary conditions for a differential equation on a bounded domain, these volume constraints need to be defined so that the problem is well posed. Unlike boundary conditions which are values only known at the boundary, volume constraints are present over some nonzero volume outside of the boundary. The region where this volume constraint is present for the 1-D nonlocal diffusion problem is denoted as Γ in Figure 3.2. Volume constraints are necessary for the cases of $x \in \Omega$ where the horizon $[x-\delta, x+\delta]$ includes points outside of Ω.

We begin by looking at how the nonlocal integral term is going to be approximated using RBF-generated quadrature stencils. The first step is to discretize the domain Ω' into a finite number K points $\{x_1, \ldots, x_K\}$. For each point $x_i \in \Omega$, quadrature points $\{x_k\} \in [x_i - \delta, x_i + \delta] \backslash \{x_i\} \cap \{x_1, x_2, \ldots, x_K\}$ are found where k corresponds with its position in $\{x_1, \ldots, x_K\}$. Using these

quadrature points, the RBF-generated quadrature method from §3.1 is used to find quadrature weights $c_{i,k}$ so that each nonlocal integral at the point x_i can be represented by the following approximation

$$\int_{x_i-\delta}^{x_i+\delta} \frac{u(x_i,t)-u(z,t)}{|x_i-z|^{1+2s}}dz \approx \sum_k c_{i,k}\frac{u(x_i,t)-u(x_k,t)}{|x_i-x_k|^{1+2s}}. \tag{3.11}$$

We assume the quadrature weights $c_{i,k}$ are found in a preprocessing step as described in §3.1.

In order to approximate the time derivative term $\frac{\partial u(x,t)}{\partial t}$ discretely, the well-known BDFs (Backward Difference Formulas) are used [1]. In order to achieve the higher order convergence rates offered by the RBF quadrature method high order BDF rules for the time derivative must be used in practice. However for illustration purposes, the first order BDF, or the backward Euler method, $\frac{\partial u(x,t_n)}{\partial t} \approx \frac{u(x,t_n)-u(x,t_{n-1})}{\Delta t}$, where $\Delta t = t_n - t_{n-1}$, is used to approximate the time derivative at time t_n. It should be easily seen how higher order methods can be used in our scheme.

Now discretizing in time as follows $t_0 = 0, t_1 = t_0 + \Delta t, \ldots, t_M = t_{M-1} + \Delta t = T$ where Δt is a uniform time step, we have the equation

$$\frac{\partial u(x_i,t_n)}{\partial t} + \frac{2(1-s)}{\delta^{2-2s}}\int_{x_i-\delta}^{x_i+\delta}\frac{u(x_i,t_n)-u(z,t_n)}{|x_i-z|^{1+2s}}dz = f(x_i,t_n) \tag{3.12}$$

for all points $x_i \in \Omega$ at a time step t_n. For simplicity in exposition consider the case where we use x_{i-1} and x_{i+1} for quadrature points at the point x_i. At a time step t_n, we can replace (3.12) with approximations to form the following equation for all points $x_i \in \Omega$

$$U_i^n + \Delta t\frac{2(1-s)}{\delta^{2-2s}}\left[c_{i,i-1}\frac{U_i^n - U_{i-1}^n}{|x_i-x_{i-1}|^{1+2s}} + c_{i,i+1}\frac{U_i^n - U_{i+1}^n}{|x_i-x_{i+1}|^{1+2s}}\right] = \Delta t f(x_i,t_n) + U_i^{n-1} \tag{3.13}$$

where $U_i^n \approx u(x_i,t_n)$. In the case when $x_k \in \Gamma$ for $k = i-1$ or $k = i+1$, U_k^n can be replaced by the volume constraint $g(x_k,t_n)$ and this term can be moved to the right-hand side of the equation. Moving this term to the right-hand side of the equation allows for a linear system to be set up and solved for the unknowns $U_i^n,\ \forall x_i \in \Omega$. Solving linear systems recursively from $t_1 = \Delta t, \ldots, t_M = T$ using the uniform time step Δt results in the discrete approximation $U_i^M,\ \forall x_i \in \Omega$ at the time step $T = M\Delta t$. Here we have used a uniform time step for simplicity of exposition but a nonuniform time step causes no difficulties.

If a uniform time step is used, the matrix that is being used to solve the system is the same at each time step. Using some form of matrix factorization such as an LU decomposition would

result in an efficient way to solve the system at each time step. The structure of the matrix in the system that is being solved is dependent on the spatial discretization and the horizon defined by δ for the nonlocal integral. For example, setting up the system at each time step with (3.13) which uses a two point RBF quadrature rule results in a matrix A with the terms

$$\begin{cases} A_{i,i} = 1 + \Delta t C_s \left(\dfrac{c_{i,i-1}}{|x_i - x_{i-1}|^{1+2s}} + \dfrac{c_{i,i+1}}{|x_i - x_{i+1}|^{1+2s}} \right) \\ A_{i,i-1} = -\Delta t C_s \dfrac{c_{i,i-1}}{|x_i - x_{i-1}|^{1+2s}}, \ i > 1 \\ A_{i,i+1} = -\Delta t C_s \dfrac{c_{i,i+1}}{|x_i - x_{i+1}|^{1+2s}}, \ i < K \\ A_{i,j} = 0, \ \text{otherwise} \end{cases} \tag{3.14}$$

where $C_s = \frac{2(1-s)}{\delta^{2-2s}}$. Therefore, the matrix that results from this scheme with two quadrature points is tridiagonal. In general, the resultant matrix for this problem will be banded assuming proper ordering of the points, however it will not always be structured like the tridiagonal case. The nonuniform grid case is one situation where the matrix will not be as structured because the same number of quadrature points are not guaranteed to be in each point's horizon.

3.4 Numerical Results for a 1-D Nonlocal Diffusion Problem

To demonstrate how well our RBF quadrature approach from §3.3 approximates the solution to a nonlocal model, a simple test problem is considered. For the nonlocal diffusion problem given in (3.9), the spatial domain is set with $\Omega = (0, 1)$ and the exact solution where $u(x, t) = x^2(1-x^2)\sin t$ is considered. The right-hand side $f(x, t)$ is then given by

$$f(x, t) = x^2(1 - x^2)\cos t + \sin t(-2 + \delta^2 + 12x^2) \tag{3.15}$$

which is determined by using the technique involving Taylor series introduced in §3.2 to find the exact form of the nonlocal integral and finding the time derivative of $u(x, t)$ exactly. In addition, we define homogenous initial conditions at $t = 0$ and the volume constraint $g(x, t) = x^2(1 - x^2)\sin t, \ \forall x \in \Gamma$. The solution is then sought at the time $T = .1$.

For this example, the spatial domain $\Omega' = [0 - \delta, 1 + \delta]$ is discretized uniformly with spacing h. The time step is chosen to be $\Delta t = h^2$ and a fourth order BDF is used for time stepping.

When the horizon is fixed and h is reduced there are more discretization points in the integration interval $[x_i - \delta, x_i + \delta]$ so the obvious approach is to use all of these points as quadrature points to approximate the nonlocal integral because it does not increase the size of the linear system. Another option is to fix the rule at say a two-point rule which only includes the endpoints of the horizon; however this results in a second order approximation. Therefore, we use the approach where all points within the integration interval are used as quadrature points for the RBF quadrature rules.

Table 3.21 gives results for $s = 0$ as h is decreased and the horizon is fixed with $\delta = .125$ for both the RBF Quadrature approximation using Gaussian RBFs with shape parameter $\varepsilon = 2.0$ and a standard Finite Element approximation using piecewise linear elements. Similarly, Table 3.22 and Table 3.23 give these results for the $s = 0.25$ and $s = .5$ cases. Numerical results from the RBF quadrature scheme show that very high convergence rates are obtained as h decreases due to more quadrature points being used. Again, we see that the convergence rates for the error tend to degrade with more quadrature points as s increases due to inaccuracies when solving for the RBF-generated quadrature weights. Later in this section, more results will be shown with the RBF scheme to show how changing the shape parameter alters the error and convergence rates. In comparison with the Finite Element approximation with piecewise linear elements, the RBF quadrature method using $\varepsilon = 2.0$ provides a much better approximation. However, in order to make a fair comparison between these two methods we must consider the amount of computation necessary for these problems.

Table 3.21: RBF quadrature and Finite Element approximation to the nonlocal diffusion problem for the case where $s = 0$ and $\delta = .125$.

	RBF Approximation			FEM Approx. PW Linear	
h	# Quad. Pts.	Relative Error	Conv. Rate	Relative Error	Conv. Rate
.125	2	6.71403e-03	-	3.30153e-02	-
.0625	4	2.09577e-05	8.32	8.10212 e-03	2.01
.03125	8	4.95983e-09	12.04	2.00853 e-03	2.01

As h decreases in Table 3.21-3.23, the linear systems that needs to be solved at each time step increases in size for both methods. Both methods have the same sized linear system for any value of h. In addition to increased size of the linear system, as h is decreased more quadrature points are added to the RBF quadrature rule for each point. Due to the increased number of quadrature points the system derived in §3.3 for the RBF quadrature scheme has a larger bandwidth. On the

Table 3.22: RBF quadrature and Finite Element approximation to the nonlocal diffusion problem for the case where $s = .25$ and $\delta = .125$.

	RBF Approximation			FEM Approx. PW Linear	
h	# Quad. Pts.	Relative Error	Conv. Rate	Relative Error	Conv. Rate
.125	2	7.67885e-03	-	3.31014e-02	
.0625	4	1.11043e-05	9.43	8.16408e-03	2.02
.03125	8	5.57195e-06	1.00	2.03525e-03	2.00

Table 3.23: RBF quadrature and Finite Element approximation to the nonlocal diffusion problem for the case where $s = .5$ and $\delta = .125$.

	RBF Approximation			FEM Approx. PW Linear	
h	# Quad. Pts.	Relative Error	Conv. Rate	Relative Error	Conv. Rate
.125	2	8.95566e-03	-	3.27850e-03	
.0625	4	1.73349e-05	9.01	8.11285e-03	2.01
.03125	8	1.89500e-06	3.19	2.02748e-03	2.00

other hand, the system that is solved for the finite element approximation with piecewise linear elements stays fixed with a bandwidth of three as h decreases. The Finite Element problem should then takes less computational effort at each time step. However, this is not necessarily true since we have not considered the amount of calculation it takes to formulate the system for finding the Finite Element approximation. In [27], a detailed Finite Element formulation for the nonlocal problem and numerical results for solving the nonlocal problem with the Finite Element method are presented. The matrix assembly of the Finite Element approach requires evaluation of a double integral even in this one-dimensional case as compared with the RBF entries which are found by evaluating (3.3) to find quadrature weights for the entries such as those in (3.14). This double integral from the Finite Element approach was found to be difficult to approximate accurately as the integrand is singular. In [27], it was found that expensive quadrature routines are often necessary in order to get the expected error rates using the Finite Element method on nonlocal integrals. If a uniform time step is used, then the matrix assembly for the Finite Element approach only has to be done once but when adaptive time stepping is used the matrix must be assembled at each time step resulting in a significant time saving for the RBF approach. Another factor that needs to be consider is that the right-hand side must be assembled at each time step for both methods. For the RBF approach this requires simply evaluating the forcing function at a given point whereas the Finite Element

approach requires evaluating an integral of the forcing function.

An important thing to note about the numerical results from Tables 3.21-3.23 is that the error does not continue to decrease as h decreases past a certain value. Once h becomes too small the system for finding the RBF-generated quadrature weights becomes ill-conditioned, so more accurate weights cannot be found without more numerical precision. One way we can sometimes bypass this ill-conditioning issue is to change the shape parameter ε to a higher value. Altering the shape parameter allows for a finer discretization to be used that can result in higher accuracy but has the adverse effect of decreasing the accuracy when fewer points are used. Table 3.24 gives results for the RBF quadrature method on the nonlocal problem where $s = .5$ and two different shape parameters are used. These results show that with our original shape parameter of $\varepsilon = 2.0$ the condition number of the matrix that solves for the RBF quadrature weights is larger than 10^{16} causing the quadrature weights to lose accuracy since we are only using double precision arithmetic. Increasing the shape parameter to $\varepsilon = 15.0$ alleviates this ill-conditioning problem and can result in lower error than the $\varepsilon = 2.0$ case.

Table 3.24: RBF quadrature approximation to the nonlocal problem for the case where $s = .5$ with shape parameter $\varepsilon = 2.0$ and $\varepsilon = 15.0$.

		$\varepsilon = 2.0$			$\varepsilon = 15.0$		
# Quad. Pts.	h	Relative Error	Conv. Rate	Condition #	Relative Error	Conv. Rate	Condition #
2	.125	8.95566e-03	-	2.2e+00	1.28289e-01	-	1.1e+00
4	.0625	1.73349e-05	9.01	4.4e+03	6.47300e-02	.99	5.8e+00
8	.03125	1.89500e-06	3.19	9.4e+12	1.28327e-04	8.97	8.8e+02
16	0.015625	1.84470e-06	0.04	4.8e+16	1.11718e-07	10.16	2.8e+10

In practice, the horizon length δ of the nonlocal integral is often taken as a function of the discretization variable h, e.g. $\delta = 3h$. Using a horizon length that is some integer multiple of the uniform discretization variable h results in RBF-generated quadrature rules that use a fixed number of quadrature points for any value of h. As the integer multiple increases, the horizon becomes larger increasing the number of uniform quadrature points for the RBF-generated quadrature rule. Test numerical results like these are shown for an RBF-generated quadrature rule applied to a single point where the quadrature rule is fixed and the horizon decreases in §3.2.2 and §3.2.4. We are now interested in numerically approximating the nonlocal diffusion problem using the RBF quadrature scheme where δ is a function of different integer multiples of h. These numerical results are shown for the $s = 0$ case in Table 3.25-3.27 where $\delta = h$, $\delta = 2h$ and $\delta = 3h$ respectively. Similarly the

numerical results for the $s = .5$ case are shown in Table 3.28-3.30 where $\delta = h$, $\delta = 2h$ and $\delta = 3h$. Looking at these results, we see similar errors and convergence rates between those found when testing the RBF-generated quadrature rule on a single point using uniformly spaced quadrature points in §3.2.2 and §3.2.4. These similarities make sense because we use the same quadrature rules except they are now working together to solve the nonlocal diffusion problem.

Table 3.25: RBF quadrature approximation to the nonlocal diffusion problem for the case where $s = 0$ and $\varepsilon = 5.0$ with the horizon fixed as $\delta = h$.

h	δ	Relative Error	Conv. Rate
.125	.125	3.24816e-02	-
.0625	.0625	9.20493e-03	1.82
.03125	.03125	2.85070e-03	1.69
.015625	.015625	7.42276e-04	1.94

Table 3.26: RBF quadrature approximation to the nonlocal diffusion problem for the case where $s = 0$ and $\varepsilon = 5.0$ with the horizon fixed as $\delta = 2h$.

h	δ	Relative Error	Conv. Rate
.125	.25	3.64651e-03	-
.0625	.125	3.74437e-04	3.28
.03125	.0625	3.12638e-05	3.58
.015625	.03125	2.10168e-06	3.89

Table 3.27: RBF quadrature approximation to the nonlocal diffusion problem for the case where $s = 0$ and $\varepsilon = 5.0$ with the horizon fixed as $\delta = 3h$.

h	δ	Relative Error	Conv. Rate
.125	.375	1.35046e-02	-
.0625	0.1875	1.23291e-04	6.78
.03125	.09375	3.24365e-06	5.25
.015625	.046875	5.67388e-08	5.84

Table 3.28: RBF quadrature approximation to the nonlocal diffusion problem for the case where $s = .5$ and $\varepsilon = 5.0$ with the horizon fixed as $\delta = h$.

h	δ	Relative Error	Conv. Rate
.125	.125	4.59456e-02	-
.0625	.0625	1.22736e-02	1.90
.03125	.03125	3.80113e-03	1.69
.015625	.015625	9.89715e-04	1.94

Table 3.29: RBF quadrature approximation to the nonlocal diffusion problem for the case where $s = .5$ and $\varepsilon = 5.0$ with the horizon fixed as $\delta = 2h$.

h	δ	Relative Error	Conv. Rate
.125	.25	8.34788e-03	-
.0625	.125	3.84910e-04	4.44
.03125	.0625	2.67908e-05	3.84
.015625	.03125	1.70823e-06	3.97

Table 3.30: RBF quadrature approximation to the nonlocal diffusion problem for the case where $s = .5$ and $\varepsilon = 5.0$ with the horizon fixed as $\delta = 3h$.

h	δ	Relative Error	Conv. Rate
.125	.375	2.9354590469e-02	-
.0625	0.1875	1.43651e-04	7.67
.03125	.09375	3.43742e-06	5.38
.015625	.046875	6.15580e-08	5.80

CHAPTER 4

USING RBFS TO APPROXIMATE THE 2-D NONLOCAL DIFFUSION PROBLEM

Now that RBF-generated quadrature rules for nonlocal integrals in one dimension have been investigated thoroughly, we want to extend these quadrature rules to nonlocal integrals in more than one dimension. In this chapter, we will focus specifically on finding quadrature rules for the nonlocal integral in two dimensions as it will be easier to work with than higher dimensional problems. After extending the RBF-generated quadrature rules from one to two dimensions, it should be clear how quadrature rules for the nonlocal integral in more dimensions are possible. In this chapter, two different schemes for finding RBF-generated quadrature rules in two dimensions will be discussed. The first will modify the method from §3.1 with the two-dimensional nonlocal integral and the second will use a tensor product of first order rules to derive a quadrature rule for the two-dimensional nonlocal integral. Once these quadrature rules have been derived and numerically tested using a number of different quadrature point configurations, a nonlocal diffusion problem in two dimensions will be approximated using a strategy like the one used in §3.3.

This chapter begins by defining the nonlocal integral in two dimensions that we are interested in approximating. Next, the two schemes for deriving RBF-generated quadrature rules in two dimensions will be discussed and numerically tested. Finally, a nonlocal diffusion problem will be approximated using the RBF-generated quadrature rules derived in this chapter.

4.1 The 2-D Nonlocal Integral

The natural extension from the one-dimensional nonlocal integral that was used in the previous chapter to the nonlocal integral in n dimensions is

$$\int_{H_{\mathbf{x},\delta}} \frac{u(\mathbf{x}) - u(\mathbf{z})}{\|\mathbf{x} - \mathbf{z}\|^{n+2s}} d\mathbf{z} \tag{4.1}$$

where $\delta > 0$, $s \in [0, 1)$, and $H_{\mathbf{x},\delta}$ is the horizon defined by an n-dimensional ball of radius δ centered at the point $\mathbf{x} \in \mathbb{R}^n$. Typically the horizon $H_{\mathbf{x},\delta}$ is defined as

$$H_{\mathbf{x},\delta} = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{y}\|_2 \leq \delta\}.$$

In the two-dimensional case where $n = 2$, the horizon $H_{\mathbf{x},\delta}$ results in a circular shaped region. Defining the vectors $\mathbf{x}$ and $\mathbf{z}$ as $\mathbf{x} = (x, y)$ and $\mathbf{z} = (x', y')$ for the two-dimensional case, the nonlocal integral in (4.1) can be reformulated as

$$\int_{H_{\mathbf{x},\delta}} \frac{u(\mathbf{x}) - u(\mathbf{z})}{\|\mathbf{x} - \mathbf{z}\|^{2+2s}} d\mathbf{z} = \int_{x-\delta}^{x+\delta} \int_{y-\sqrt{\delta^2-(x'-x)^2}}^{y+\sqrt{\delta^2-(x'-x)^2}} \frac{u(x, y) - u(x', y')}{\sqrt{(x - x')^2 + (y - y')^2}^{(2+2s)}} dy' dx' \tag{4.2}$$

since the equation for the circle centered at (x, y) is $(x' - x)^2 + (y' - y)^2 = \delta^2$. Typically nonlocal problems in two dimensions use the circular horizon $H_{\mathbf{x},\delta}$, however we will also consider the nonlocal integral over the square horizon where the L^∞-norm is used instead of the L^2-norm in the definition of $H_{\mathbf{x},\delta}$. The nonlocal integral over the square horizon is defined as

$$\int_{\mathcal{H}_{\mathbf{x},\delta}} \frac{u(\mathbf{x}) - u(\mathbf{z})}{\|\mathbf{x} - \mathbf{z}\|^{2+2s}} d\mathbf{z} = \int_{x-\delta}^{x+\delta} \int_{y-\delta}^{y+\delta} \frac{u(x, y) - u(x', y')}{\sqrt{(x - x')^2 + (y - y')^2}^{(2+2s)}} dy' dx$$

where $\mathbf{x} = (x, y)$, $\mathbf{z} = (x', y')$, $\delta > 0$, $s \in [0, 1)$ and $\mathcal{H}_{\mathbf{x},d}$ is the square horizon. The square horizon in the two-dimensional case is defined as

$$\mathcal{H}_{\mathbf{x},\delta} = \{\mathbf{y} \in \mathbb{R}^2 : \|\mathbf{x} - \mathbf{y}\|_\infty \leq \delta\}.$$

The circular horizon $H_{\mathbf{x},\delta}$ is normally used as the nonlocal integral usually represents the interaction between a point $\mathbf{x}$ and all other points within a Euclidean distance δ. The square horizon $\mathcal{H}_{\mathbf{x},\delta}$ consists of all points within $H_{\mathbf{x},\delta}$, but also includes points outside of the circular horizon which slightly changes the nonlocal integral operator. We will not get into the details of how the nonlocal integral operator changes with a different shaped horizon. However, the reason we consider the square horizon is that it makes it much easier to numerically compute the integrals necessary for approximating the nonlocal diffusion problem with Finite Element method and the RBF-generated quadrature rules. The square horizon is used in our case mainly so that we can formulate a quadrature rule that uses a tensor product of one-dimensional RBF-generated quadrature rules. We will also use the square horizon to compare with the performance of RBF quadrature rules that use the circular horizon.

4.2 Direct RBF-generated Quadrature Rules in 2-D

The first method for finding an RBF-generated quadrature rule for the two-dimensional nonlocal integral that we will consider is one derived by mirroring the derivation of the one-dimensional RBF-generated quadrature rule in §3.1. We call this method the **direct method** as it calculates quadrature weights directly from the two-dimensional nonlocal integral formulation. This RBF quadrature rule is derived for both the circular and square horizon. A quadrature rule for the two-dimensional nonlocal integral with the circular horizon takes the form

$$\int_{H_{\mathbf{x}_c,\delta}} \frac{u(\mathbf{x}_c) - u(\mathbf{z})}{\|\mathbf{x}_c - \mathbf{z}\|^{2+2s}} d\mathbf{z} \approx \sum_{i=1}^{N} c_i \frac{u(\mathbf{x}_c) - u(\mathbf{x}_i)}{\|\mathbf{x}_c - \mathbf{x}_i\|^{2+2s}} \tag{4.3}$$

where $\mathbf{x}_c \in \mathbb{R}^2$ is the point at which the nonlocal integral is centered, $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in H_{\mathbf{x}_c,\delta} \backslash \{\mathbf{x}_c\}$ is a finite set of two-dimensional quadrature points, and $\{c_1, \ldots, c_N\}$ are quadrature weights corresponding to the given set of quadrature points. The quadrature rule for the two-dimensional nonlocal integral with the square horizon has the same form as (4.3) with $H_{\mathbf{x}_c,\delta}$ replaced by $\mathcal{H}_{\mathbf{x}_c,\delta}$. Like the one-dimensional case, we will not include the point $\mathbf{x}_c$ in the quadrature points due to the discontinuity in the integrand at that point. To determine the quadrature weights, we require (4.3) to be exact when $u(\mathbf{x})$ is replaced by RBF translates $\phi(\|\mathbf{x} - \mathbf{x}_j\|, \varepsilon),\ j = 1, \ldots, N$. This gives N linear equations that can be formed into the following matrix problem

$$\begin{pmatrix} \frac{\phi(\|\mathbf{x}_c-\mathbf{x}_1\|,\varepsilon)-\phi(\|\mathbf{x}_1-\mathbf{x}_1\|,\varepsilon)}{\|\mathbf{x}_c-\mathbf{x}_1\|^{2+2s}} & \frac{\phi(\|\mathbf{x}_c-\mathbf{x}_1\|,\varepsilon)-\phi(\|\mathbf{x}_1-\mathbf{x}_2\|,\varepsilon)}{\|\mathbf{x}_c-\mathbf{x}_2\|^{2+2s}} & \cdots & \frac{\phi(\|\mathbf{x}_c-\mathbf{x}_1\|,\varepsilon)-\phi(\|\mathbf{x}_1-\mathbf{x}_N\|,\varepsilon)}{\|\mathbf{x}_c-\mathbf{x}_N\|^{2+2s}} \\ \frac{\phi(\|\mathbf{x}_c-\mathbf{x}_2\|,\varepsilon)-\phi(\|\mathbf{x}_2-\mathbf{x}_1\|,\varepsilon)}{\|\mathbf{x}_c-\mathbf{x}_1\|^{2+2s}} & \frac{\phi(\|\mathbf{x}_c-\mathbf{x}_2\|,\varepsilon)-\phi(\|\mathbf{x}_2-\mathbf{x}_2\|,\varepsilon)}{\|\mathbf{x}_c-\mathbf{x}_2\|^{2+2s}} & \cdots & \frac{\phi(\|\mathbf{x}_c-\mathbf{x}_2\|,\varepsilon)-\phi(\|\mathbf{x}_2-\mathbf{x}_N\|,\varepsilon)}{\|\mathbf{x}_c-\mathbf{x}_N\|^{2+2s}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\phi(\|\mathbf{x}_c-\mathbf{x}_N\|,\varepsilon)-\phi(\|\mathbf{x}_N-\mathbf{x}_1\|,\varepsilon)}{\|\mathbf{x}_c-\mathbf{x}_1\|^{2+2s}} & \frac{\phi(\|\mathbf{x}_c-\mathbf{x}_N\|,\varepsilon)-\phi(\|\mathbf{x}_N-\mathbf{x}_2\|,\varepsilon)}{\|\mathbf{x}_c-\mathbf{x}_2\|^{2+2s}} & \cdots & \frac{\phi(\|\mathbf{x}_c-\mathbf{x}_N\|,\varepsilon)-\phi(\|\mathbf{x}_N-\mathbf{x}_N\|,\varepsilon)}{\|\mathbf{x}_c-\mathbf{x}_N\|^{2+2s}} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} \tag{4.4}$$

$$= \begin{pmatrix} \int_{H_{\mathbf{x}_c,\delta}} \frac{\phi(\|\mathbf{x}_c - \mathbf{x}_1\|, \varepsilon) - \phi(\|\mathbf{z} - \mathbf{x}_1\|, \varepsilon)}{\|\mathbf{x}_c - \mathbf{z}\|^{2+2s}} d\mathbf{z} \\ \int_{H_{\mathbf{x}_c,\delta}} \frac{\phi(\|\mathbf{x}_c - \mathbf{x}_2\|, \varepsilon) - \phi(\|\mathbf{z} - \mathbf{x}_2\|, \varepsilon)}{\|\mathbf{x}_c - \mathbf{z}\|^{2+2s}} d\mathbf{z} \\ \vdots \\ \int_{H_{\mathbf{x}_c,\delta}} \frac{\phi(\|\mathbf{x}_c - \mathbf{x}_N\|, \varepsilon) - \phi(\|\mathbf{z} - \mathbf{x}_N\|, \varepsilon)}{\|\mathbf{x}_c - \mathbf{z}\|^{2+2s}} d\mathbf{z} \end{pmatrix}$$

which can be solved for the quadrature weights that are used to approximate the nonlocal integral.

The system in (4.4) has a very similar form to the system (3.3) for finding RBF quadrature weights for the one-dimensional RBF quadrature rule. The differences are that the RBFs now

use the Euclidean distance in two dimensions, the exponent in the denominator of the nonlocal integrand has changed, and the right hand side terms are integrals in two dimensions. Since the problem is similar to the one-dimensional problem it runs into some of the same issues. The main issues with (4.4) are that the matrix has not been proven to always be invertible and the right hand side integrals are difficult to approximate to high accuracy as s increases. For a more complete discussion on these topics refer back to §3.1.

4.3 RBF-generated Quadrature in 2-D Using the Tensor Product

Another way that a quadrature rule for the two-dimensional nonlocal integral can be derived is by using a tensor product of one-dimensional RBF-generated quadrature rules. A rule of this kind is only considered for the nonlocal integral with the square horizon $\mathcal{H}_{\mathbf{x},\delta}$ as tensor product rules require the integration regions to be an interval in each variable. The two-dimensional nonlocal integral that we are interested in approximating has the form

$$\int_{x_c-\delta}^{x_c+\delta}\int_{y_c-\delta}^{y_c+\delta}\frac{u(x_c,y_c)-u(x',y')}{\sqrt{(x_c-x')^2+(y_c-y')^2}^{(2+2s)}}dy'dx' \tag{4.5}$$

where (x_c, y_c) is the point at which the nonlocal integral is centered.

To find a tensor product formulation to approximate (4.5), we require one-dimensional RBF-generated quadrature rules in both the x' and y' variables. Typical tensor product rules will use quadrature points and weights from a previously determined one-dimensional quadrature rule. We cannot do that in this case because the integrand of the nonlocal integral changes with the number of dimensions. Therefore, the one-dimensional quadrature weights for the tensor product rule are calculated using the method from §3.1 with the exponent in the denominator term of the integrand changed to $2 + 2s$. Assume we have N quadrature weights $\{c_1, \ldots, c_N\}$ corresponding to a set of quadrature points $\{x_1, \ldots, x_N\} \in [x_c - \delta, x_c + \delta]\backslash\{x_c\}$ in the x variable and M quadrature weights $\{d_1, \ldots, d_M\}$ corresponding to a set of quadrature points $\{y_1, \ldots, y_M\} \in [y_c - \delta, y_c + \delta]\backslash\{y_c\}$ in the variable y. We can then use these quadrature points and weights to form the tensor product approximation

$$\int_{x_c-\delta}^{x_c+\delta}\int_{y_c-\delta}^{y_c+\delta}\frac{u(x_c,y_c)-u(x',y')}{\sqrt{(x_c-x')^2+(y_c-y')^2}^{(2+2s)}}dy'dx' \approx \sum_{i=1}^{N}\sum_{j=1}^{M}c_i d_j\frac{u(x_c,y_c)-u(x_i,y_j)}{\sqrt{(x_c-x_i)^2+(y_c-y_j)^2}^{(2+2s)}}.$$

Although we cannot use previously calculated one-dimensional RBF quadrature rules, this tensor product formulation can still be very beneficial to use as it makes extending to nonlocal integrals in multiple dimensions a simple task. Since the two-dimensional horizon defined for the tensor product rule is the square horizon, the integration intervals for the x' and y' variables are the same. This means that if the quadrature points for each interval are equivalently placed then the RBF-generated quadrature rules for each will be equivalent. It is then possible to come up with a tensor product rule by calculating quadrature weights for a single RBF-generated quadrature rule in one dimension. The big limitation for the tensor product rule is that it only works on the square horizon

4.4 Numerical Results for 2-D Nonlocal Integrals

We now want to test the two-dimensional RBF quadrature methods from §4.2 and §4.3 for both the circular and square shaped horizons. Like the one-dimensional case, we are interested in observing how the approximation error behaves as $\delta \to 0$ for a fixed number of quadrature points and how it behaves as δ is fixed and the number of quadrature points is increased. Also, we are interested in comparing the approximation error between the two methods when the same quadrature points are chosen. In our tests, multiple different schemes for choosing quadrature points are considered to show how the choice of quadrature points affects approximation error.

In order to determine how well RBF-generated quadrature rules approximate a two-dimensional nonlocal integral, we first need to come up with an exact solution to the nonlocal integral. In [27], a Taylor series method is presented for finding exact solutions to the two-dimensional nonlocal integral with a circular horizon. This Taylor series method for the two-dimensional nonlocal integral is derived in a similar way to the method for the one-dimensional nonlocal integral shown in §3.2 except that it uses a Taylor series expansion in two dimensions. Using the Taylor series expansion in two dimensions makes the process of finding an exact solution much more complicated than that of the one-dimensional case and often requires a symbolic solver. Therefore instead of finding exact solutions, we find good approximations to the exact solutions by using MATLAB's *integral2* function which is an adaptive quadrature routine. Using *integral2* we are able to find approximations to nonlocal integrals that are adequate for performing error analysis of the two-dimensional RBF-generated quadrature rules.

Like the one-dimensional RBF-generated quadrature rule, we need to discuss the formation of problem (4.4) which is solved to find the quadrature weights for the direct method. The matrix in (4.4) can be found exactly since the quadrature weights satisfy $\mathbf{x}_i \neq \mathbf{x}_c,\ i = 1, \ldots, N$. The integral terms in the right-hand side of (4.4) on the other hand are difficult to find analytically for both the square and circular horizon, so we will use approximations rather than the exact values. For the square horizon, we use a two-dimensional extension of the composite Gaussian quadrature rule that was used in the one-dimensional case to approximate the right-hand side terms. For the circular horizon, a composite Gaussian cubature rule is used to approximate the right-hand side terms. These strategies for approximating the right-hand side terms were found to be adequate for most horizon lengths however we will see cases in the upcoming numerical tests where issues occur when the horizon is too small. For more details on the difficulties of approximating these right-hand terms refer back to the discussion for the one-dimensional case in §3.2.

In our tests of two-dimensional RBF-generated quadrature rules, we will consider a couple of different schemes for choosing quadrature points. For the first scheme, we will use a uniform grid of points over the given horizon. Considering the case of the square horizon centered at the point (x_c, y_c) and a given $\delta > 0$, a uniform grid of points with a spacing of $h = \frac{\delta}{N}$ where $N \in \mathbb{N}$ consists of the following set of two-dimensional points:

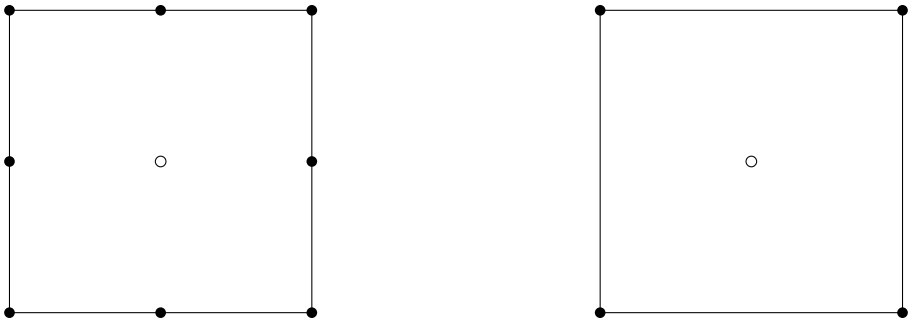
$$X = \{x_c - \delta,\ x_c - \delta + h,\ \ldots,\ x_c - \delta + (N-1)h,\ x_c,\ x_c + \delta - (N-1)h,\ \ldots, x_c + \delta - h,\ x_c + \delta\} \times$$
$$\{y_c - \delta,\ y_c - \delta + h,\ \ldots,\ y_c - \delta + (N-1)h,\ y_c,\ y_c + \delta - (N-1)h,\ \ldots, y_c + \delta - h,\ y_c + \delta\}.$$

Since we do not want the point (x_c, y_c) to be a quadrature point, the single point (x_c, y_c) is removed from X leaving the quadrature points for a uniform grid over the square horizon which we define as $X_{sq} = X \backslash (x_c, y_c)$. We can then define the set of uniform quadrature points for the circular horizon as

$$X_{circ} = \{(v, w) \in X_{sq} : \sqrt{(v - x_c)^2 + (w - y_c)^2)} \leq \delta\}.$$

Another scheme for choosing quadrature points that we consider uses subsets of X_{sq} and X_{circ} consisting of all diagonal points which results in an "X shaped" set of points over the respective horizons. These quadrature points are defined as

$$X_{sqX} = \{(v, w) \in X_{sq} : |v| = |w|\}$$



(a) Quadrature points in X_{sq} (b) Quadrature points in X_{sqX} and X_{tensor}

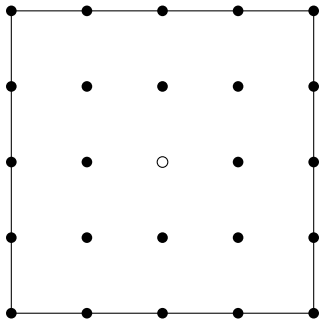
Figure 4.1: Quadrature point schemes for the square horizon where $N = 1$.

for the square horizon and

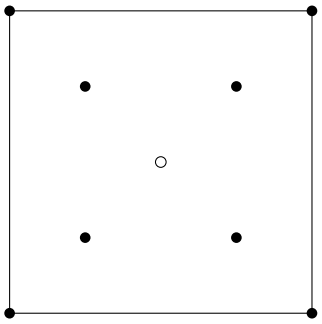
$$X_{circX} = \{(v, w) \in X_{circ} : |v| = |w|\}$$

for the circular horizon. The last scheme we consider chooses the quadrature points that result from the tensor product of uniformly spaced one-dimensional quadrature rules defined in §3.2.2. These quadrature points are only considered for the square horizon as we want to use these points to compare with the tensor product rule from §4.3. We will denote these points by X_{tensor}. We consider these types of schemes for choosing quadrature points as they allow for a systematic way to add more quadratures points to the RBF-generated quadrature rule by changing the spacing h. Figure 4.1-4.4 show examples for choosing quadrature point over the square and circular horizons that we will consider for numerical testing in the following sections.

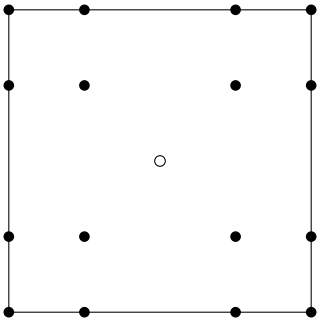
The numerical results for the direct RBF-generated quadrature rule from §4.2 and the Tensor product rule from §4.3 will now be shown in §4.4.1-§4.4.3 for both square and circular horizons using these different schemes for choosing quadrature points. To do the testing of the two-dimensional RBF quadrature rules, the nonlocal integral with $u(x, y) = x^2 y$ is considered for the case where $s = 0$. Only the $s = 0$ case will be tested as it is the only value of s that can we find a reliable approximation to the exact value that we can compare against the RBF quadrature approximation. We choose to use Gaussian RBFs with a shape parameter of $\varepsilon = 5.0$ and arbitrarily choose the center of the nonlocal integral to be at the point $(x_c, y_c) = (.5, .5)$. The shape parameter ε is set to a relatively high value in these tests in an attempt to avoid ill-conditioning effects from the system for finding quadrature weights.



(a) Quadrature points in X_{sq}

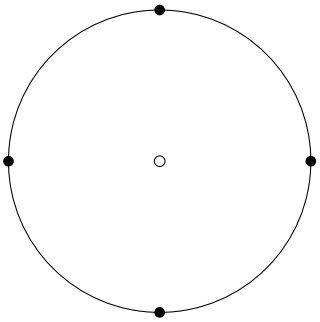


(b) Quadrature points in X_{sqX}

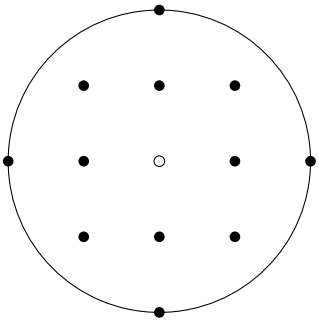


(c) Quadrature points in X_{tensor}

Figure 4.2: Quadrature point schemes for the square horizon where $N = 2$.

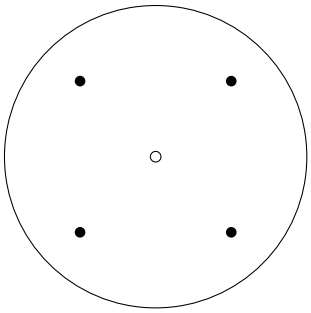


(a) Quadrature points for the $N = 1$ case

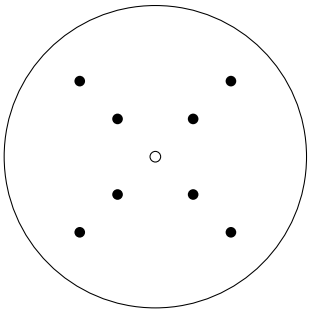


(b) Quadrature points for the $N = 2$ case

Figure 4.3: Quadrature point schemes from X_{circ}.



(a) Quadrature points for the $N = 2$ case



(b) Quadrature points for the $N = 4$ case

Figure 4.4: Quadrature point schemes from X_{circX}.

4.4.1 Numerical Results Using Direct RBF-generated Quadrature on the Circular Horizon

The first results we will look at are those that come from approximating our test nonlocal integral on a circular horizon using quadrature weights derived from the direct RBF-generated quadrature rule. For this case, we will consider the quadrature point sets X_{circ} and X_{circX}. The results as we decrease the horizon for the quadrature point set X_{circ} are shown for $N = 1$ in Table 4.1 and for $N = 2$ in Table 4.2. Likewise, the results as we decrease the horizon for the quadrature point set X_{circX} are shown for $N = 2$ in Table 4.3 and for $N = 4$ in Table 4.4. Results are shown for a fixed horizon $\delta = .125$ and quadrature points varying with N for the quadrature point set X_{circ} in Table 4.5 and for the quadrature point set X_{circX} in Table 4.6.

Table 4.1: Direct RBF quadrature on the 2-D nonlocal integral with a circular horizon as δ decreases using quadrature points from the set X_{circ} where $N = 1$ for $s = 0$ and $u(x, y) = x^2y$. The case where $N = 1$ uses 4 quadrature points; see Figure 4.3a.

δ	Relative Error	Conv. Rate
.125	6.97595e-04	-
.0625	6.77846e-05	3.36
.03125	4.57258e-06	3.89
.015625	2.88492e-07	3.99

From the results in Table 4.1 and Table 4.2, we observe that using the quadrature point set X_{circ} results in error decreasing at a fourth order rate when $N = 1$ and at a sixth order rate when $N = 2$ as δ is decreased. In this case, it makes sense that the convergence order increases with N

Table 4.2: Direct RBF quadrature on the 2-D nonlocal integral with a circular horizon as δ decreases using quadrature points from the set X_{circ} where $N = 2$ for $s = 0$ and $u(x, y) = x^2y$. The case where $N = 2$ uses 12 quadrature points; see Figure 4.3b.

δ	Relative Error	Conv. Rate
.125	5.39653e-06	-
.0625	9.69642e-08	5.80
.03125	1.56905e-09	5.95
.015625	2.42621e-09	-.63

Table 4.3: Direct RBF quadrature on the 2-D nonlocal integral with a circular horizon as δ decreases using quadrature points from the set X_{circX} where $N = 2$ for $s = 0$ and $u(x, y) = x^2y$. The case where $N = 2$ uses 4 quadrature points; see Figure 4.4a.

δ	Relative Error	Conv. Rate
.125	2.32719e-05	-
.0625	3.92500e-07	5.89
.03125	6.29232e-09	5.96
.015625	2.30253e-09	1.45

Table 4.4: Direct RBF quadrature on the 2-D nonlocal integral with a circular horizon as δ decreases using quadrature points from the set X_{circX} where $N = 4$ for $s = 0$ and $u(x, y) = x^2y$. The case where $N = 4$ uses 8 quadrature points; see Figure 4.4b.

δ	Relative Error	Conv. Rate
.125	2.53750e-05	-
.0625	3.66933e-07	6.11
.03125	5.60157e-09	6.03
.015625	2.31503e-09	1.27

because we are using more quadrature points and these points are filling the horizon better, i.e. the quadrature points decrease the fill distance. Along with the higher convergence rates, we also see that error is lower for all cases of δ using $N = 2$. From the results in Table 4.3 and Table 4.4, we see similar sixth order convergence rates and errors for the cases of $N = 2$ and $N = 4$ using the quadrature point set X_{circX} as δ is decreased. It makes sense that errors do not improve as N increases because adding more quadrature points for this scheme does not fill the horizon with quadrature points any better. The interesting thing to note about using quadrature points from the set X_{circX} is that a rule with only four quadranture points performs similarly to the rule using twelve quadrature points from the set X_{circ}. In Tables 4.2-4.4, a significant drop off in convergence

rates is seen when $\delta = \frac{1}{64}$. This drop off in convergence rates is most likely due to the inability to approximate the right-hand side terms of (4.4) accurately enough as δ becomes too small.

Table 4.5: Direct RBF quadrature on the 2-D nonlocal integral with a circular horizon fixed with $\delta = .125$ as the number of uniform quadrature points in X_{circ} increases for $s = 0$ and $u(x, y) = x^2y$.

N	Num. Quad. Pts	Relative Error	Conv. Rate
1	4	6.97595e-04	-
2	12	5.39653e-06	7.01
4	48	6.22784e-10	13.08

Table 4.6: Direct RBF quadrature on the 2-D nonlocal integral with a circular horizon fixed with $\delta = .125$ as the number of uniform quadrature points in X_{circX} increases for $s = 0$ and $u(x, y) = x^2y$.

N	Num. Quad. Pts	Relative Error	Conv. Rate
2	4	2.32719e-05	-
4	8	2.53749e-05	-.12
8	20	8.15064e-04	-5.01

From the results of Table 4.5, we observe high convergence rates as N increases for the quadrature point set X_{circ}. These convergence rates can again be attributed to the quadrature points being able to fill up the horizon as N increases. The results of Table 4.6 show that no convergence is seen for the quadrature point set X_{circX} as N increases. This is no surprise as we saw little to no improvement when increasing N from Table 4.3 to Table 4.4. We see similar behavior for quadrature points from X_{sqX} in Table 4.12 where the square horizon is fixed and the number of quadrature points is increased.

4.4.2 Numerical Results Using Direct RBF-generated Quadrature on the Square Horizon

The next results we will look at are those that come from solving our test nonlocal integral on a square horizon using quadrature weights derived from the direct RBF-generated quadrature rule. We will consider the quadrature point sets X_{sq} and X_{sqX} for this case. Table 4.7 and Table 4.8 show results as the horizon is decreased using $N = 1$ and $N = 2$ quadrature points from the set X_{sq} respectively. Likewise Table 4.9 and Table 4.10 show results for a decreasing horizon with $N = 1$ and $N = 2$ quadrature points in the set X_{sqX}. Results are shown for a fixed horizon $\delta = .125$

and quadrature points varying with N for the quadrature point set X_{sq} in Table 4.11 and for the quadrature point set X_{sqX} in Table 4.12.

Table 4.7: Direct RBF quadrature on the 2-D nonlocal integral with a square horizon as δ decreases using quadrature points from the set X_{sq} where $N = 1$ for $s = 0$ and $u(x, y) = x^2y$. The case where $N = 1$ uses 8 quadrature points; see Figure 4.1a.

δ	Relative Error	Conv. Rate
.125	6.24377e-03	-
.0625	4.45219e-05	7.13
.03125	6.14858e-07	6.18
.015625	9.33599e-09	6.04

Table 4.8: Direct RBF quadrature on the 2-D nonlocal integral with a square horizon as δ decreases using quadrature points from the set X_{sq} where $N = 2$ for $s = 0$ and $u(x, y) = x^2y$. The case where $N = 2$ uses 24 quadrature points; see Figure 4.2a.

δ	Relative Error	Conv. Rate
.125	7.77798185347e-06	-
.0625	1.56195337733e-07	5.64
.03125	2.10097017496e-09	6.22
.015625	1.98656862228e-10	3.40

Table 4.9: Direct RBF quadrature on the 2-D nonlocal integral with a square horizon as δ decreases using quadrature points from the set X_{sqX} where $N = 1$ for $s = 0$ and $u(x, y) = x^2y$. The case where $N = 1$ uses 4 quadrature points; see Figure 4.1b.

δ	Relative Error	Conv. Rate
.125	8.86326e-04	-
.0625	2.04007e-04	2.12
.03125	1.51619e-05	3.75
.015625	9.82162e-07	3.95

From the results in Table 4.7 and Table 4.8, we observe that using the quadrature point set X_{sq} results in error decreasing at a sixth order rate for both $N = 1$ and $N = 2$ cases, however we see that approximations for the $N = 2$ case has approximately two more digits of accuracy over the $N = 1$ case. The convergence rates are the same for these two cases as the fill distance decreases at a similar rate. The $N = 2$ case has better approximation errors because the fill distance is smaller overall. The results from Table 4.9 and Table 4.10 show that using the quadrature point

Table 4.10: Direct RBF quadrature on the 2-D nonlocal integral with a square horizon as δ decreases using quadrature points from the set X_{sqX} where $N = 2$ for $s = 0$ and $u(x, y) = x^2y$. The case where $N = 2$ uses 8 quadrature points; see Figure 4.2b.

δ	Relative Error	Conv. Rate
.125	3.29225e-04	-
.0625	4.22515e-06	6.28
.03125	6.28859e-08	6.07
.015625	9.71009e-10	6.02

set X_{sq} with $N = 1$ has error that decreases at a fourth order rate and with $N = 2$ has error that decreases at sixth order rates as δ decreases. Also, we find lower errors in the $N = 2$ case than the $N = 1$ case at all values of δ. Again, these results can be explained by the quadrature points being able to fill the horizon better in the $N = 2$ case. An important comparison to make between the two quadrature point schemes is that errors for the eight quadrature point case from X_{sqX} are all better than the eight quadrature point case from X_{sq}. Some other interesting comparisons that can be made are between the results in Table 4.7 and Table 4.9 and those in Table 4.8 and Table 4.10. These cases are interesting to compare as they show what happens as quadrature points are taken out of the quadrature rules. From the results in Table 4.7 and Table 4.9, we see that for the $N = 1$ case that takes out four quadrature points from the quadrature point set X_{sq} gives larger errors and lowers the convergence rate from sixth order to fourth order. From the results in Table 4.8 and Table 4.10, we observe that for the $N = 2$ case similar sixth order convergence rates are seen even though three times as many quadrature points are used for the case in Table 4.8.

Table 4.11: Direct RBF quadrature on the 2-D nonlocal integral with a square horizon fixed at $\delta = .125$ as the number of uniform quadrature points in X_{sq} increases for $s = 0$ and $u(x, y) = x^2y$.

N	Num. Quad. Pts	Relative Error	Conv. Rate
1	8	6.24377e-03	-
2	24	7.77798e-06	9.65
4	80	2.76003e-08	8.14

Like the results from increasing N for the quadrature points X_{circ} on the circular horizon with $\delta = .125$, we observe high convergence rates as N increased for the quadrature point set X_{sq} in Table 4.11. The results are similar for these two cases because they fill the horizon with quadrature points in a very similar manner. In Table 4.12, we see that little to no convergence is seen as N

Table 4.12: Direct RBF quadrature on the 2-D nonlocal integral with a square horizon fixed at $\delta = .125$ as the number of uniform quadrature points in X_{sqX} increases for $s = 0$ and $u(x, y) = x^2 y$.

N	Num. Quad. Pts	Relative Error	Conv. Rate
1	4	8.86326e-04	-
2	8	3.29225e-04	1.43
4	16	8.18240e-04	-1.31

is increased for the set of points X_{sqX}. These results are similar to the case where N is increased for the point set X_{circX} because these schemes fill the horizon with quadrature points in a similar manner.

4.4.3 Numerical Results Using Tensor Product Quadrature on the Square Horizon

The last results we will consider are those that come from solving our test nonlocal integral on a square horizon using the tensor product method to derive quadrature weights. For this case, the quadrature points considered are those that come from taking the tensor product of uniform quadrature points used in RBF quadrature rules for one-dimensional nonlocal integrals. These quadrature points are defined by the set of points from X_{tensor} for different values of N. Table 4.13 shows results where δ is decreased using quadrature points X_{tensor} where $N = 1$. Likewise Table 4.14 gives results as δ is decreased using the quadrature points X_{tensor} where $N = 2$. Table 4.15 shows results for the fixed horizon $\delta = .125$ and the number of quadrature points varying with N.

Table 4.13: Tensor Product RBF quadrature on the 2-D nonlocal integral with a square horizon as δ decreases using quadrature points from the set X_{tensor} where $N = 1$ for $s = 0$ and $u(x, y) = x^2 y$. For this case 4 quadrature points are used; see Figure 4.1b.

δ	Relative Error	Conv. Rate
.125	3.58280e-03	-
.0625	2.00959e-04	4.16
.03125	1.51857e-05	3.73
.015625	9.82755e-07	3.95

From the results in Table 4.13 and Table 4.14, we observe that the error decreases at a fourth order rate for the $N = 1$ case and at a sixth order rate for the $N = 2$ case. It is reasonable to expect an increase in the rate of convergence as N grows in these cases as the quadrature points fill

Table 4.14: Tensor Product RBF quadrature on the 2-D nonlocal integral with a square horizon as δ decreases using quadrature points from the set X_{tensor} where $N = 2$ for $s = 0$ and $u(x, y) = x^2y$. For this case 16 quadrature points are used; see Figure 4.2c.

δ	Relative Error	Conv. Rate
.125	2.32218e-05	-
.0625	4.01896e-07	5.85
.03125	6.76443e-09	5.89
.015625	7.66892e-09	-.18

Table 4.15: Tensor Product RBF quadrature on the 2-D nonlocal integral with a square horizon fixed at $\delta = .125$ as the number of uniform quadrature points in X_{tensor} increases with N for $s = 0$ and $u(x, y) = x^2y$.

N	Num. Quad. Pts	Relative Error	Conv. Rate
1	4	3.58280e-03	-
2	16	2.32218e-05	7.27
4	64	5.07883e-08	8.84

the horizon better as N gets larger. An important comparison to make here is between the results from the tensor product rule and the results using quadrature points from X_{sq} and X_{sqX} with the direct method. Comparing the results of the tensor product method and the direct method with quadrature points from X_{sq} in Table 4.7 and Table 4.8, we see very similar errors and convergence rates. However, the tensor product method uses less quadrature points and the quadrature weights are easier to find as only the quadrature weights for a single one-dimensional rule needs to be found. Comparing the results of the tensor product method and the direct method with quadrature points from X_{sqX} in Table 4.9 and Table 4.10, we again see very similar errors and convergence rates. The difference now is that the tensor product rule uses more quadrature points than the direct method with quadrature points from X_{sqX}.

From the results for the decreasing horizon, one might suspect that the direct method with quadrature points chosen from X_{sqX} should be the preferred method as it uses fewer quadrature points. However, it is not necessarily true that it is the best choice since we saw in Table 3.13 that the approximation with this rule on a fixed horizon and an increasing number of quadrature points did not improve. The results from Table 4.15 on the other hand show that the tensor product rule on a fixed horizon with an increasing number of quadrature points does provide an improvement

in the approximation similar to the direct method using quadrature points from X_{sq}. Thus, these two methods should prove to be a better option.

Table 4.16: Direct RBF quadrature on the 2-D nonlocal integral with a square horizon fixed at $\delta = .125$ as the number of uniform quadrature points in X_{tensor} increases with N for $s = 0$ and $u(x, y) = x^2 y$.

N	Num. Quad. Pts.	Relative Error	Conv. Rate
1	4	8.86326e-04	-
2	16	1.04976e-04	3.08
4	64	2.42962e-08	12.08

Table 4.17: Quadrature weights for sample quadrature points in X_{tensor} from both the Direct RBF methods and the Tensor Product method for the $N = 2$ case with $\delta = .125$. Only four of the 16 points are shown as the rule is symmetric; the points chosen are from the top right quadrant of Figure 4.2c.

	Quadrature Points / Quadrature Weights			
Method	Top Right	Bottom Right	Bottom Left	Top Left
Direct	-4.8495e-03	5.6733e-03	9.0230e-03	5.6733e-03
Tensor	1.5450e-04	1.3981e-03	1.2651e-02	1.3981e-03

One other thing we are interested in testing with the tensor product rule is how it compares to the direct RBF-generated quadrature rule when the same quadrature points used. Table 4.16 shows results for the quadrature rule where quadrature weights are found using the RBF direct method with quadrature points from X_{tensor} as N increases for a fixed horizon $\delta = .125$. The results from Table 4.16 show that similar errors are seen using quadrature weights derived from these different methods applied to the same quadrature points. However, we see in Table 4.17 that the quadrature weights derived from both methods are different. Table 4.17 shows the quadrature weights from both methods for the cases of $N = 2$ for quadrature points chosen from X_{tensor}.

4.5 Numerical Results for 2-D Nonlocal Diffusion Problems

We now want to test our two-dimensional RBF-generated quadrature rules to see how well they approximate a two-dimensional nonlocal diffusion model. The two dimension nonlocal diffusion model we will consider is attained by extending the one-dimensional model from §3.3. Since the

problem is in two dimensions the space variable is a vector such that $\mathbf{x} = (x, y)$, so we adjust the one-dimensional model accordingly to produce the two-dimensional model

$$\begin{cases} \dfrac{\partial u(\mathbf{x},t)}{\partial t} + \dfrac{1}{\delta^2}\displaystyle\int_{H_{\mathbf{x},\delta}} \frac{u(\mathbf{x},t) - u(\mathbf{z},t)}{\|\mathbf{x}-\mathbf{z}\|^{2+2s}} d\mathbf{z} = f(\mathbf{x},t),\ \forall (\mathbf{x},t) \in \Omega \times [0,T) \\ \\ u(\mathbf{x},t) = g(\mathbf{x},t),\ \forall (\mathbf{x},t) \in \Gamma \times [0,T) \\ \\ u(\mathbf{x},0) = u_0(\mathbf{x}),\ \forall \mathbf{x} \in \Omega. \end{cases} \tag{4.6}$$

In (4.6), the domains Ω and Γ are part of the rectangular domain Ω' which are defined in (4.7) where α is the left and bottom boundaries and β is the right and top boundaries of the rectangular domain.

$$\begin{aligned} \Omega &= (\alpha, \beta) \times (\alpha, \beta) \\ \Omega' &= [\alpha - \delta, \beta + \delta] \times [\alpha - \delta, \beta + \delta] \\ \Gamma &= [\alpha - \delta, \alpha] \cup [\beta, \beta + \delta] \times [\alpha - \delta, \alpha] \cup [\beta, \beta + \delta] \end{aligned} \tag{4.7}$$

The nonlocal integral in (4.6) uses the circular horizon $H_{\mathbf{x},\delta}$, however the model can easily be changed to use the square horizon $\mathcal{H}_{\mathbf{x},\delta}$.

In order to approximate the two-dimensional nonlocal diffusion model, an approach analogous to the method described in §3.3 for the one-dimensional nonlocal problem is taken. To demonstrate how well the RBF-generated quadrature approach approximates the nonlocal diffusion model, we consider two simple test problems. These test problems take the form in (4.6) with the domain $\Omega = (0, 1) \times (0, 1)$. We will consider test cases using both the square and circular horizon.

The RBF-generated quadrature rules for the nonlocal integral term for the results in this section will be calculated with the direct method using the Gaussian RBF with a shape parameter of $\varepsilon = 8.0$ and quadrature point sets X_{sq} and X_{circ}. The direct method with quadrature points from X_{sq} and X_{circ} are chosen because preliminary results from §4.4 showed that the method with these quadrature points worked well for both cases where the horizon was fixed and the number of quadrature points for the quadrature rule was increased and where the horizon was decreased with a fixed quadrature rule. One other method that could have been considered was the tensor product method for the square horizon, however results from this method should be similar to those using the direct method as they were for the numerical tests of a single quadrature rule in §4.4. We do

not consider the direct method using quadrature points from X_{sqX} and X_{circX} because error does not decrease on a fixed horizon when the number of quadrature points increases.

The first test problem we consider will use the test function $u(x, y, t) = xyt$. The forcing term $f(x, y, t)$ for the problem with the square and circular horizon is then given by

$$f(x, y, t) = xy$$

which can be found by using the Taylor series method mentioned previously to calculate the nonlocal integral term and finding the time derivative term of $u(x, y, t)$ exactly. It turns out that the nonlocal integral term drops out in this case due to the linearity of x and y. In addition, we define homogenous initial conditions at $t = 0$ and the volume constraint $g(x, y, t) = xyt$. The time step is set with $\Delta t = h^2$ and a fourth order BDF is used for time stepping to find the number approximation at $T = .1$.

Table 4.18 gives results using the circular horizon fixed with $\delta = .125$ as h is decreased for both the RBF quadrature approximation and a standard Finite Element approximation using piecewise bilinear elements. Similarly, Table 4.19 gives results using the square horizon, however no Finite Element approximations were available to do a comparison for this case. From these results, we see that the RBF approximation approximates the nonlocal diffusion problem nearly exactly up to machine precision for both the circular and square horizon whereas the Finite Element approach does not due to quadrature error when evaluating the triple integral from the nonlocal integral term. As the number of quadrature points increases, the RBF approximation becomes worse due to the problem of finding the RBF-generated quadrature weights being a more ill-conditioned problem, however the approximation is still near exact up to machine precision. The Finite Element approximation for the circular horizon does not give this nearly exact approximation and thus is an inferior approximation for this test case.

To make a fair comparison between the RBF quadrature and the Finite Element approximation, we also need to do a comparison between the computation necessary for each of the methods. This comparison for the two-dimensional problem is very similar to the one done in §3.4 so we will not go into much detail. An important point to mention is that for the two-dimensional problem the Finite Element integral approximations that are necessary require the integration over another dimension. This greatly increases the amount of computation necessary to compute the approximation especially when expensive quadrature routines are used. The RBF approximation adds very

little computation when moving to multiple dimensions. One thing that does add computation for the RBF quadrature scheme is the need to do the right-hand side integration to calculate quadrature weights in an added dimension since we are using the direct method, however these calculations would normally be done in a pre-processing step. This integration in an extra dimension would be not necessary if the tensor product method was used.

Table 4.18: RBF quadrature and Finite Element approximation to the nonlocal diffusion problem using the circular horizon fixed at $\delta = .125$ where $u(x, y, t) = xyt$ for the case where $s = 0$.

	RBF Approximation		FEM Approx. Bilinear	
h	Num. Quad. Pts.	Relative Error	Relative Error	Conv. Rate
.125	4	5.48151e-16	1.7244e-02	-
.0625	12	8.48498e-15	4.1271e-03	2.06
.03125	48	1.00969e-12	8.1020e-04	2.35

Table 4.19: RBF quadrature approximation to the nonlocal diffusion problem using the square horizon fixed at $\delta = .125$ where $u(x, y, t) = xyt$ for the case where $s = 0$.

h	Num. Quad. Pts.	Relative Error
.125	8	8.85600774509e-16
.0625	24	1.69695855735e-14
.03125	80	9.41293586772e-09

The second test problem we consider uses the slightly more complicated test function $u(x, y, t) = x^2yt$. The analytical form of the forcing term $f(x, y, t)$ for this test case is difficult to find so we use an approximation to the forcing term using the MATLAB function *integral2* instead of the exact value. We define homogenous initial conditions at $t = 0$ and the volume constraint $g(x, y, t) = x^2yt$. The numerical approximation is sought once again at $T = .1$ using the time step $\Delta t = h^2$ with a fourth order BDF. Table 4.20 and Table 4.21 gives results as h is decreased for the circular and square horizon respectively with the horizon length fixed at $\delta = .125$. Unlike the first test case, the approximation for this test case does not give an exact solution up to machine precision. So this test case gives a better idea of how the RBF quadrature approximation will act for a more general problem. From the results in Table 4.20 and Table 4.21, we observe for both the circular and square horizon that this RBF approximation to the nonlocal diffusion problem have higher errors and lower convergence rates than the RBF quadrature rule tested at a single point shown in

Table 4.5 and Table 4.11. The reason for the worse performance in the nonlocal diffusion problem is that it needs to be solved at multiple time steps which brings in more error since the forcing term being used was not exact.

Table 4.20: RBF quadrature approximation to the nonlocal diffusion problem using the circular horizon fixed at $\delta = .125$ where $u(x, y, t) = x^2yt$ for the case where $s = 0$.

h	Num. Quad. Pts.	Relative Error	Conv. Rate
.125	4	1.44354894783e-02	-
.0625	12	2.06776355825e-04	6.13
.03125	48	2.85926157204e-07	9.50

Table 4.21: RBF quadrature approximation to the nonlocal diffusion problem using the square horizon where $u(x, y, t) = x^2yt$ for the case where $s = 0$ with $\delta = .125$.

h	Num. Quad. Pts.	Relative Error	Conv. Rate
.125	8	8.45076e-02	-
.0625	24	4.48005e-05	10.88
.03125	80	5.17267e-06	3.11

Similar to the one-dimensional nonlocal diffusion problem, we also want to numerically test the RBF quadrature approximation where the horizon length δ is taken as a function of an integer multiple of the discretization variable h for both the circular and square horizon. Results for the numerical tests of the nonlocal diffusion problem using the test function $u(x, y, t) = x^2yt$ where $\delta = 2h$ are shown for the circular horizon in Table 4.22 and for the square horizon in Table 4.23. Comparing these results to their counterparts in Table 4.2 and Table 4.8 where an RBF quadrature rule using the same quadrature points are tested at a single point, we observe that errors are larger and convergence rates for the error are lower in the case where the nonlocal diffusion problem is solved. The reason for the worse performance for the nonlocal diffusion problem is that it is solved at multiple time steps with inexact forcing terms.

Table 4.22: RBF quadrature approximation to the nonlocal diffusion problem using the circular horizon with $\delta = 2h$ where $u(x, y, t) = x^2yt$ for the case where $s = 0$. For this case, 12 quadrature points are used to generated the RBF-generated quadrature rules; see Figure 4.3b.

h	δ	Relative Error	Conv. Rate
.25	.50	1.01337e-01	-
.125	.25	8.76494e-03	3.53
.0625	.125	3.74676e-04	4.55

Table 4.23: RBF quadrature approximation to the nonlocal diffusion problem using the square horizon with $\delta = 2h$ where $u(x, y, t) = x^2yt$ for the case where $s = 0$. For this case, 24 quadrature points are used to generated the RBF-generated quadrature rules; see Figure 4.2a.

h	δ	Relative Error	Conv. Rate
.25	.50	3.95808e-02	-
.125	.25	2.54031e-03	3.96
.0625	.125	6.87021e-05	5.21

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this work, a novel approach for approximating nonlocal diffusion equations using RBF-generated quadrature rules was introduced and tested numerically. Through numerical results from test cases of one and two-dimensional problems, we were able to show that this approach is viable for approximating nonlocal diffusion problems. Also, we saw that this approach has the potential of finding accurate approximations with high convergence rates as the grid for the problem is discretized more. In particular, we observed significantly better results for the RBF quadrature approach over a standard Finite Element approach when test problems were approximated on the same grid. Additionally, we found that formulating the problem for approximating the nonlocal diffusion problem in higher dimensions with the RBF quadrature approach does not bring about the computational complexities that are seen when using a Finite Element approach to approximate the same problem.

In the future, one of the main objectives will be to find a way to work around the ill-conditioned problem that often occurs when solving for the RBF-generated quadrature weights. One possibility for doing this would be to use higher precision arithmetic when solving for the quadrature weights. This would not significantly add to computation time since these quadrature weights are normally calculated during a pre-processing routine. Another objective is to find a better way to approximate right-hand side integral terms that are used within the problem for finding quadrature weights. Both of these things will allow for better results because the quadrature weights will be more accurate and more quadrature points for the rule will be able to be used. Also, we want to test the RBF quadrature approach on anomalous diffusion test cases that were not shown in this work including higher dimensional problems where $s > 0$. Other things we want to investigate further are applications where nonlocal diffusion models are used and different nonlocal models including nonlinear nonlocal diffusion problems. These topics are of interest since they are options where our RBF-generated quadrature weights can be used similar to how they were in this work.

BIBLIOGRAPHY

[1] Uri M Ascher and Linda R Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. Siam, 1998.

[2] Pouria Assari, Hojatollah Adibi, and Mehdi Dehghan. A numerical method for solving linear integral equations of the second kind on the non-rectangular domains based on the meshless method. *Applied Mathematical Modelling*, 37(22):9269–9294, 2013.

[3] Stephen D Bond, Richard B Lehoucq, and Stephen T Rowe. A galerkin radial basis function method for nonlocal diffusion. In *Meshfree Methods for Partial Differential Equations VII*, pages 1–21. Springer, 2015.

[4] Susanne Brenner and Ridgway Scott. *The mathematical theory of finite element methods*, volume 15. Springer Science & Business Media, 2007.

[5] Martin Dietrich Buhmann. Radial functions on compact support. *Proceedings of the Edinburgh Mathematical Society (Series 2)*, 41(01):33–46, 1998.

[6] Tom Cecil, Jianliang Qian, and Stanley Osher. Numerical methods for high dimensional hamilton–jacobi equations using radial basis functions. *Journal of Computational Physics*, 196(1):327–347, 2004.

[7] Xi Chen and Max Gunzburger. Continuous and discontinuous finite element methods for a peridynamics model of mechanics. *Computer Methods in Applied Mechanics and Engineering*, 200(9):1237–1250, 2011.

[8] Marta D'Elia and Max Gunzburger. The fractional laplacian operator on bounded domains as a special case of the nonlocal diffusion operator. *Computers & Mathematics with Applications*, 66(7):1245–1260, 2013.

[9] Tobin A Driscoll and Bengt Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Computers & Mathematics with Applications*, 43(3):413–422, 2002.

[10] Qiang Du, Max Gunzburger, Richard B Lehoucq, and Kun Zhou. Analysis and approximation of nonlocal diffusion problems with volume constraints. *SIAM review*, 54(4):667–696, 2012.

[11] Greg Fasshauer. Meshfree approximation with matlab. 2008.

[12] Gregory E Fasshauer. Solving differential equations with radial basis functions: multilevel methods and smoothing. *Advances in Computational Mathematics*, 11(2-3):139–159, 1999.

[13] Natasha Flyer, Grady B Wright, and Bengt Fornberg. Radial basis function-generated finite differences: A mesh-free method for computational geosciences. *Handbook of geomathematics: Springer Reference*, pages 1–30, 2014.

[14] Bengt Fornberg and Cécile Piret. A stable algorithm for flat radial basis functions on a sphere. *SIAM Journal on Scientific Computing*, 30(1):60–80, 2007.

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

[16] Richard Franke. A critical comparison of some methods for interpolation of scattered data. Technical report, DTIC Document, 1979.

[17] Rolland L Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research*, 76(8):1905–1915, 1971.

[18] Edward J Kansa. Multiquadrics?a scattered data approximation scheme with applications to computational fluid-dynamics?ii solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & mathematics with applications*, 19(8):147–161, 1990.

[19] Charles A Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive approximation*, 2(1):11–22, 1986.

[20] C Shu, H Ding, and KS Yeo. Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible navier–stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 192(7):941–954, 2003.

[21] Stewart A Silling. Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids*, 48(1):175–209, 2000.

[22] Xingping Sun. Scattered hermite interpolation using radial basis functions. *Linear algebra and its applications*, 207:135–146, 1994.

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

[24] Holger Wendland. Sobolev-type error estimates for interpolation by radial basis functions. *Surface Fitting and Multiresolution Methods*, pages 337–344, 1997.

[25] Holger Wendland. Meshless galerkin methods using radial basis functions. *Mathematics of Computation of the American Mathematical Society*, 68(228):1521–1531, 1999.

[26] Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.

[27] David Witman. *Reduced ordered modeling for a nonlocal approach to anomalous diffusion problems.* dissertation, Florida State University, 2016.

[28] Grady Barrett Wright. Radial basis function interpolation: numerical and analytical developments. 2003.

BIOGRAPHICAL SKETCH

Isaac Lyngaas received a BS in Mathematics in May of 2014 from South Dakota State University. In August of 2014, he began graduate work in the Department of Scientific Computing.