1-1-2012

A Local Radial Basis Function Method for the Numerical Solution of Partial Differential Equations

Maggie Elizabeth Chenoweth
chenoweth8@marshall.edu

A LOCAL RADIAL BASIS FUNCTION METHOD FOR THE NUMERICAL SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS

A thesis submitted to

the Graduate College of

Marshall University

In partial fulfillment of

the requirements for the degree of

Master of Arts in Mathematics

by

Maggie Elizabeth Chenoweth

Approved by

Dr. Scott Sarra, Committee Chairperson
Dr. Anna Mummert
Dr. Carl Mummert

Marshall University
May 2012

ACKNOWLEDGMENTS

I would like to begin by expressing my sincerest appreciation to my thesis advisor, Dr. Scott Sarra. His knowledge and expertise have guided me during my research endeavors and the process of writing this thesis. Dr. Sarra has also served as my teaching mentor, and I am grateful for all of his encouragement and advice. It has been an honor to work with him.

I would also like to thank the other members of my thesis committee, Dr. Anna Mummert and Dr. Carl Mummert. Their feedback and counsel have been exceedingly beneficial while finalizing my thesis. The leadership of the chair of the Mathematics Department, Dr. Alfred Akinsete, and formerly Dr. Ralph Oberste-Vorth, as well as the guidance of the graduate advisor, Dr. Bonita Lawrence, have been outstanding. I am lucky to have been a teaching assistant in a department with these individuals.

If it had it not been for numerous teachers, I would not be where I am today. Specifically, I would like to thank the professors I had the privilege of taking mathematics courses from while at Marshall: Dr. John Drost, Dr. Judith Silver, Dr. Karen Mitchell, Dr. Ariyadasa Aluthge, Dr. Evelyn Pupplo-Cody, Dr. Yulia Dementieva, Dr. Scott Sarra, Dr. Ralph Oberste-Vorth, Dr. Anna Mummert, Dr. Carl Mummert, and Dr. Bonita Lawrence. You have taught me not only mathematics, but a love for a subject that I hope to instill in the hearts and minds of future students.

Finally, I would like to thank my family and friends for all of their continued love and support. This thesis is dedicated to them.

CONTENTS

ACKNOWLEDGMENTS **iii**

ABSTRACT **viii**

1 INTRODUCTION **1**

2 RADIAL BASIS FUNCTION METHODS **3**

2.1 GLOBAL RBF INTERPOLATION 5

2.2 CATEGORIES OF RBFS 7

2.3 OTHER PROPERTIES OF RBFS 9

2.4 USING RBFS TO APPROXIMATE DERIVATIVES 12

2.4.1 GLOBAL DERIVATIVE EXAMPLE 13

3 EFFICIENT AND ACCURATE IMPLEMENTATION OF RBF METHODS FOR TIME-DEPENDENT PDES **17**

3.1 THE LOCAL METHOD 17

3.1.1 LOCAL DERIVATIVE EXAMPLE 19

3.1.2 THE LOCAL METHOD USING SINGULAR VALUE DECOMPOSITION 21

3.2 GLOBAL VERSUS LOCAL RBF APPROXIMATION 22

3.3 1D ADVECTION DIFFUSION 23

3.4 2D ADVECTION DIFFUSION REACTION 26

4 PATTERN FORMATION IN MATHEMATICAL BIOLOGY **31**

4.1 APPLICATION 1: TURING PATTERNS 31
4.2 APPLICATION 2: CHEMOTAXIS 34

5 CONCLUSION **40**

A MATLAB CODE **42**

REFERENCES **47**

CURRICULUM VITAE **49**

LIST OF FIGURES

2.1 Global RBF interpolation 7

2.2 Algebraic versus spectral convergence 8

2.3 Relationship between κ and ε for the MQ 11

2.4 Derivative calculated using RBF methods 16

3.1 Local RBF approximation of the derivative of $f(x) = e^{\sin(\pi x)}$. 20

3.2 Global RBF approximation of the derivative of $f(x) = e^{x^3} - \cos(2x)$. 22

3.3 Eigenvalue stability 25

3.4 One-dimensional linear advection-diffusion results 25

3.5 Circle with stencil 27

3.6 The initial condition of Equation (3.16). 28

3.7 Error in the numerical solution of Equation (3.16) at t=5. 29

3.8 Local method solution at t=5. 29

3.9 Local method error at t=5. 30

4.1 Turing patterns observed on a leopard and in galaxies 32

4.2 Turing patterns on a fish 33

4.3 Turing patterns on a butterfly domain. 34

4.4 Density plots for chemotaxis experimental results 38

4.5 Chemotaxis experimental results 39

LIST OF TABLES

2.1 Global, infinitely differentiable RBFs 8

3.1 Error versus stencil size from Equation (3.11). 23

3.2 Error versus stencil size from Equation (3.16). 28

4.1 Parameters for Equations (4.4) and (4.5) 36

ABSTRACT

A LOCAL RADIAL BASIS FUNCTION METHOD FOR THE NUMERICAL SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS

Maggie Elizabeth Chenoweth

Most traditional numerical methods for approximating the solutions of problems in science, engineering, and mathematics require the data to be arranged in a structured pattern and to be contained in a simply shaped region, such as a rectangle or circle. In many important applications, this severe restriction on structure cannot be met, and traditional numerical methods cannot be applied.

In the 1970s, radial basis function (RBF) methods were developed to overcome the structure requirements of existing numerical methods. RBF methods are applicable with scattered data locations. As a result, the shape of the domain may be determined by the application and not the numerical method.

Radial basis function methods can be implemented both globally and locally. Comparisons between these two techniques are made in this work to illustrate how the local method can obtain very similar accuracy to the global method while only using a small subset of available points, and thus using substantially less computer memory.

Finally, radial basis function methods are applied to solve systems of nonlinear partial differential equations (PDEs) that model pattern formation in mathematical biology. The local RBF method will be used to evaluate Turing pattern and chemotaxis models that are both modeled by advection-reaction-diffusion type PDEs.

Chapter 1

INTRODUCTION

Radial basis function (RBF) methods were first studied by Roland Hardy, an Iowa State geodesist, in 1968, when he developed one of the first effective methods for the interpolation of scattered data [8]. Polynomial methods had previously been used, but they do not have a unisolvency property for two-dimensional and higher dimensional scattered data. After much investigation, Hardy developed what would later be known as the multiquadric (MQ) radial basis function [9]. This is only one of many existing RBFs.

Then, in 1979, Richard Franke published a study of all known methods of scattered data interpolation and concluded that the MQ RBF method was the best method. Because of Franke's extensive numerical experiments with the MQ, he is often credited for introducing the MQ into the field of mathematical science [6].

The next significant event in RBF history was in 1986 when Charles Micchelli, an IBM mathematician, developed the theory behind the MQ method. He proved that the system matrix for the MQ method was invertible, which means that the RBF scattered data interpolation problem is well-posed [15]. Four years later, physicist Edward Kansa first used the MQ method to solve partial differential equations [12]. In 1992, results from Wolodymyr Madych and Stuart Nelson [14] showed the spectral convergence rate of MQ interpolation. Since Kansa's discovery, research in RBF methods has rapidly grown, and RBFs are now considered an effective way to solve partial differential equations [30]. All RBF methods using an infinitely differentiable RBF have been proven to be generalizations of the polynomial based pseudospectral methods [22].

Over the years, RBF interpolation has been shown to work in many cases where polynomial interpolation has failed [20]. RBF methods overcome the limitation of polynomial interpolation because they do not need to be implemented on tensor product grids or in rectangular domains. RBF methods are frequently used to represent topographical surfaces as well as other intricate three-dimensional shapes [23], having been successfully applied in such diverse areas as climate modeling, facial recognition, topographical map production, auto and aircraft design, ocean floor mapping, and medical imaging. RBF methods have been actively developed over the last 40 years and the RBF research area remains very active as many open questions still remain.

In this work, comparisons will be made between global and local RBF approximation methods. It will be shown how the local method can obtain very similar accuracy to that of the global method while using only a small subset of available points. Hence, less computer memory is required. This will be illustrated with several numerical examples including those that involve pattern formation obtained from Turing and chemotaxis models.

Chapter 2

RADIAL BASIS FUNCTION METHODS

In order to effectively understand RBF methods, several definitions are first required.

Definition 1. *A function* $\Phi\colon \mathbb{R}^d \to \mathbb{R}$ *is called* ***radial*** *if there exists a univariate function* $\varphi\colon [0,\infty) \to \mathbb{R}$ *such that*

$$\Phi(x) = \varphi(r),$$

where $r = \|x\|$*, and* $\|\cdot\|$ *is a norm on* $\mathbb{R}^d$*.* ($\|\cdot\|$ *is typically the Euclidean norm.*)

Definition 2. *A* ***radial basis function****,* $\phi(r)$*, is a one-variable, continuous function defined for* $r \geq 0$ *that has been radialized by composition with the Euclidean norm on* $\mathbb{R}^d$*. RBFs may have a free parameter, the shape parameter, denoted by* ε*.*

Definition 3. *The* ***scattered data interpolation problem*** *states that given data* (x_j, f_j)*, with* $j = 1, \ldots, N, x_j \in \mathbb{R}^d$*, and* $f_j \in \mathbb{R}$*, find a smooth function* s *such that* $s(x_j) = f_j$*, for* $j = 1, \ldots, N$*.*

Given a set of N centers, $x_1^c, \ldots, x_N^c$, in $\mathbb{R}^d$, a radial basis function interpolant is of the form

$$s(x) = \sum_{j=1}^{N} \alpha_j \phi(\| x - x_j^c \|_2). \tag{2.1}$$

The α_j coefficients in the RBF are determined by enforcing the interpolation condition

$$s(x_i) = f(x_i)$$

at a set of points that usually coincides with the N centers. Enforcing the interpolation condition at the N centers results in the $N \times N$ linear system

$$B\alpha = f \tag{2.2}$$

to be solved for the expansion coefficients α. The matrix B, called the interpolation matrix or the system matrix, has entries

$$b_{ij} = \phi(\|x_i^c - x_j^c\|_2), \qquad i, j = 1, \ldots, N.$$

In Section 2.3 it will be shown that the system matrix is always invertible.

For the distance matrix r (as defined in Definition 1) that is used in the calculation of the system matrix, it may seem as though two loops should be created in Matlab. This can be naively coded as a double loop:

```
N = length(xc);
for i=1:N
  for j=1:N
     r(i,j) = abs( xc(i) - xc(j) );
  end
end
```

However, when loops are used in Matlab, the program can take a long time to run, and it is not as efficient as other approaches. For instance, a summation, which in a computer language is coded as a loop, can be replaced by a more efficient dot product with a vector of ones.

In Matlab, the distance matrix is formed as follows where xc represents the vector of N distinct centers.

```
o = ones(1,length(xc));
r = abs(  xc*o - (xc*o)' );
```

The interpolant is evaluated using (2.1) at the M points, $s(x_i)$, by forming an $M \times N$ evaluation matrix H with entries

$$h_{ij} = \phi(\|x_i - x_j^c\|_2), \qquad i = 1, \ldots, M \text{ and } j = 1, \ldots, N.$$

In Matlab, the evaluation matrix is formed in a manner similar to the system matrix where xc represents the vector of N distinct centers and x is the vector of the M points at which to evaluate the RBF interpolant.

```
xc = xc(:);
 x = x(:);
 r = abs( x*ones(1,length(xc)) - ones(length(x),1)*xc' );
 H = mqRbf(r,shape);
```

The interpolant is then evaluated at the M points by the matrix-vector product

$$f_a = H\alpha.$$

2.1 GLOBAL RBF INTERPOLATION

RBF approximation methods may be either global or local. The global approach uses information from every center in the domain to approximate a function value or derivative at a single point. In contrast, the local method only uses a small subset of the available centers. In this section, the mechanics of the global interpolation method is illustrated in the following example.

For this example, the MQ RBF, as defined in Table 2.1, is used to interpolate a function. Let $f(x) = e^{\sin(\pi x)}$ be restricted to the interval [0,1]. This function is interpolated using the following three centers that are not evenly spaced: $x_1^c = 0$, $x_2^c = 0.6$, and $x_3^c = 1$. The interpolant is evaluated at the five evenly spaced evaluation points $x_1 = 0$, $x_2 = 0.25$, $x_3 = 0.5$, $x_4 = 0.75$, and $x_5 = 1$. For the MQ, a shape parameter is also required. We will let $\varepsilon = 1.25$. A discussion of how how to choose the best value of the shape parameter can be found in Section 2.3.

Let

$$\alpha = [\alpha_1 \qquad \alpha_2 \qquad \alpha_3]$$

be the unknown vector of the expansion coefficients, let

$$f = [f(x_1^c) \qquad f(x_2^c) \qquad f(x_3^c)]^T = \begin{bmatrix} 1 & e^{\sin(0.6\pi)} & 1 \end{bmatrix}^T,$$

and let

$$B = \begin{bmatrix} \phi(\|x_1^c - x_1^c\|_2) & \phi(\|x_1^c - x_2^c\|_2) & \phi(\|x_1^c - x_3^c\|_2) \\ \phi(\|x_2^c - x_1^c\|_2) & \phi(\|x_2^c - x_2^c\|_2) & \phi(\|x_2^c - x_3^c\|_2) \\ \phi(\|x_3^c - x_1^c\|_2) & \phi(\|x_3^c - x_2^c\|_2) & \phi(\|x_3^c - x_3^c\|_2) \end{bmatrix}$$

be the 3×3 system matrix. The linear system $B\alpha = f$ can now be solved resulting in the following expansion coefficients:

$$\alpha = [9.5029 \qquad -26.3975 \qquad 15.3013]^T.$$

The 5×3 evaluation matrix, H, is used to evaluate the interpolant.

$$H = \begin{bmatrix} \phi(\|x_1 - x_1^c\|_2) & \phi(\|x_1 - x_2^c\|_2) & \phi(\|x_1 - x_3^c\|_2) \\ \phi(\|x_2 - x_1^c\|_2) & \phi(\|x_2 - x_2^c\|_2) & \phi(\|x_2 - x_3^c\|_2) \\ \phi(\|x_3 - x_1^c\|_2) & \phi(\|x_3 - x_2^c\|_2) & \phi(\|x_3 - x_3^c\|_2) \\ \phi(\|x_4 - x_1^c\|_2) & \phi(\|x_4 - x_2^c\|_2) & \phi(\|x_4 - x_3^c\|_2) \\ \phi(\|x_5 - x_1^c\|_2) & \phi(\|x_5 - x_2^c\|_2) & \phi(\|x_5 - x_3^c\|_2) \end{bmatrix}$$

The next step is to calculate $f_a = H\alpha$. The interpolated values are:

$$f_a = \begin{bmatrix} 1 & 2.1167 & 2.6473 & 2.1994 & 1 \end{bmatrix}^T.$$

The exact solutions are 1, 2.0281, 2.7183, 2.0281, and 1. The point-wise errors are as follows:

0.000000000000000

0.088631801660779

0.070973556874207

0.171280064870869

0.000000000000000.

This is illustrated in Figure 2.1. The errors at the endpoints are zero, as the evaluation points coincide with centers, and the interpolation conditions dictate that the interpolant agree with the function values at the centers. This is just a basic example to demonstrate how MQ RBFs are computed, but in further examples, there are multiple strategies that can be used to minimize point-wise errors.

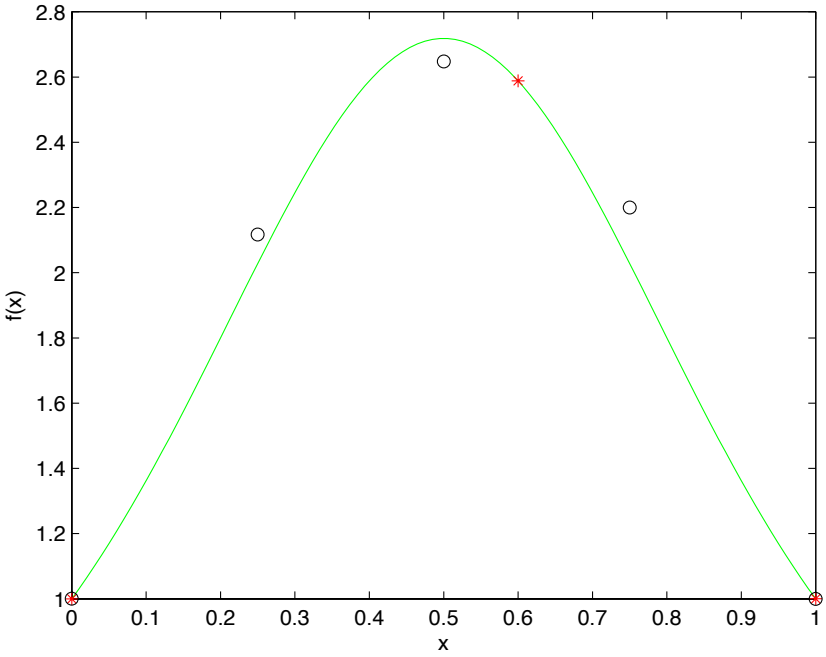

Figure 2.1: Global RBF interpolation of the function $f(x) = e^{\sin(\pi x)}$. Red asterisks denote centers and open black circles denote evaluation points.

2.2 CATEGORIES OF RBFS

Three primary categories of radial basis function methods exist: compactly supported, global with finite smoothness, and global infinitely differentiable. Compactly supported RBFs have an algebraic convergence rate.

Definition 4. ***Algebraic convergence rates*** *occur when error decays at the rate* $\mathcal{O}(N^{-m})$ *for some constant* m.

Second, we have the globally supported, finitely differentiable radial basis functions, which also have algebraic convergence rates [28]. Finally, there are the globally supported, infinitely differentiable radial basis functions. This category can achieve spectral convergence rates for smooth functions [10, 17].

Definition 5. ***Spectral convergence rates*** *occur when error decays at the rate of* $\mathcal{O}(\eta^N)$ *as* N *increases where* $0 < \eta < 1$. *In particular with RBF methods,* N *is equal to* $\frac{1}{\varepsilon h}$, *where*

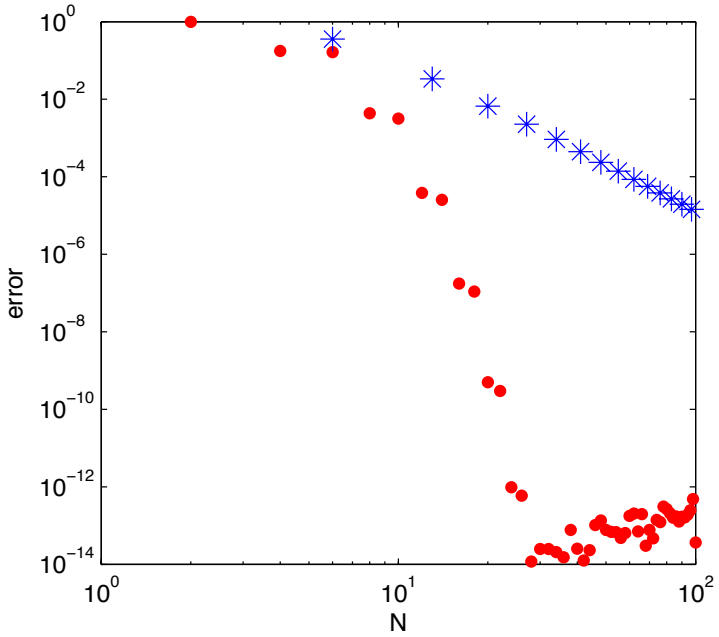

Figure 2.2: Algebraic convergence (represented by blue asterisks) versus spectral convergence (represented by red dots).

ε is the shape parameter and h is the minimum separation distance between centers. Error decays as both the shape parameter and minimum separation distance decrease.

The difference between algebraic convergence and spectral convergence is illustrated in Figure 2.2.

Four common RBFs that are globally supported and infinitely differentiable are contained in Table 2.1. The multiquadric (MQ) is arguably the most popular RBF that is used in applications and is representative of the class of global infinitely differentiable RBFs. For that reason the MQ has been used in all examples throughout.

RBF Name	Equation
Gaussian (GA)	$\phi(r,\varepsilon) = e^{-\varepsilon^2 r^2}$
Inverse Quadratic (IQ)	$\phi(r,\varepsilon) = 1/(1+\varepsilon^2 r^2)$
Inverse Multiquadric (IMQ)	$\phi(r,\varepsilon) = 1/(\sqrt{1+\varepsilon^2 r^2})$
Multiquadric (MQ)	$\phi(r,\varepsilon) = \sqrt{1+\varepsilon^2 r^2}$

Table 2.1: Global, infinitely differentiable RBFs

2.3 OTHER PROPERTIES OF RBFS

Radial basis function methods have numerous properties. Several of those properties will be highlighted in this section.

Definition 6. *A real symmetric matrix A is called* ***positive semi-definite*** *if its associated quadratic form is non-negative, i.e.,*

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k A_{jk} \geq 0 \tag{2.3}$$

for $\mathbf{c} = [c_1, ..., c_N]^T \in \mathbb{R}^N$. If the inequality is zero only for $\mathbf{c} \equiv \mathbf{0}$, then A is called ***positive definite****.*

A positive definite matrix is non-singular since all of the eigenvalues of a positive definite matrix are positive. In other words, a well-posed interpolation problem exists if the basis functions generate a positive definite system matrix. Definition 6 relates positive definite functions to positive semi-definite matrices. To make sure that the interpolation problem is well-posed, this definition can be refined to that of a *strictly* positive definite function.

Definition 7. *A complex-valued continuous function $\Phi : \mathbb{R}^d \to \mathbb{C}$ is called* ***positive definite*** *on $\mathbb{R}^d$ if*

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j \overline{c_k} \Phi(x_j - x_k) \geq 0 \tag{2.4}$$

for any N pairwise different points $x_1, ..., x_N \in \mathbb{R}^d$, and $\mathbf{c} = [c_1, ..., c_N]^T \in \mathbb{C}^N$. The function Φ is called ***strictly positive definite*** *on $\mathbb{R}^d$ if the inequality is zero only for $\mathbf{c}$.*

The inverse quadratic, inverse multiquadric, and Gaussian functions are all strictly positive definite and thus have a positive definite system matrix which is invertible.

After many experiments, Franke concluded that the system matrix was always uniquely solvable because the linear system resulting from the scattered data interpolation problem was nonsingular [6]. However, Franke never provided a proof. In 1986, Micchelli provided the conditions necessary for the proof of Theorem 1, and he generalized the idea of uniquely solvable systems to all radial basis functions [15].

In order to understand Theorem 1, the definition of a function being *completely monotone* is necessary.

Definition 8. *A function ψ is* ***completely monotone*** *on $[0, \infty)$ if*

i. $\psi \in C[0, \infty)$

ii. $\psi \in C^{\infty}(0, \infty)$

iii. $(-1)^l \psi^l(r) \geq 0$ *where* $r > 0$ *and* $l = 0, 1, ...$

Theorem 1. *Let $\psi(r) = \phi(\sqrt{r}) \in C[0, \infty)$ and $\psi(r) > 0$ for $r > 0$. Let $\psi'(r)$ be completely monotone and nonconstant on $(0, \infty)$. Then for any set of N distinct centers $\{x_j^c\}_{j=1}^N$, the $N \times N$ matrix B with entries $b_{jk} = \phi(\| x_j^k - x_k^c \|_2)$ is invertible. Such a function is said to be conditionally positive definite of order one.*

The MQ is an example of a global, infinitely differentiable RBF that is a conditionally positive definite function of order one.

To show this, let

$$\psi(r) = \phi(\sqrt{r}) = \sqrt{1 + \varepsilon^2 r} \tag{2.5}$$

and evaluate the first several derivates with respect to r:

$$\begin{aligned} \psi'(r) &= \frac{\varepsilon^2}{2\sqrt{1 + \varepsilon^2 r}} \\ \psi''(r) &= \frac{-\varepsilon^4}{4\left(1 + \varepsilon^2 r\right)^{3/2}} \\ \psi^{(3)}(r) &= \frac{3\varepsilon^6}{8\left(1 + \varepsilon^2 r\right)^{5/2}} \\ \psi^{(4)}(r) &= \frac{-15\varepsilon^8}{16\left(1 + \varepsilon^2 r\right)^{7/2}} \\ \vdots \quad &= \quad \vdots \end{aligned} \tag{2.6}$$

From this pattern, it can be seen that the sign alternates for even and odd derivatives [4]. Thus, by Definition 8 and Theorem 1, the MQ is conditionally positive definite of order one.

The MQ contains a free variable, ε, known as the shape parameter. The shape parameter affects both the conditioning of the system matrix and the accuracy of the RBF method when given a set of fixed centers. The condition number

$$\kappa(B) = \|B\|_2 \left\|B^{-1}\right\|_2 = \frac{\sigma_{\max}}{\sigma_{\min}} \tag{2.7}$$

is a measure of how difficult a problem is to be accurately approximated by a numerical algorithm. The condition number is defined using any matrix norm. In the particular case of the 2-norm, it is the ratio of the maximum to minimum singular values. The singular values of B are represented by σ. As a rule of thumb, when the condition number is 10^n, one would expect to lose approximately n accurate digits when solving a general linear system $Ax = b$.

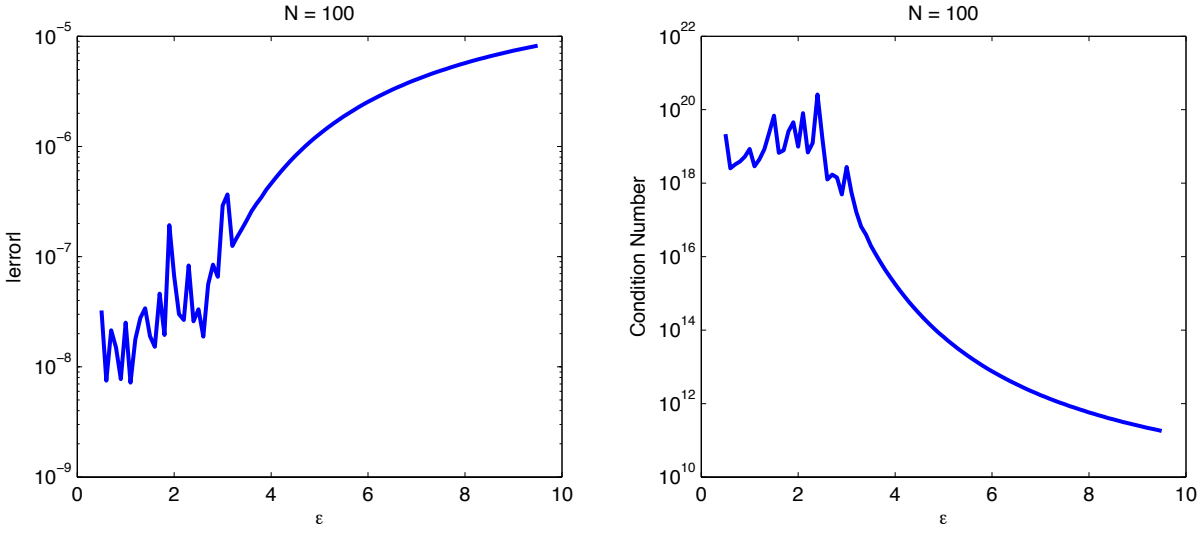

Figure 2.3: MQ RBF interpolation of the function, $f(x) = e^x$. Left: Error versus the shape parameter. Right: Condition number versus the shape parameter.

As shown is Figure 2.3, the condition number exponentially increases as the shape parameter decreases. In order for the system matrix to be well-conditioned, the shape parameter must not be too small. However, small shape parameters are required to obtain good accuracy for the RBF method, but this results in the system matrix being ill-conditioned. Hence, having the best accuracy and conditioning can obviously not occur at the same time.

This is known as the Uncertainty Principle, which indicates the more favorably valued one quantity is the less favorably valued the other is [24].

For small ε, the error improves, but the condition number grows. There are several approaches that can be taken to find the best shape parameter. These approaches include using the power function [29], "leave-one-out" cross validation [19], or the Contour-Pade Algorithm [5]. When the system matrix is ill-conditioned, RBF methods are most accurate. The shape parameter can be selected so that the resulting system matrix has a condition number, $\kappa(B)$, in the range $10^{13} \leq \kappa(B) \leq 10^{15}$ in order to determine the corresponding value for ε. These bounds for the condition number are valid when using a computer that implements double precision floating point arithmetic, but the bounds will be different when using other floating point number systems [16].

2.4 USING RBFS TO APPROXIMATE DERIVATIVES

The derivative of a function $f(x)$ is approximated by the RBF method as

$$\frac{\partial}{\partial x_i} f(x) = \sum_{j=1}^{N} \alpha_j \frac{\partial}{\partial x_i} \phi(\| x - x_j^c \|_2). \tag{2.8}$$

When evaluated at the N centers, $\left\{x_j^c\right\}_{j=1}^{N}$, (2.8) becomes

$$\frac{\partial}{\partial x_i} f(x) = \frac{\partial}{\partial x_i} H\alpha. \tag{2.9}$$

The $N \times N$ evaluation matrix, H, has entries

$$h_{i,j} = \frac{\partial}{\partial x_i} \phi(\|x_i^c - x_j^c\|_2), \qquad i, j = 1, \ldots, N. \tag{2.10}$$

From Equation (2.2), it is known that $\alpha = B^{-1} f$. The differentiation matrix can be defined as

$$D = \frac{\partial}{\partial x_i} H B^{-1}. \tag{2.11}$$

The derivative is then approximated as

$$\frac{\partial}{\partial x_i} \approx \frac{\partial}{\partial x_i} f(x) = Df. \tag{2.12}$$

(Note: In Matlab, the inverse of B is not actually formed. Instead, $D = H/B$.) Using the chain rule to differentiate the radial basis function, $\phi[r(x)]$, the result is

$$\frac{\partial \phi}{\partial x_i} = \frac{d\phi}{dr}\frac{\partial r}{\partial x_i} \tag{2.13}$$

for the first derivative, with

$$\frac{\partial r}{\partial x_i} = \frac{x_i}{r}. \tag{2.14}$$

The second derivative is calculated as follows

$$\frac{\partial^2 \phi}{\partial x_i^2} = \frac{d\phi}{dr}\frac{\partial^2 r}{\partial x_i^2} + \frac{d^2\phi}{dr^2}\left(\frac{\partial r}{\partial x_i}\right)^2, \tag{2.15}$$

with

$$\frac{\partial^2 r}{\partial x_i^2} = \frac{1 - \left[\frac{\partial r}{\partial x_i}\right]^2}{r}. \tag{2.16}$$

In particular, for the MQ,

$$\frac{d\phi}{dr} = \frac{\varepsilon^2 r}{\sqrt{1+\varepsilon^2 r^2}} \tag{2.17}$$

and

$$\frac{d^2\phi}{dr^2} = \frac{\varepsilon^2}{[1+\varepsilon^2 r^2]^{3/2}}. \tag{2.18}$$

2.4.1 GLOBAL DERIVATIVE EXAMPLE

For this example, $f'(x)$ is approximated for $f(x) = e^{\sin(\pi x)}$ on the interval [0,1] using the following three center locations $x_1^c = 0$, $x_2^c = 0.6$, and $x_3^c = 1$ to approximate the derivative. The shape parameter is taken to be $\varepsilon = 1.25$.

To begin, let

$$f = \begin{bmatrix} f\left(x_1^c\right) & f\left(x_2^c\right) & f\left(x_3^c\right) \end{bmatrix}^T = \begin{bmatrix} 1 \\ e^{\sin(0.6\pi)} \\ 1 \end{bmatrix},$$

and let

$$\begin{aligned} B &= \begin{bmatrix} \phi(\|x_1^c - x_1^c\|_2) & \phi(\|x_1^c - x_2^c\|_2) & \phi(\|x_1^c - x_3^c\|_2) \\ \phi(\|x_2^c - x_1^c\|_2) & \phi(\|x_2^c - x_2^c\|_2) & \phi(\|x_2^c - x_3^c\|_2) \\ \phi(\|x_3^c - x_1^c\|_2) & \phi(\|x_3^c - x_2^c\|_2) & \phi(\|x_3^c - x_3^c\|_2) \end{bmatrix} \\ &= \begin{bmatrix} \phi(\|0-0\|_2) & \phi(\|0-0.6\|_2) & \phi(\|0-1\|_2) \\ \phi(\|0.6-0\|_2) & \phi(\|0.6-0.6\|_2) & \phi(\|0.6-1\|_2) \\ \phi(\|1-0\|_2) & \phi(\|1-0.6\|_2) & \phi(\|1-1\|_2) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1.25 & 1.6008 \\ 1.25 & 1 & 1.1180 \\ 1.6008 & 1.1180 & 1 \end{bmatrix} \end{aligned}$$

be the 3×3 system matrix.

The 3×3 evaluation matrix, H, is formed to evaluate our interpolant:

$$\begin{aligned} \frac{\partial}{\partial x} H &= \begin{bmatrix} \phi_x(\|x_1^c - x_1^c\|_2) & \phi_x(\|x_1^c - x_2^c\|_2) & \phi_x(\|x_1^c - x_3^c\|_2) \\ \phi_x(\|x_2^c - x_1^c\|_2) & \phi_x(\|x_2^c - x_2^c\|_2) & \phi_x(\|x_2^c - x_3^c\|_2) \\ \phi_x(\|x_3^c - x_1^c\|_2) & \phi_x(\|x_3^c - x_2^c\|_2) & \phi_x(\|x_3^c - x_3^c\|_2) \end{bmatrix} \\ &= \begin{bmatrix} \phi_x(\|0-0\|_2) & \phi_x(\|0-0.6\|_2) & \phi_x(\|0-1\|_2) \\ \phi_x(\|0.6-0\|_2) & \phi_x(\|0.6-0.6\|_2) & \phi_x(\|0.6-1\|_2) \\ \phi_x(\|1-0\|_2) & \phi_x(\|1-0.6\|_2) & \phi_x(\|1-1\|_2) \end{bmatrix} \\ &= \begin{bmatrix} 0 & -0.75 & -0.9761 \\ 0.75 & 0 & -0.5590 \\ 0.9761 & 0.5590 & 0 \end{bmatrix}. \end{aligned}$$

In Matlab, the differentiation matrix is formed as $D = H/B$, and this is calculated to be

$$D = \begin{bmatrix} -2.0783 & 3.1217 & -1.1393 \\ -0.7439 & -0.8939 & 1.6313 \\ 0.5809 & -3.4902 & 2.9723 \end{bmatrix}.$$

The derivative can now be approximated by

$$\begin{aligned} f_a' &= Df \\ &= \begin{bmatrix} -2.0783 & 3.1217 & -1.1393 \\ -0.7439 & -0.8939 & 1.6313 \\ 0.5809 & -3.4902 & 2.9723 \end{bmatrix} \begin{bmatrix} 1 \\ e^{\sin(0.6\pi)} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 4.8628 \\ -1.4265 \\ -5.4810 \end{bmatrix}. \end{aligned}$$

The exact derivative of $f(x) = e^{\sin(\pi x)}$ is $f'(x) = \pi e^{\sin(\pi x)} \cos(\pi x)$. The results for this example are represented in Figure 2.4. If more centers are added, it is possible to obtain better accuracy.

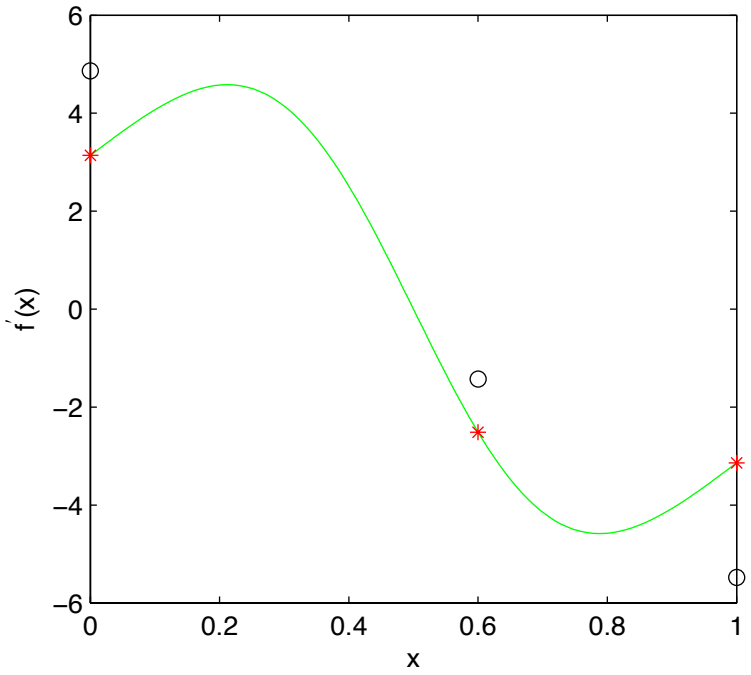

Figure 2.4: The green line represents the derivative of $f(x) = e^{\sin(\pi x)}$. The red asterisks are the exact values at the centers. The black open circles are the approximate values at the evaluation points.

Chapter 3

EFFICIENT AND ACCURATE IMPLEMENTATION OF RBF METHODS FOR TIME-DEPENDENT PDES

3.1 THE LOCAL METHOD

Up until this point in this thesis, the main focus has been on the global method for radial basis functions. In many cases, the local method can be as accurate as the global method. The main advantage to using the local method is that less computer storage and flops are needed. By definition, a flop refers to one addition, subtraction, multiplication, or division of two floating point numbers.

The local RBF interpolant takes the form

$$I_n f(x) = \sum_{k \in I_i} \alpha_k \phi(\| x - x_k^c \|_2) \tag{3.1}$$

at each of the N centers $x_1^c, \ldots, x_N^c$ in $\mathbb{R}^d$. In the above equation, I_i is a vector associated with center i that contains the center number and the indices of the $n-1$ nearest neighboring centers, α is a vector of expansion coefficients, and ϕ is a RBF. Each center and its $n-1$ neighbors are called a stencil. The α_k coefficients in Equation (3.1) are chosen by enforcing the interpolation condition so that

$$I_n f(x_k) = f_k \tag{3.2}$$

with $k \in I_i$ on each stencil. This gives N linear systems each with dimension $n \times n$ of the

form

$$B\alpha = f. \tag{3.3}$$

This equation will be solved for the expansion coefficients. Just like the global method, the matrix B is called the interpolation matrix or the system matrix with entries

$$b_{jk} = \phi(\|x_j^c - x_k^c\|_2), \qquad j,k = I_i(1), \ldots, I_i(n).$$

Because the system matrix B is always invertible, the expansion coefficients are uniquely defined on each stencil.

The derivative of a function can be approximated at the center locations by applying a linear differential operator $\mathfrak{L}$ to the local interpolant of the RBF. Depending on the problem at hand, $\mathfrak{L}$ will either be a single derivative operator or a linear combination of derivative operators. The equation

$$\mathfrak{L}f(x) = \sum_{j=1}^{N} \alpha_j \mathfrak{L}\phi(\| x - x_j^c \|_2) \tag{3.4}$$

is then obtained by evaluating it at the center where the stencil is based. The equation can be simplified to a dot product of the form

$$\mathfrak{L}f(x) = h\alpha. \tag{3.5}$$

In this case, h is a $1 \times n$ vector containing the elements

$$h_i = \mathfrak{L}\phi(\| x - x_j^c \|_2), \tag{3.6}$$

and α is the $n \times 1$ vector of RBF expansion coefficients. The equation can be simplified by recognizing that

$$\mathfrak{L}f(x_i) = hB^{-1}f(I_i) = (hB^{-1})f(I_i) = w \cdot f(I_i) \tag{3.7}$$

with the stencil weights as

$$w = hB^{-1}. \tag{3.8}$$

The weights are the solution of the linear system

$$wB = h. \tag{3.9}$$

Derivatives are approximated by multiplying the weights by the function values at the centers.

3.1.1 LOCAL DERIVATIVE EXAMPLE

For this example, $f'(x)$ is approximated for $f(x) = e^{\sin(\pi x)}$ on the interval [0,1] using the local RBF method with $\varepsilon = 0.43$. Let $x_0, \ldots, x_{10}$ be $N = 11$ evenly spaced points on the interval [0,1]. For convenience, a stencil size of $n = 3$ will be chosen. In order to approximate the derivative at $x_5 = 0.5$, theoretically the equation $f'(x_5) \approx w_4 f(x_4) + w_5 f(x_5) + w_6 f(x_6)$ would need to be solved. This is accomplished by calculating

$$\begin{aligned} B &= \begin{bmatrix} \phi(\|0.4-0.4\|_2) & \phi(\|0.4-0.5\|_2) & \phi(\|0.4-0.6\|_2) \\ \phi(\|0.5-0.4\|_2) & \phi(\|0.5-0.5\|_2) & \phi(\|0.5-0.6\|_2) \\ \phi(\|0.6-0.4\|_2) & \phi(\|0.6-0.5\|_2) & \phi(\|0.6-0.6\|_2) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1.0009 & 1.0037 \\ 1.0009 & 1 & 1.0009 \\ 1.0037 & 1.0009 & 1 \end{bmatrix} \end{aligned}$$

and

$$\begin{aligned} h &= \begin{bmatrix} \mathfrak{L}\phi(\|0.5-0.4\|_2) & \mathfrak{L}\phi(\|0.5-0.5\|_2) & \mathfrak{L}\phi(\|0.5-0.6\|_2) \end{bmatrix} \\ &= \begin{bmatrix} 0.0185 & 0 & -0.0185 \end{bmatrix} \end{aligned}$$

to find that

$$\begin{bmatrix} w_4 & w_5 & w_6 \end{bmatrix} \begin{bmatrix} 1 & 1.0009 & 1.0037 \\ 1.0009 & 1 & 1.0009 \\ 1.0037 & 1.0009 & 1 \end{bmatrix} = \begin{bmatrix} 0.0185 & 0 & -0.0185 \end{bmatrix}.$$

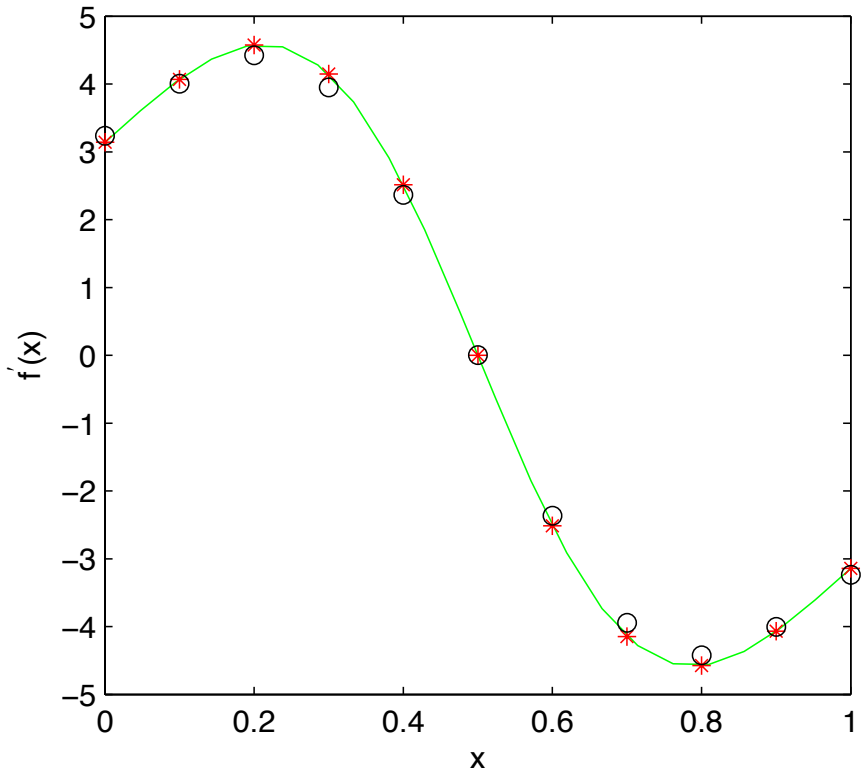

Figure 3.1: The green line represents the derivative of $f(x) = e^{\sin(\pi x)}$. The red asterisks are the exact values at the centers. The black open circles are the approximate values at the evaluation points using local RBF approximation.

Solving this system results in

$$\begin{bmatrix} w_4 & w_5 & w_6 \end{bmatrix} = \begin{bmatrix} -5 & 0 & 5 \end{bmatrix}.$$

The derivative can now be approximated at $x_5 = 0.5$ by

$$\begin{aligned} f'(x_5) &\approx w_4 f_4 + w_5 f_5 + w_6 f_6 \\ &\approx -5(2.5884) + 0(2.7183) + 5(2.5884) \\ &\approx 1.7764 \times 10^{-15}. \end{aligned}$$

The exact derivative at $x_5 = 0.5$ is 0, so this approximation is extremely accurate.

The Matlab results for this example when the local RBF method is applied at each data point are represented in Figure 3.1. The red asterisks are the exact derivatives at each of the 11 points, and the open circles represent the approximated solutions found using the local method.

3.1.2 THE LOCAL METHOD USING SINGULAR VALUE DECOMPOSITION

In order to calculate $wB = h$, the singular value decomposition (SVD) of the system matrix B is used. The SVD of $B = U \sum V^T$. Since B is an $N \times N$ matrix, U and V are $N \times N$ orthogonal matrices [26]. The matrix $\sum$ is a diagonal $N \times N$ matrix having N singular values of B as its nonzero elements. Because U and V are orthogonal, the inverse of each these two matrices is the transpose of the matrix. The inverse of the system matrix is $V \sum^{-1} U^T$, and by multiplying both sides of Equation (3.9) by B^{-1}, the weights utilized in the local method are calculated as follows:

$$w = h(V \sum^{-1} U^T).$$

Condition numbers can also be calculated using the SVD. In the following pseudocode, the shape parameter is adjusted until the condition number is in the desired range of $10^{13} \leq \kappa(B) \leq 10^{15}$:

```
condtionNumber = 1
K = conditionNumber
while (K < minConditionNumber OR K > maxConditionNumber)
     construct B
     [U,S,V] = svd(B)
          condtionNumber = maximum(S)/minimum(S)
          if K < minimumConditionNumber
               shape = shape - shapeIncrement
          elseif K > maximumConditionNumber
               shape = shape + shapeIncrement
```

3.2 GLOBAL VERSUS LOCAL RBF APPROXIMATION

As previously mentioned, the local RBF method on various stencils can be just as accurate and efficient as the global RBF method. This is especially true for derivative approximation, as the derivative is a local property.

In Section 3.1, a small example was examined that utilized global RBF interpolation. For this section, an example with a larger N that uses the global method to approximate the derivative of a function will be analyzed. It will then be compared to the local derivative method. Matlab code for the local method is provided in Appendix A.

As a motivating example, the derivative of the function

$$f(x) = e^{x^3} - \cos(2x) \tag{3.10}$$

will be approximated on the interval $[-1, 1]$ using both the global and local RBF method.

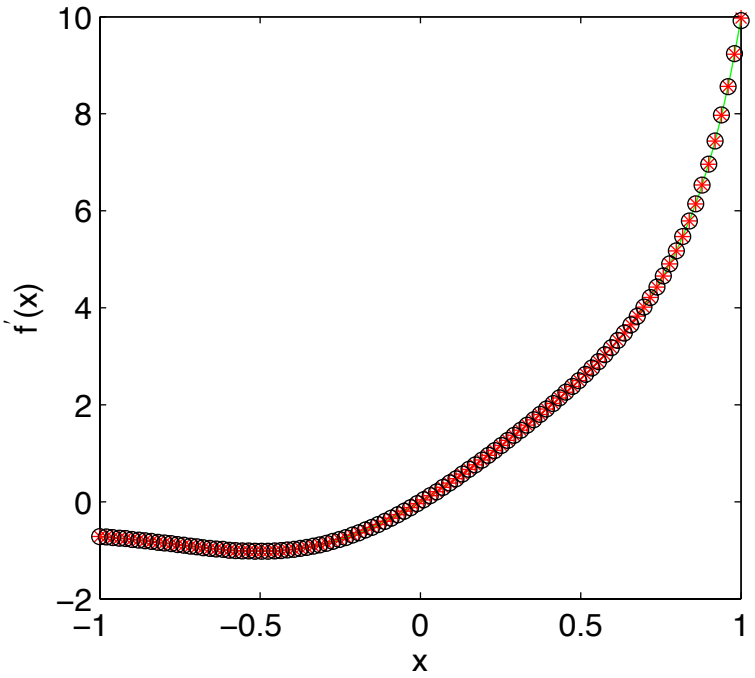

Figure 3.2: The green line represents the derivative of $f(x) = e^{x^3} - \cos(2x)$. The red asterisks are the exact values at the centers. The black open circles are the approximate values at the evaluation points using global RBF approximation.

The exact derivative of Equation (3.10) is

$$f'(x) = 3e^{x^3}x^2 + 2\sin(2x). \tag{3.11}$$

In Figure 3.2, the green line represents the derivative, $f'(x) = 3e^{x^3}x^2 + 2\sin(2x)$, and the red asterisks are the exact values of the derivative at 100 data points. The black open circles are the approximate solutions at each of the corresponding 100 data points using the global RBF method. The error ranges from 8.7128×10^{-11} to 0.0562 at the right endpoint. On average, the error is 8.3183×10^{-4}.

A typical result with RBF methods is that the accuracy of the global method can be matched with the local method with a problem dependent $n < N$. This can be seen in Table 3.1 where the local method was used to approximate the derivative of Equation (3.11). When the stencil size was equal to 7, it was found that the average error resembled the average error obtained when using the global RBF method.

Stencil Size	Average Error
3	1.6124×10^{-3}
7	5.1948×10^{-4}
15	1.8976×10^{-3}

Table 3.1: Error versus stencil size from Equation (3.11).

3.3 1D ADVECTION DIFFUSION

After the space derivatives of a PDE have been discretized by the RBF method, a system of ODEs

$$u_t = F(u) \tag{3.12}$$

remains to be advanced in time. Any numerical ODE method can be used. This approach is called the method of lines. In all numerical examples, the following fourth-order Runge-

Kutta method

$$\begin{aligned} k_1 &= \Delta t F(u^n, t^n) \\ k_2 &= \Delta t F(u^n + 0.5k_1, t^n + 0.5\Delta t) \\ k_3 &= \Delta t F(u^n + 0.5k_2, t^n + 0.5\Delta t) \\ k_4 &= \Delta t F(u^n + k_3, t^n + \Delta t) \\ u^{n+1} &= u^n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{3.13}$$

has been used [1]. The eigenvalues (scaled by Δt) of the discretized linearized operator must lie within the stability region of the ODE methods. A Runge-Kutta stability region, along with the scaled eigenvalues of a 1d advection diffusion operator, is shown in Figure 3.3. As a rule of thumb, the RBF method is stable if the eigenvalues of the linearized spatial discretization operator, scaled by Δt, lie in the stability region of the time-discretization operator [25]. As can be seen in Figure 3.3, several eigenvalues are located outside of the Runge-Kutta stability region when $\varepsilon = 1.5$. Hence, instability occurs for this one-dimensional advection-diffusion equation when the shape parameter equals 1.5. When the shape parameter was changed to $\varepsilon = 6$, all of the eigenvalues were contained in the region, and as a result, the method was stable.

The solution of the one-dimensional advection-diffusion equation

$$u_t + u_x - \nu u_{xx} = 0 \tag{3.14}$$

will be approximated on $\Omega = [0, 1]$. The exact solution for $\nu > 0$ is

$$u(x,t) = \frac{1}{2}\left[\operatorname{erfc}\left(\frac{x-t}{2\sqrt{\nu t}}\right) + \exp\left(\frac{x}{\nu}\right)\operatorname{erfc}\left(\frac{x+t}{2\sqrt{\nu t}}\right)\right]. \tag{3.15}$$

Initial and boundary conditions are prescribed according to the exact solution. Let $N = 51$ evenly spaced centers, $\varepsilon = 6$ be the shape parameter, and $\nu = 0.002$. The fourth-order Runge-Kutta method is used with a time-step of 0.005 to advance the problem until the final time is 0.5. At $t = 0.5$, the maximum point-wise error was 4.72×10^{-4}. These results

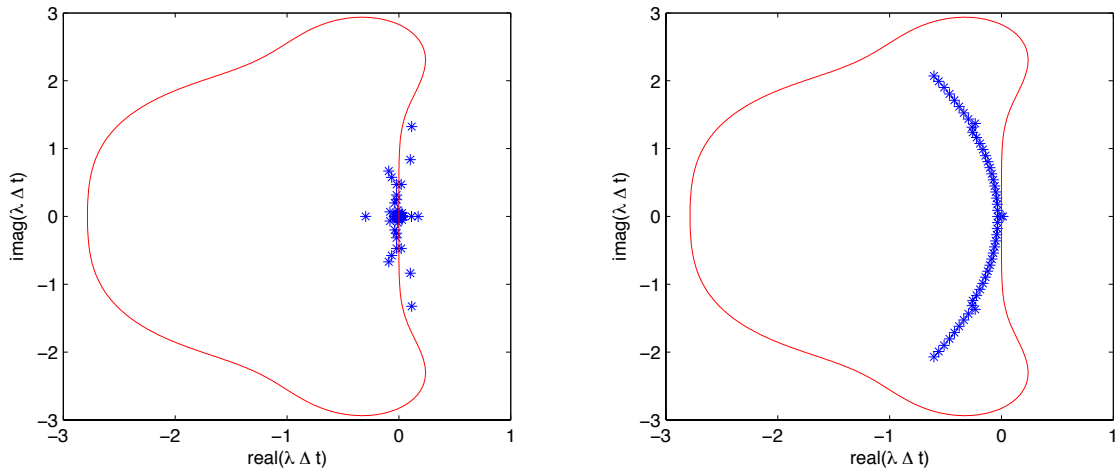

Figure 3.3: Runge-Kutta stability region with the eigenvalues from Equation (3.14). Left: Instability with $\varepsilon = 1.5$ and $\kappa(B) = 1.3482 \times 10^{19}$. Right: Stability with $\varepsilon = 6$ and $\kappa(B) = 7.5254 \times 10^{13}$.

are illustrated in Figure 3.4, and Matlab code is provided in Appendix A.

Because this is a one-dimensional problem on a small interval, the global method can be used. However, this will not be the case later in two-dimensional problems that may require as many as $N = 25{,}000$ centers.

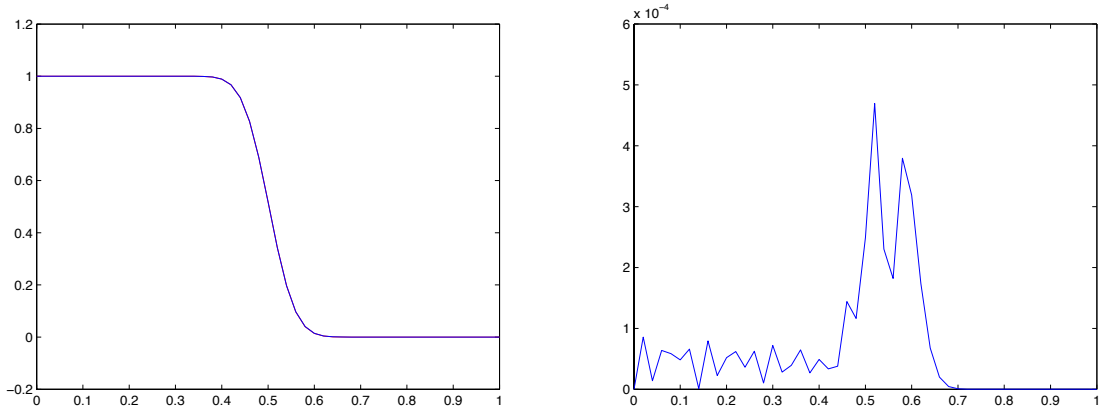

Figure 3.4: Left: Exact solution versus MQ solution of Equation 3.14. Right: Point-wise error.

3.4 2D ADVECTION DIFFUSION REACTION

This next example is of an advection diffusion reaction equation with an exact solution available. This equation has all the elements of the more complicated chemotaxis and Turing equations that will be used later. It is defined as

$$u_t = \nu(u_{xx} + u_{yy}) - \alpha(u_x + u_y) + \gamma u^2(1-u) \tag{3.16}$$

where ν is the viscosity coefficient, α is the advection coefficient, and γ is the reaction coefficient. The partial differential equation is linear with a nonlinear reaction term.

Consider Equation (3.16) with $\nu = 0.5$, $\alpha = 1$, and $\gamma = \frac{1}{\nu}$. The analytical solution is given by

$$u(x, y, t) = \frac{1}{1 + e^{a(x+y-bt)+c}} \tag{3.17}$$

where $a = \sqrt{\frac{\gamma}{4\nu}}$, $b = 2\alpha + \sqrt{\gamma\nu}$, and $c = a(b-1)$. Dirichlet boundary conditions are prescribed for $t > 0$ by using the exact solution. As a domain, take the circle of radius 1.5 centered at (1.5, 1.5).

When locating the boundary points on a circle, the results may not be very precise. By simply setting the distance from the circle's center to the boundary point equal to the radius, the computer would be storing the points as floating point numbers. This problem can be eliminated by using the distance between adding a factor of '100 $\times$ machine epsilon' to the radius and subtracting a factor of '100 $\times$ machine epsilon.' Machine epsilon refers to the smallest positive number that when added to 1 results in a number that the computer recognizes as different from 1. For double-precision, machine epsilon is equal to 2^{-52} [7].

```
bi = find((sqrt(x.^2+y.^2)<(R+100*eps)) & ...
(sqrt(x.^2+y.^2)>(R-100*eps)));
```

For this example, the global method is implemented on 4000 points in a circle beginning at a time of $t = 0$ and moving by a time step of 0.0005 until the time equals 5. The condition number was equal to 6.9955×10^{14} and the error was 7.4401×10^{-6}. The results

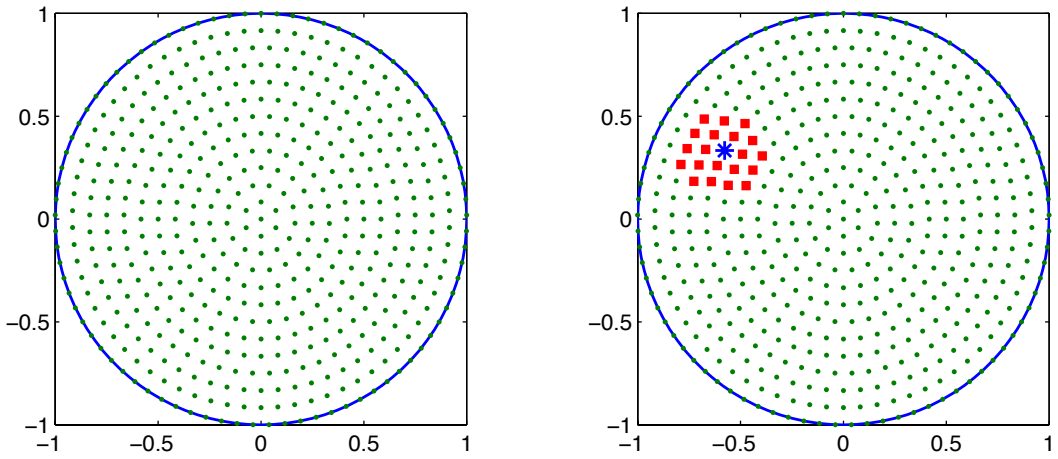

Figure 3.5: Left: Circle with N = 500 centers. Right: Stencil size of n = 21. The center of the stencil is marked with blue asterisk, and the other points in stencil are marked with red squares.

are illustrated in Figures 3.6 and 3.7. It took two minutes for the condition number to be computed and seventeen minutes for the entire program to run. When trying to increase the number of centers, out of memory errors occurred.

On an average desktop computer, the global method cannot be implemented with a large number of centers, such as 15,000, due to memory restraints when the 15,000 $\times$ 15,000 dense matrix is formed and every element has to be stored. However, it can implement the local method on these problems. Using the local method, sparse matrices are stored, and we can obtain desired accuracy. This is illustrated in Table 3.2 using Equation (3.16). When the local method is executed with $n = 15$, it is just as accurate as when $n = 300$. As the stencil size decreases, not only is less computer memory needed, but the faster the execution time of the program. An example of a circle with $N = 500$ centers with a stencil size of $n = 21$ is illustrated in Figure 3.5. A problem with a large number of centers that would be impossible to run on an average desktop computer using the global method (due to memory restraints) can execute in a matter of seconds using the local method.

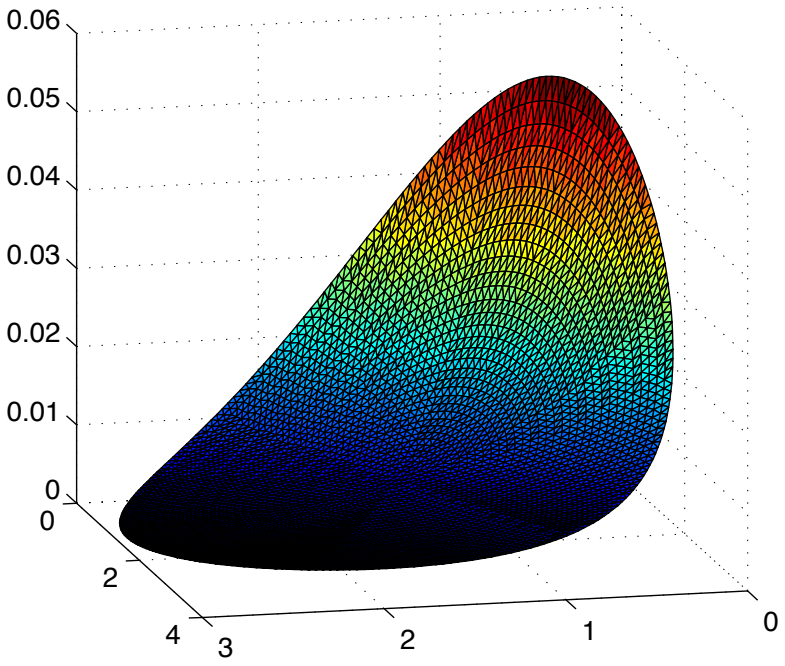

Figure 3.6: The initial condition of Equation (3.16).

Stencil Size	Time in Seconds	Max Error
300	30.693051	2.1994×10^{-5}
150	16.567052	1.4822×10^{-5}
80	9.766413	2.3395×10^{-5}
40	5.9083	1.2764×10^{-5}
15	2.744234	2.9272×10^{-5}
8	2.09120	2.9×10^{-3}

Table 3.2: Error versus stencil size from Equation (3.16).

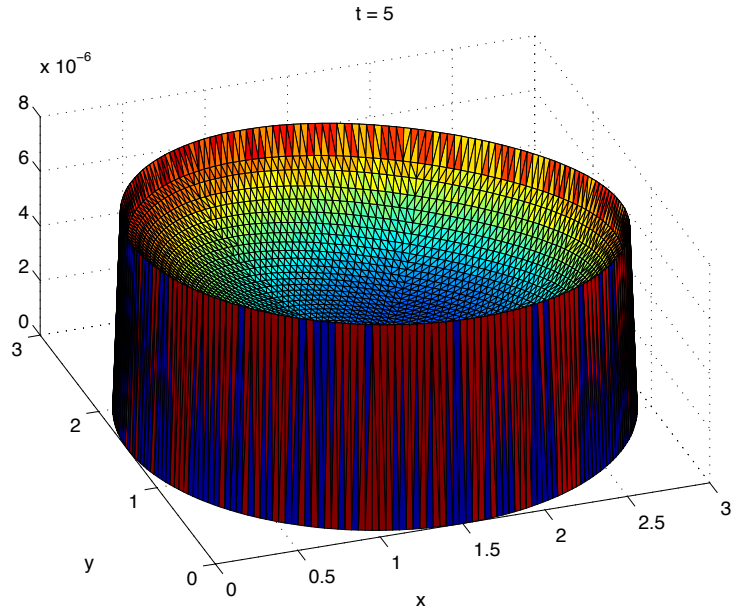

Figure 3.7: Error in the numerical solution of Equation (3.16) at t=5.

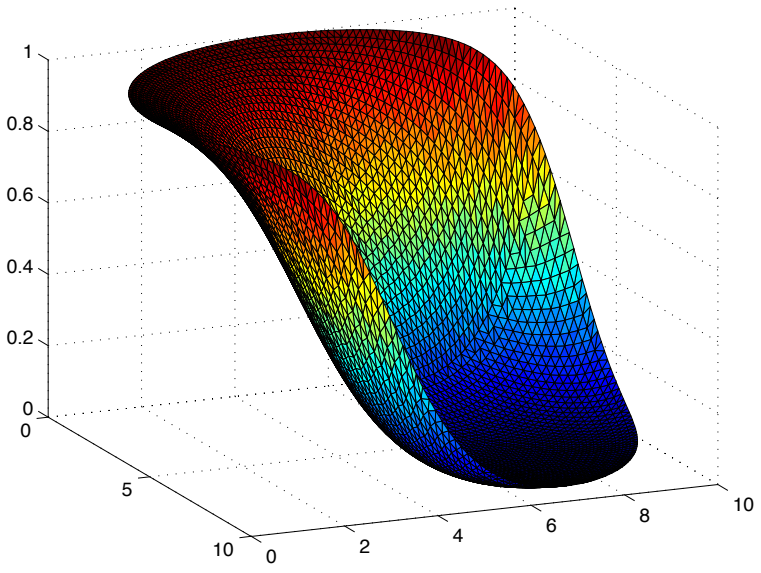

Figure 3.8: Numerical solution from the local method at t=5.

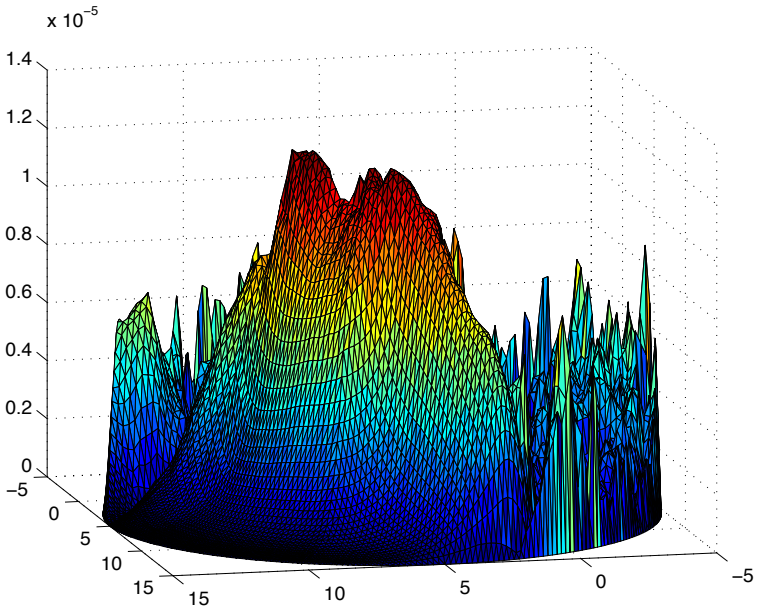

Figure 3.9: Local method error at t=5.

Chapter 4

PATTERN FORMATION IN MATHEMATICAL BIOLOGY

Many phenomena that occur in nature can be modeled using partial differential equations. More specifically, two such biological examples that utilize time-dependent advection-diffusion-reaction equations are Turing patterns and chemotaxis. RBF methods can be implemented efficiently on these types of problems with large data sets on complexly shaped domains to approximate the solutions. However, as it will be seen, the global RBF method is not always the best approach.

4.1 APPLICATION 1: TURING PATTERNS

Alan Turing (1912–1954) is perhaps most notably remembered as an early computer science visionary who conceptualized a universal computing device now known as a Turing machine. Later in his life, Turing became interested in modeling how genes of an organism can be seen in physical traits, such as spots and stripes. Examples of Turing patterns can be seen in Figures 4.2 and 4.1.

Turing patterns are modeled by a reaction-diffusion system. In addition to having a method for diffusing chemicals, this system has two primary parts. First, there is an activator that can generate more of itself. Second, there is an inhibitor. The inhibitor slows down the activator. Patterns are formed when the chemicals are affected by the activator and inhibitor and spread across a given region. In the 1980s, these patterns were able to be reproduced on a layer of gel in a petri dish by chemists and simulated on computers to show that they were indeed Turing patterns [18, 3].

Figure 4.1: Left: Many animals, such as the leopard, have prints that resemble Turing patterns. Right: Galaxies are also believed to exhibit properties of Turing patterns [13].

Turing equations are a reaction-diffusion model that have the form

$$u_t = D\nabla^2 u + f(u,v) \tag{4.1}$$

$$v_t = \nabla^2 v + g(u,v).$$

The evolution of the chemical concentration is given by $u(x,y,t)$ and $v(x,y,t)$ at spatial position (x,y) at time t. D is a constant diffusion coefficient that is a ratio of the diffusion coefficients of the two chemical concentrations. The functions f and g model the reaction. They are nonlinear reaction terms of the chemical. Under certain conditions, various patterns, such as dots or stripes, form. In addition to parameter values, these patterns vary depending on boundary conditions and the overall shape of the domain.

Consider the following Turing system

$$u_t = D\delta\nabla^2 u + \alpha u(1-\tau_1 v^2) + v(1-\tau_2 u) \tag{4.2}$$

$$v_t = \delta\nabla^2 v + \beta v(1+\frac{\alpha\tau_1}{\beta}uv) + u(\alpha+\tau_2 v)$$

on a domain shaped like a butterfly. The boundary of the domain is outlined by the

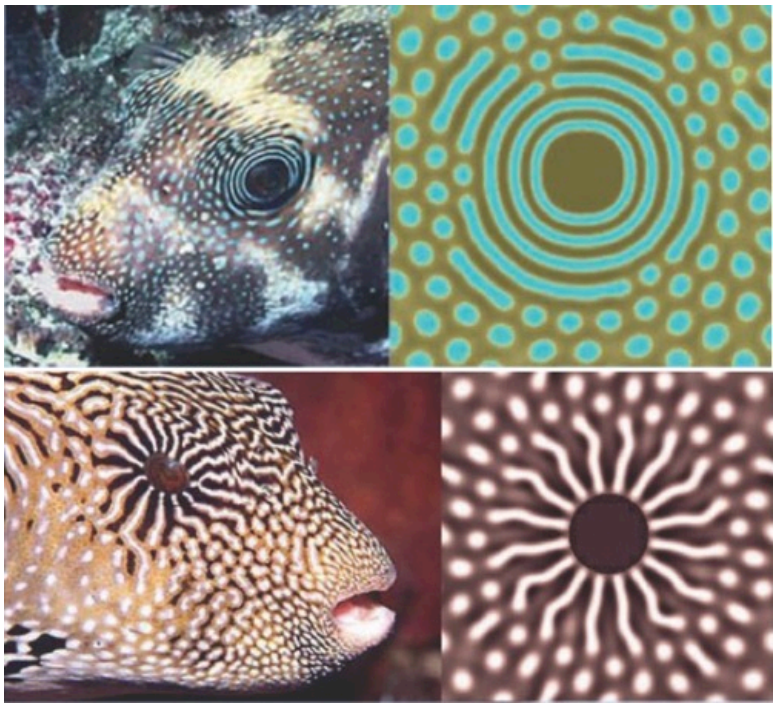

Figure 4.2: Left: Turing patterns observed on a fish. Right: Computer generated Turing pattern [13].

parametric curve

$$r(\theta) = 3e^{\sin\theta}\sin^2(2\theta) + 3e^{\cos\theta}\cos^2(2\theta), \qquad 0 \leq \theta \leq 2\pi. \tag{4.3}$$

The parameters in the equations were chosen according to the results found in [18, 3] where spotted patterns were formed with $D = 0.516$, $\delta = 0.0045$, $\alpha = 0.899$, $\tau_1 = 0.02$, $\beta = -0.91$, $\tau_2 = 0.2$, and $\gamma = -\alpha$. Zero Dirichlet boundary conditions are applied to u and v where u and v have initial conditions that were randomly chosen between -0.5 and 0.5. On the butterfly domain, there are 8,125 centers, and the local RBF method is applied on a stencil size of 100. The results demonstrate that spotted patterns can be formed from the Turing system. This is illustrated in Figure 4.3. These findings are in qualitative agreement with numerical and experimental results from [18, 3].

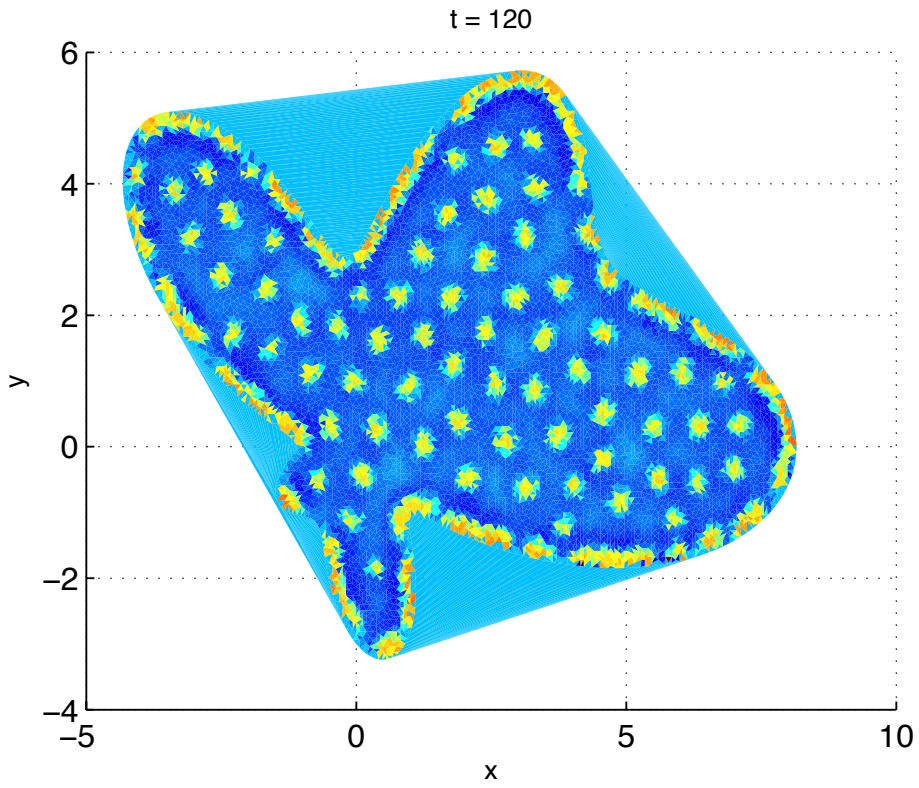

Figure 4.3: A butterfly shaped domain with the solution to the Turing system from Equation (4.2) at time $t = 120$.

4.2 APPLICATION 2: CHEMOTAXIS

Chemotaxis is an essential process in a variety of organisms ranging from the smallest bacteria to the most complex vertebrates. The single-celled prokaryotic bacteria move away from hostile conditions and toward nutrient clusters using chemotaxis. Similarly, the DNA carrying eukaryotes use chemotaxis for immune responses and wound healing. So what exactly is chemotaxis?

At its most basic level, chemotaxis is cell movement. The prefix "chemo" means "combining form" whereas the suffix "taxis" is Greek for "arrange, turning." By definition, chemotaxis is the movement of a motile cell or organism, or part of one, in a direction corresponding to a gradient of increasing or decreasing concentration of a particular substance [11].

Budrene and Berg's biological experiments on the patterns that Escherichia coli and Salmonella typhimurium form provide the basis for our research [2]. Simply stated, when bacteria were exposed to a liquid medium, patterns in the bacteria materialize and rearrange before they eventually disappear. Their experiments showed that the biological processes that caused the patterns formed by the bacteria were the results of random migration and chemotaxis.

In order for the patterns to form, the bacteria were exposed to tricarboxylic acid (TCA). Succinate and fumarate accounted for the most effective results. In response to the TCA, the bacteria released the strong chemoattractant aspartate. This chemoattractant is what causes the cells to move in a direction corresponding to a particular gradient. This movement increases cell density whereas diffusion has the opposite effect. These two combatting forces is the primary reason behind the formation of the patterns. Hence, when studying chemotaxis, we must focus on the cells, the stimulate, and the chemoattractant. In Budrene and Berg's experiments, these were respectively the bacteria, succinate and fumarate, and aspartate [2].

Tyson, Lubkin, and Murray ultimately analyzed a second order ordinary differential equation modeled after Budrene and Berg's experiments in order to understand the numerical properties of chemotaxis [27]. Their research focused on the patterns formed by E. coli and salmonella bacteria as they undergo chemotaxis in liquids. The dimensionless form of the mathematical model that was used for the experiments as defined in [27] is

$$u_t = d_u \nabla^2 u - \alpha \nabla \cdot \left[\frac{u}{(1+v)^2} \nabla v \right] \tag{4.4}$$

$$v_t = \nabla^2 v + w \frac{u^2}{\mu + u^2}, \tag{4.5}$$

where u represents the cell density, v is the chemoattractant concentration, and w represents the succinate concentration. For the experiments, the succinate is not consumed, and it is uniformly distributed in the beginning. Hence, w is constant, and let $u = 1$, $v = 0$, and $w = 1$. Also, ∇ is the gradient operator, and ∇^2 is the Laplacian operator.

In order to simplify the calculations, let $f = \frac{u}{(1-v^2)}$ and let $v = \langle v_x, v_y \rangle$. Then Equation (4.4) becomes

$$\begin{aligned} u_t &= d_u(u_{xx} + u_{yy}) - \alpha \nabla \cdot \langle f v_x, f v_y \rangle \\ &= d_u(u_{xx} + u_{yy}) - \alpha \left((f v_x)_x + (f v_y)_y \right) \\ &= d_u(u_{xx} + u_{yy}) - \alpha \left(f_x v_x + f v_{xx} + f_y v_y + f v_{yy} \right) \end{aligned} \tag{4.6}$$

and Equation (4.5) becomes

$$v_t = (v_{xx} + v_{yy}) + w \left(\frac{u^2}{\mu + u^2} \right). \tag{4.7}$$

In Matlab, this is coded as:

```
fp = zeros(N,2);


fp(:,1) =   du*(d2*V(:,1)) ...
              - alpha*((d2*V(:,2)).*(V(:,1)./(1+V(:,2)).^2)...
              + (d1x*V(:,2)).*(d1x*(V(:,1)./(1+V(:,2)).^2)))...
              - alpha*((d1y*V(:,2)).*(d1y*(V(:,1)./(1+V(:,2)).^2)));


fp(:,2) = d2*V(:,2) + (w*V(:,1).^2)./(mu + V(:,1).^2);
```

Paramater	Suggested Value	Value Used in Experiment
α	80-90	80
d_u	0.25-0.5	0.33
μ	Unknown	1

Table 4.1: Parameters for Equations (4.4) and (4.5)

The goal of this two dimensional chemotaxis experiment is to find solutions to (4.4) and (4.5) that oscillate, grow, and finally decay as time progresses. To simplify the results, like the experiments of Tyson, Lubkin, and Murray, we assumed that the bacteria do not die

[27]. Therefore, a death variable was not included in Equation (4.4). Similarly, even though a chemoattractant is produced, there is no deterioration variable in Equation (4.5). Hence, in the experiments the chemoattractant increases continually, and diffusion eventually dominates the effects of chemotaxis.

As can be seen from Figures 4.4 and 4.5, the initially random bacteria arrange themselves into high-density collections. It is obvious that the collections of cells are very dense as compared to the regions between each collection. When $t = 1$, there is one cluster in the center. However, when $t = 75$, it is evident that a set of continuous rings has developed. If we continued to increase the amount of time that passes, it is likely that the clusters will reach a steady-state pattern. This is exactly what happened in Tyson, Lubkin, and Murray's numerical experiments and Budrene and Berg's biological experiments [2, 27].

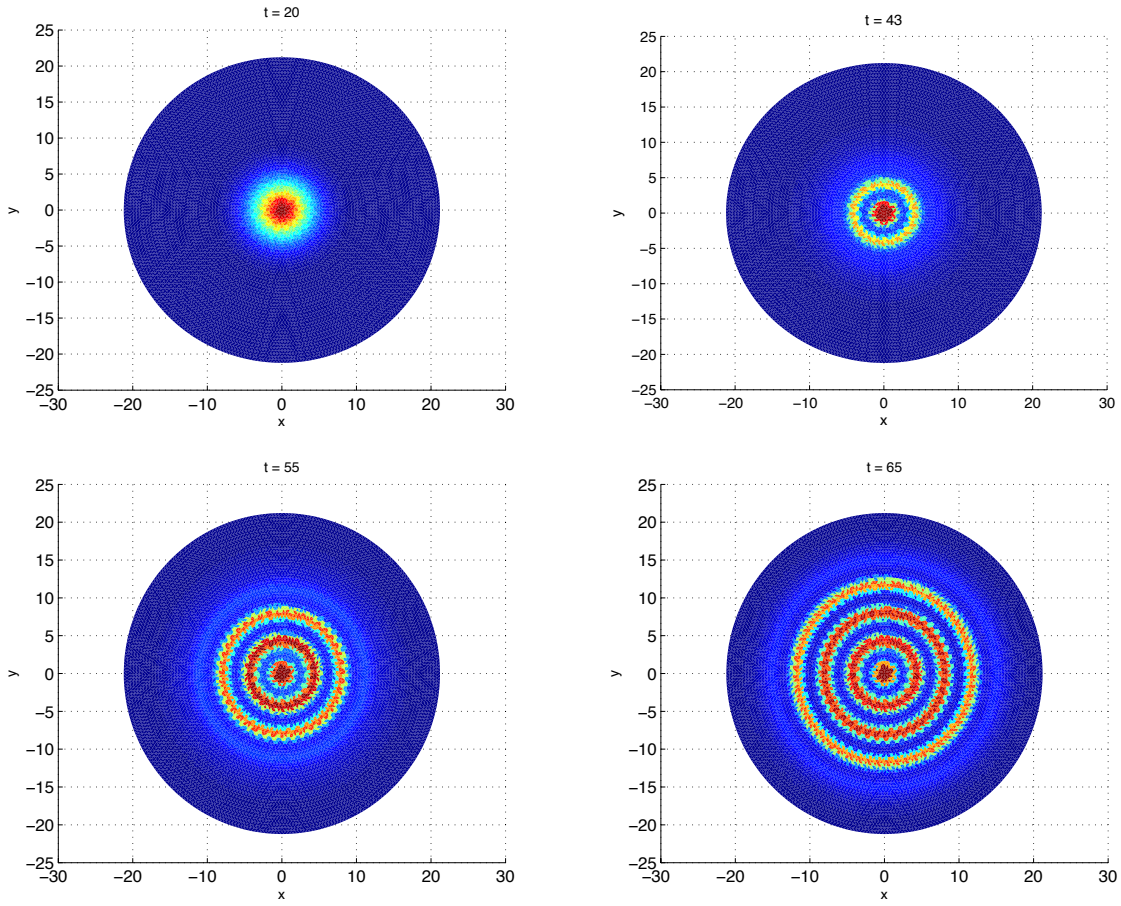

Figure 4.4: Density plots of the chemotaxis experimental results when Top Left: t=20, Top Right: t=43, Bottom Left: t=55, Bottom Right: t=65

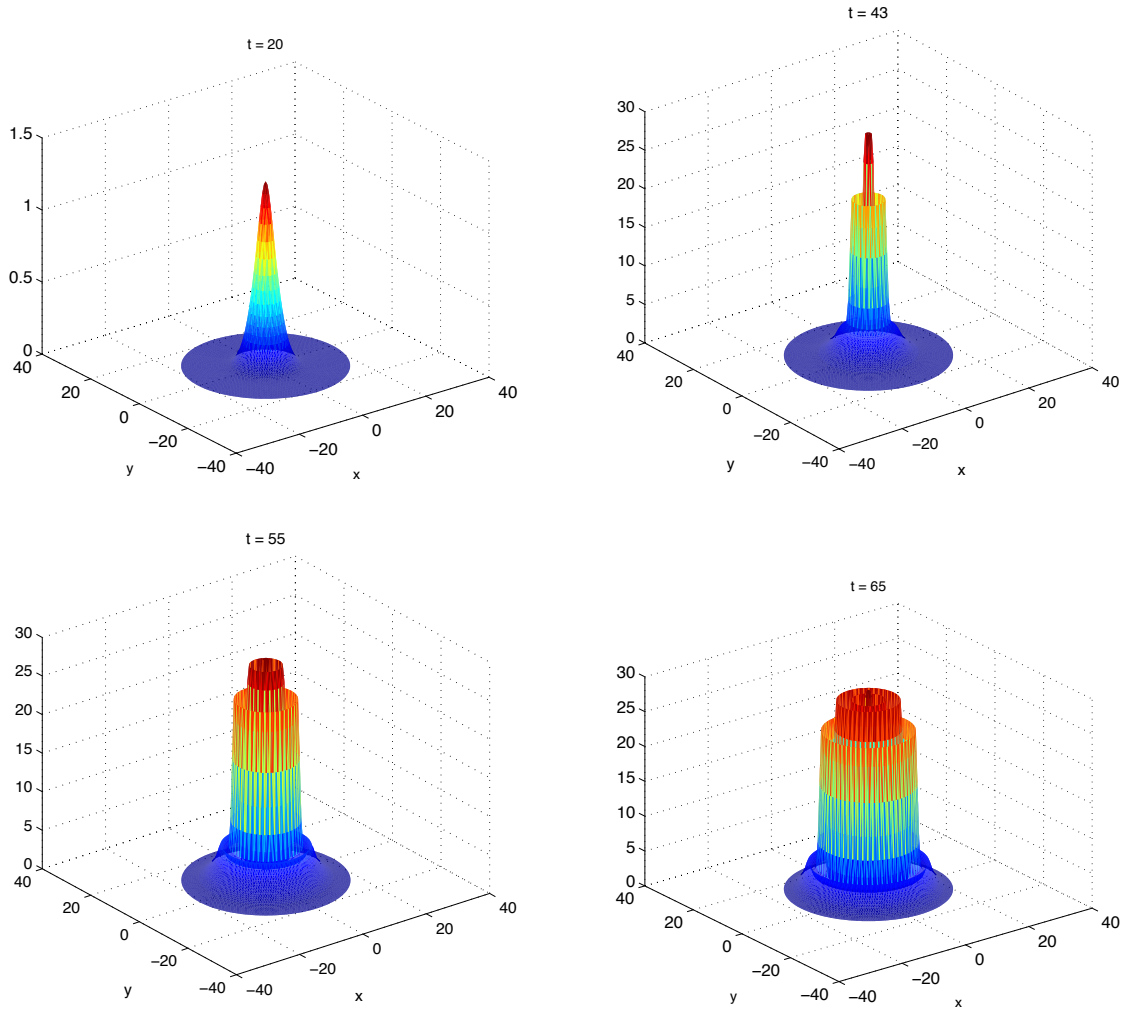

Figure 4.5: Chemotaxis experimental results when Top Left: t=20, Top Right: t=43, Bottom Left: t=55, Bottom Right: t=65

Chapter 5

CONCLUSION

Due to the extreme flexibility of the RBF methods, they have attracted much attention from mathematicians and scientists. RBF methods overcome some of the deficiencies of polynomial methods and can be applied on scattered data sites and in regions with complicated boundaries to interpolate data and to solve differential equations.

As seen in this thesis, both global and local methods can be used when working with radial basis functions. Specifically, these methods were applied to Turing equations and chemotaxis models, but the applications are endless. Similar to these examples, other situations arise that require tens of thousands of data points for centers, and desktop computers cannot efficiently execute such large problems in a timely manner with the global method. Not only is time an issue, but so is computer memory. The local method overcomes these issues by allowing the user to select small stencils of points that can provide very similar accuracy to the global method while using substantially less computer memory. As a result, the local method can execute effectively in a considerable less amount of time.

In today's modern scientific world, numerical computing is vital. Today's computing is often done heterogeneously with both CPUs and GPUs. For a future direction of this work, a high performance computer (HPC) cluster could be used to dramatically increase the accuracy and efficiency of radial basis function methods. By using a HPC cluster, it would be possible to carry out extensive calculations that require large amounts of data storage and memory allocation in a vastly shorter time frame than the typical desktop computer.

Marshall University has recently installed a HPC cluster. Known as "BigGreen," it

currently features 276 central processing unit cores, 552 gigabytes of memory, and more than 10 terabytes of storage. It currently has 8 NVIDIA Tesla M2050 GPU computing modules installed with 448 cores each. This configuration provides support for extremely large parallel computation with an estimated peak of six Teraflops. This is equal to six trillion floating point operations per second. Because RBF methods are an active research area with widespread application in engineering and science, formulating the methods to work efficiently on multi-core processors will further enhance the popularity and applicability of the RBF methods.

Appendix A

MATLAB CODE

```
%
% driverD.m
%

CENTERS = 1;        % step 1: load centers
STENCILS = 0;       % step 2: set up the stencils
WEIGHTS = 0;        % step 3: find the stencil weights
GO = 0;             % step 4: graph the solution and determine the error


if CENTERS
    disp('Centers')
    xc = linspace(-1,1,100)';
    xp = linspace(-1,1,200)';
end

if STENCILS
   disp('Stencils')
   ns = 17                      % stencils size
   st = stencilsD(xc,ns);
end

if WEIGHTS
    disp('Weights')
    shape = 0.43;       % initial shape parameter
    dc = 0.001;         % shape parameter increment
    minK = 1e5;         % minimum condition number of the system matrix
    maxK = 1e15;        % maximum condition number of the system matrix
    D = weightsD(st,xc,shape,minK,maxK,dc);
```

```
end

if GO
    disp('Graphing')
    u = go(xc,D,xp);
end
```

```
%
% stencilsD.m
%

function st = stencilsD(xc,ns)

    N = length(xc);

    for i=1:N % stencils for derivative approximation
        x0 = xc(i);
        r = abs(xc(:) - x0); %distance between center i and the rest of the centers
        [r,ix] = sort(r);
        st(i,:) = ix(1:ns);
    end
```

```
%
% weightsD.m
%

function D = weightsD(st,xc,shape,minK,maxK,dc)
% INPUTS
%      st  indexes of the stencil centers
%      xc  centers
%   shape  initial shape parameter
%    minK  min condition number of the RBF matrix (1e+13)
%    maxK  max condition number of the RBF matrix (1e+15)
%      dc  shape parameter increment
% OUTPUTS
%       D  weights to discretize

warning off
N = length(xc);              %  total centers
n = length(st(1,:));
o = ones(1,n);
```

```
D = sparse(N,n);

for i=1:N                       %  interior centers

  pn = st(i,:);
  rx = xc(pn)*o - (xc(pn)*o)';
  r = abs(rx);
  K = 1;

  while (K<minK || K>maxK)
    B = mq(r,shape);
    [U,S,V] = svd(B);
    K = S(1,1)/S(n,n);
    if K<minK,  shape = shape - dc;
    elseif K>maxK, shape = shape + dc; end
  end

 Bi = V*diag(1./diag(S))*U';

 h = mqDerivatives(sqrt((xc(i)- xc(pn)).^2), xc(i)- xc(pn), shape, 1);

 D(i,pn) = h'*Bi;

end,  warning on
```

```
%
% go.m
%

function u = go(xc,D,xp)

f = exp(xc.^3) - cos(2.*xc);

fDerivativeExact = 3.*exp(xc.^3).*xc.^2 + 2.*sin(2.*xc);

fDerivativeApprox = D*f;

    format long, format compact
    pointWiseErrors = abs(fDerivativeApprox - fDerivativeExact)
    format

```

```
    mean(pointWiseErrors)

    twoNormError = norm(fDerivativeApprox-fDerivativeExact,2)

    plot(xp,3.*exp(xp.^3).*xp.^2 + 2.*sin(2.*xp),'g',xc,fDerivativeExact,'r*', ...
                xc,fDerivativeApprox,'ko')
    xlabel 'x', ylabel 'f^\prime(x)'
```

```
function advectionDiffusion1D ()

N = 51;              %number of centers
dt = 0.005;          %timestep
finalTime = 0.5;     %final time
a = 1;               %variable in equation - advection coefficient
nu = 0.002;          %variable in equation - diffusion coefficient
shape = 6;           %shape parameter

x = linspace(0,1,N)';
o = ones(1,length(x));
rx = x*o - (x*o )'; % signed distance matrix
r = abs(rx); % distance matrix

H = zeros(N,N); % evaluation matrix

H(1,:) = mq(r(1,:),shape); % Dirichlet boundary condition at x=0
H(2:N-1,:) = nu.*mqDerivatives(r(2:N-1,:),rx(2:N-1,:),shape,2)...
    - a.*mqDerivatives(r(2:N-1,:),rx(2:N-1,:),shape,1);

B = mq(r,shape); % system matrix
dm = H/B; % discretization D, of linear operator L

U = exactSolution (x,0); %initial condition

t=0;

    while t < finalTime   %while our time is less than our final time
        u = rk4(U,t,dt,@F);
        t = t + dt; %advance in time
        u(1) = 1;
        u(N) = exactSolution(1,t);
        U=u;
```

```
    end
35
exact = exactSolution(x,finalTime);
37
plot(x,exact,'r',x,u,'b')
39
figure
41 plot(x,abs(U-exact))

43     function fp = F(u,t)
           u(1) = 1;
45         u(N) = exactSolution(1,t);
           fp = dm*u;
47     end

49     function ex = exactSolution (x,t)
           if t<dt
51             if length(x)>1
                   ex(1)=1;
53                 ex(2:length(x))=0;
                   ex = ex(:);
55             else, ex = 0;
               end
57         else
               den = 2.0*sqrt(nu*t);
59             frac1 = (x-t)/den;
               frac2 = (x+t)/den;
61             ex = 0.5*(erfc(frac1) + exp(x/nu).*erfc(frac2));
           end
63     end

65 end
```

Matlab code used for the 2d advection-diffusion-reaction problem, the Turing system, and the chemotaxis problem was modified from reference [21].

REFERENCES

[1] J. P. Boyd, *Chebyshev and Fourier spectral methods*, second ed., Dover Publications, Mineola, New York, 2000.

[2] E.O. Budrene and H.C. Berg, *Complex patterns formed by motile cells of Escheria Coli*, Nature (1991), 349:630–633.

[3] J. L. Aragon C. Varea and R. A. Barrio, *Turing patterns on a sphere*, The American Physical Society **60** (1999), no. 4, 4588–4592.

[4] B. Fornberg and N. Flyer, *Accuracy of radial basis function interpolation and derivative approximations on 1-d infinite grids*, Advances in Computational Mathematics **23** (2005), 37–55.

[5] B. Fornberg and G. Wright, *Stable computation of multi quadratic interpolants for all values of the shape parameter*, Computers and Mathematics with Applications **47** (2004), 497–523.

[6] R. Franke, *A critical comparison of some methods for interpolation of scattered data*, Technical Report NPS (1979), 53–79.

[7] D. Goldberg, *What every computer scientist should know about floating-point arithmetic*, Computing Surveys (1991), 171–264.

[8] R. L. Hardy, *Multiquadric equations of topography and other irregular surfaces*, Journal of Geophysical Research **76** (1971), no. 8, 1905–1915.

[9] ———, *Theory and applications of the multiquadric-biharmonic method: 20 years of discovery*, Computers and Mathematics with Applications **19** (1990), no. 8/9, 163–208.

[10] I. R. H. Jackson, *Convergence properties of radial basis functions*, Constructive Approximation **4** (1988), no. 1, 243–264.

[11] T. Jin and D. Hereld, *Chemotaxis methods and protocols*, Humana Press, New York, 2009.

[12] E. J. Kansa, *Multiquadrics – A scattered data approximation scheme with applications to computational fluid dynamics I: Surface approximations and partial derivative estimates*, Computers and Mathematics with Applications **19** (1990), no. 8/9, 127–145.

[13] Brandon Keim, *Alan Turing's patterns in nature, and beyond*, `http://www.wired.com/wiredscience/2011/02/turing-patterns/`.

[14] W. R. Madych and S. A. Nelson, *Bounds on multivariate interpolation and exponential error estimates for multiquadric interpolation*, Journal of Approximation Theory **70** (1992), 94–114.

[15] C. Micchelli, *Interpolation of scattered data: Distance matrices and conditionally positive definite functions*, Constructive Approximation **2** (1986), 1122.

[16] M. L. Overton, *Numerical computing with IEEE floating point arithmetic*, Society for Industrial and Applied Mathematics, Philadelphia, 2001.

[17] R. B. Platte, *How fast do radial basis function interpolants of analytic functions converge?*, IMA Journal of Numerical Analysis **31** (2011), no. 4, 1578–1597.

[18] J. L. Aragon R. A. Barrio, C. Varea and P. K. Maini, *A two-dimensional numerical study of spatial pattern formation in interacting turing systems*, Bulletin of Mathematical Biology **61** (1999), 483–505.

[19] S. Rippa, *An algorithm for selecting a good value for the parameter c in radial basis function interpolation*, Advances in Computational Mathematics **11** (1999), 193–210.

[20] S. A. Sarra, *Radial basis function approximation methods with extended precision floating point arithmetic*, Engineering Analysis with Boundary Elements **35** (2011), no. 1, 68–76.

[21] ———, *A local radial basis function method for advection-diffusion-reaction equations on complexly shaped domains.*, To appear in Applied Mathematics and Computation (2012).

[22] S. A. Sarra and E. J. Kansa, *Multiquadric radial basis function approximation methods for the numerical solution of partial differential equations*, vol. 2, Advances in Computational Mechanics, 2009.

[23] T. Sauer, *Numerical analysis*, Pearson Education, Inc., Boston, 2006.

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

[25] L. N. Trefethen, *Spectral methods in Matlab*, SIAM, Philadelphia, 2000.

[26] L. N. Trefethen and III D. Bau, *Numerical linear algebra*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1997.

[27] R. Tyson, S.R. Lubkin, and J.D. Murray, *Model and analysis of chemotactic bacterial patterns in a liquid medium*, Journal of Mathematical Biology (1998), 38:359–375.

[28] H. Wendland, *Scattered data approximation*, Cambridge University Press, Cambridge, 2005.

[29] Z. Wu and R. Schaback, *Local error estimates for radial basis function interpolation of scattered data*, IMA Journal of Numerical Analysis **13** (1993), 13–27.

[30] J. R. Xaio and M. A. McCarthy, *A local heaviside weighted meshless method for two-dimensional solids using radial basis functions*, Computational Mechanics **31** (2003), 301–315.

Maggie Elizabeth Chenoweth
chenoweth8@marshall.edu

Education

- Master of Arts. Mathematics. Marshall University, May 2012. Thesis Advisor: Scott Sarra.
- Bachelor of Arts. Double major in Applied Mathematics and Math Education 5-Adult. Minor in Computer Science. Marshall University, May 2010, *Summa cum laude*.

Teaching Experience

- Student Teaching - Chesapeake Middle School (John Gibson) and Chesapeake High School (Dee Rucker) - Spring 2010
- MTH 099 - Mathematics Skills II - Spring 2011
- MTH 121 - Concepts and Applications (Critical Thinking) - Fall 2010 and Fall 2011
- MTH 127 - College Algebra (Expanded) - Summer 2011 and Fall 2011
- MTH 225 - Introductory Statistics - Spring 2012

Publications

1. *A Numerical Study of Generalized Multiquadric Radial Basis Function Interpolation.* Maggie Chenoweth. SIURO 2 n. 2 (October 2009), 58–70.
2. *A Local Radial Basis Function Method for the Numerical Solution of Partial Differential Equations.* Master's thesis, Marshall University, May 2012.

Professional Affiliations

- Pi Mu Epsilon - Former Chapter President at Marshall University
- National Society of Collegiate Scholars
- Golden Key International
- Gamma Beta Phi
- Sigma Xi
- Society for Industrial and Applied Mathematics
- Phi Kappa Phi
- Kappa Delta Pi
- Mathematical Association of America
- American Mathematical Society

Awards and Recognitions

- John Marshall Scholar
- Graduate of the Marshall University Honors Program
- Dean's List
- National Dean's List
- 200 Level Book Award for the Marshall University Honors Program
- NASA Space Grant Recipient
- Presenter at the West Virginia Undergraduate Research Day at the Capitol
- Poster Presentation Winner at Marshall's Cyberinfrastructure Day
- Marshall Math Department Junior Student of the Year
- Outstanding Graduate in Mathematics Award
- Outstanding Graduate in the College of Education Award