Spring 1-1-2011

A Stable Numerical Algorithm for Near-Flat Radial Basis Functions

Collin Eugene Powell
University of Colorado at Boulder, collin.powell@gmail.com

Follow this and additional works at: https://scholar.colorado.edu/appm_gradetds

Part of the Applied Mathematics Commons

Recommended Citation
https://scholar.colorado.edu/appm_gradetds/23

This Thesis is brought to you for free and open access by Applied Mathematics at CU Scholar. It has been accepted for inclusion in Applied Mathematics Graduate Theses & Dissertations by an authorized administrator of CU Scholar. For more information, please contact cuscholaradmin@colorado.edu.
A Stable Numerical Algorithm for Near-Flat Radial Basis Functions

by

Collin E. Powell

B.A., Oberlin College, 2003

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Master of Science
Department of Applied Mathematics
2011
This thesis entitled: 
A Stable Numerical Algorithm for Near-Flat Radial Basis Functions 
written by Collin E. Powell 
has been approved for the Department of Applied Mathematics

__________________
Bengt Fornberg

__________________
Natasha Flyer

__________________
Jeffrey Fox

Date ______________

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.
We present here a new stable algorithm for the small $\epsilon$ (flat basis function) limit in Radial Basis Function interpolation. Based on the successes of previous methods, such as RBF-QR, we extend to the 2-D non-periodic domain using the Gaussian basis function. This new method, RBF-GA, shows substantial improvement in accuracy and conditioning over the RBF-Direct method, and it is also easy to implement. The theory involved can readily be extended to higher numbers of dimensions.
Dedication

To my parents, Bob and Elizabeth, for their love and support during my trying times.
Acknowledgements

I would like to start by thanking my advisor, Bengt Fornberg, for his great help and patience through the ups and downs of this process, and for not giving up on me when times seemed at their worst. My illness took me out for a long time but he made it easy to come back. I would also like to thank Natasha Flyer for her inspiring enthusiasm and Jeff Fox for being a great mentor and friend.

To my friends and family, thank you for the support and helping to put me back on my feet when I had been knocked down.
# Contents

## Chapter

1. **Introduction**
   
   1.1 Background ........................................... 1
   1.2 Basic Formulation ..................................... 2
   1.3 Properties of RBFs .................................... 3
       1.3.1 Accuracy ....................................... 6
       1.3.2 Computational Cost .............................. 8
       1.3.3 Stability ....................................... 8

2. **Main Thesis**
   
   2.1 Problem Statement .................................... 11
   2.2 Initial attempts using the IQ and MQ methods ......... 11
   2.3 Counting Problem Solved .............................. 13
   2.4 RBF-GA ............................................. 15
   2.5 Numerical Results for Interpolation .................... 19
   2.6 Polynomial vs. RBF-Generated Finite Difference Formulas .... 19
   2.7 Concluding remarks .................................. 32

## Bibliography

36
## Figures

**Figure**

1.1 A graphical representation of how the RBF method performs and interpolation. 4

1.2 A characterization of the uncertainty barrier in the small $\epsilon$ realm. 7

2.1 A plot of the condition number as a function of $M$ for $N = 20, 22, 25,$ and $30$, $\epsilon = 0.1$ for all cases. When $M = N$ here we get the condition number for the original RBF interpolant. Only in the cases $N = 22$ and 25 do we see a substantial improvement in conditioning when $M = 6$ (thus the first 6 basis functions are the original RBF and the remainder use the new method presented in equation 2.2). 14

2.2 The Gaussian basis functions with $\epsilon = 10^{-4}$ centered at 15 nodes distributed over the domain $[-1, 1] \times [-1, 1]$. These basis functions are visually indistinguishable from each other, and from the constant function. 20

2.3 The basis functions generated using the RBF-GA algorithm using the same nodes as in figure 2.2. They are arranged into rows corresponding to the different levels from equations (2.12),(2.13),(2.14), and further. It should be noted that these functions look very similar to the 2-D polynomial basis functions of the same order. 21
2.4 Comparison of the RBF-GA to the RBF-Direct methods. On the left we have the absolute error values for the two methods using 21, 45, and 91 nodes. On the right we have sup-norm difference between the resulting interpolants the two methods produce. ................................................................. 22

2.5 The high precision RBF-Direct error in calculating the interpolant for the function defined in equation (1.3) using 91 Halton sequence generated node locations within the unit circle. ......................................................... 23

2.6 The max norm error comparison between weights generated for \( L \) using RBF-GA and standard precision RBF-Direct compared to high precision RBF-Direct using a 10 node stencil. ................................................................. 26

2.7 The max norm error comparison between weights generated for \( L \) using RBF-GA and standard precision RBF-Direct compared to high precision RBF-Direct using a 15 node stencil. ................................................................. 27

2.8 The max norm error comparison between weights generated for \( L \) using RBF-GA and standard precision RBF-Direct compared to high precision RBF-Direct using a 36 node stencil. ................................................................. 28

2.9 The max norm error comparison between weights generated for \( L \) using RBF-GA and standard precision RBF-Direct compared to high precision RBF-Direct using a 105 node stencil. ................................................................. 29

2.10 The computation time in seconds for the three methods in generation of the finite difference stencils. Here RBF-VPA stands for Matlab’s Variable Precision Arithmetic using 200 digits of precision, the precision used for the previous error analysis plots. ................................................................. 30

2.11 The condition number for the 36 node stencil as a function of \( \epsilon \). We immediately see that for large values of \( \epsilon \) the condition number for the RBF-GA method grows out of control. ................................................................. 31
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.12</td>
<td>Condition number for the 36 node stencil system using a random ordering of nodes, again compared with RBF-Direct.</td>
<td>33</td>
</tr>
<tr>
<td>2.13</td>
<td>Condition number for the 36 hexagonally located node stencil system using a random ordering of nodes.</td>
<td>34</td>
</tr>
<tr>
<td>2.14</td>
<td>Condition number for the 36 hexagonally located node stencil system using a distance ordering of nodes.</td>
<td>35</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Radial Basis Function Interpolation has been used for many years in a variety of applications for several reasons. First and foremost is the lack of reliance on and particular grid structure for interpolation. RBFs also can be extended into any number of dimensions without increase in code complexity (in many cases the same code can be used for arbitrary number of dimensions). One road block that often arrises is the trade off of accuracy and numerical conditioning in the flat basis function limit. As the basis functions get flatter the accuracy of the method generally improves. However, the condition number for the linear system grows very rapidly as well. Over the last several years many methods have been developed to address this problem. In this paper we present another such method using the Gaussian RBF which has the particularly strong property of having 'super-spectral' accuracy. This new method is very easy to implement algorithmically and presents a very useful means of constructing scattered node finite difference stencils for solving differential equations.

1.1 Background

Radial Basis functions were first introduced as a method for analyzing topographical survey data in 1971 by Rolland Hardy [11]. For this application they had the great advantage of allowing data to fall outside a structured grid thus freeing the analysis to use any survey data that was available. At the time no other interpolation method could be used, as they were either too smooth or too oscillatory for the sparsely scattered survey data. The
method that Hardy originally developed used the multiquadric radial function (see section 1.2). Hardy’s method had another crucial advantage. The collocation matrix associated with the multiquadric was non-singular for any distinct set of node locations, a characteristic which the other methods of the time did not share according to Haar’s theorem [15]. Radial basis functions gained additional influence in 1982 when Richard Franke produced a study on 32 commonly used interpolation methods [10], one of which was the multiquadric method developed by Hardy. In his study, Franke subjected the methods to rigorous tests for properties such as accuracy, versatility, and stability. He concluded that Hardy’s method was the overall best. Franke conjectured that the non-singularity property of the collocation matrix for this method was independent of the node locations. He was unable to prove this. This was left to Charles Micchelli in 1986 [14].

Hardy’s mutiquadric method, like other interpolation methods, created an approximation function which was a linear combination of mathematically simple basis functions. In this case the basis functions were translations of a common radially symmetric function (often one such function being centered on each distinct node location, though this is not necessary). When this method was generalized with the use of functions other than the multiquadric its name was changed to the Radial Basis Function (RBF) method we use today. In 1990 Ed Kansa demonstrated RBF’s ability to solve a variety of parabolic, hyperbolic, and elliptical partial differential equations [12, 13].

1.2 Basic Formulation

The basic formulation of the radial basis function method is to construct an interpolant as a linear combination of translates of a particular radially symmetric function. The interpolant thus has the following form:

\[ \Phi(x) = \sum_{i=1}^{n} \lambda_i \phi(\|x - x_i\|) \]  

(1.1)
where $\phi$ is the basis function centered at the $i$th node location, $\|x - x_i\|$ is the euclidean 2-norm distance and the $\lambda_i$’s represent the weights associated with each translate. These weights are determined by the solution to the linear system:

$$
\begin{bmatrix}
\phi(\|x_1 - x_1\|) & \phi(\|x_1 - x_2\|) & \cdots & \phi(\|x_1 - x_n\|) \\
\phi(\|x_2 - x_1\|) & \phi(\|x_2 - x_2\|) & \cdots & \phi(\|x_2 - x_n\|) \\
\vdots & \vdots & \ddots & \vdots \\
\phi(\|x_n - x_1\|) & \phi(\|x_n - x_2\|) & \cdots & \phi(\|x_n - x_n\|)
\end{bmatrix}
\begin{bmatrix}
\lambda_1 \\
\lambda_2 \\
\vdots \\
\lambda_n
\end{bmatrix}
= 
\begin{bmatrix}
f_1 \\
f_2 \\
\vdots \\
f_n
\end{bmatrix}
$$

(1.2)

A graphical representation of this can be seen in figure 1.1.

Often times, however, we introduce an additional parameter called the shape parameter, or $\epsilon$, into the system, changing its form to:

$$
\Phi(x) = \sum_{i=1}^{n} \lambda_i \phi(\epsilon \|x - x_i\|)
$$

(1.3)

This shape parameter is most often used in instances where the radial function is infinitely smooth. It’s effect is to “flatten” or “sharpen” the functions resulting in different final interpolations. Ultimately it is this parameter and its effect that is the subject of this thesis. We will discuss this in depth later. It is sufficient to note that as $\epsilon$ goes to zero the basis function becomes flat, or simply a constant function of euclidean distance.

Table 1.1 contains a list of some of the more common basis functions used by the direct method displayed here.

1.3 Properties of RBFs

When looking at any interpolation method a few aspects need to be analyzed. Among these are the method’s level of accuracy, its computational cost, and its stability. Michelli proved that the linear system in the direct RBF method would be non-singular for any distinct set of node locations for the multiquadric basis function. Before Michelli’s proof, Bochner and Schoenberg proved (in 1933 and 1938, respectively) that any basis func-
Figure 1.1: A graphical representation of how the RBF method performs and interpolation.

<table>
<thead>
<tr>
<th>Name of RBF</th>
<th>Abbreviation</th>
<th>$\phi(r), r \geq 0$</th>
<th>Smoothness</th>
</tr>
</thead>
<tbody>
<tr>
<td>multiquadric</td>
<td>MQ</td>
<td>$\frac{1}{\sqrt{1+(er)^2}}$</td>
<td>Infinitely smooth</td>
</tr>
<tr>
<td>inverse multiquadric</td>
<td>IMQ</td>
<td>$\frac{1}{\sqrt{1+(er)^2}}$</td>
<td></td>
</tr>
<tr>
<td>inverse quadratic</td>
<td>IQ</td>
<td>$\frac{1}{1+(er)^2}$</td>
<td></td>
</tr>
<tr>
<td>Generalized multiquadric</td>
<td>GMQ</td>
<td>$(1 + (er)^2)^\beta$</td>
<td></td>
</tr>
<tr>
<td>Gaussian</td>
<td>GA</td>
<td>$e^{-(er)^2}$</td>
<td></td>
</tr>
<tr>
<td>Thin Plate Spline</td>
<td>TPS</td>
<td>$r^2 \log(r)$</td>
<td>Piecewise smooth</td>
</tr>
<tr>
<td>Linear</td>
<td>LN</td>
<td>$r$</td>
<td></td>
</tr>
<tr>
<td>Cubic</td>
<td>CU</td>
<td>$r^3$</td>
<td></td>
</tr>
<tr>
<td>Monomial</td>
<td>MN</td>
<td>$r^{2k-1}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.1: The definitions of some common Radial Basis Functions.
tion which was itself completely monotonic would produce a linear system which is strictly positive definite and therefore unconditionally non-singular [1, 17].

**Definition: Positive Definite Matrices** A real symmetric matrix $A$ is called *strictly positive definite* if its associated quadratic form is positive

$$
\sum_{j=1}^{n} \sum_{k=1}^{n} c_j c_k A_{jk} > 0 \quad (1.4)
$$

for all non-vanishing $c \in R^n$. Consequently, the eigenvalues of a positive definite matrix are all strictly positive.

Below are three key theorems regarding non-singularity of RBF matrices. For further theorems and proofs see [20, 3].

**Theorem 1** Assume that $d$ is any positive integer and that the points $x_i \in R^d, i = 1, 2, ..., n$, are all distinct. If $\phi$ can be written in the form

$$
\phi(r) = \int_0^\infty e^{-ar^2} \omega(\alpha) d\alpha, \quad (1.5)
$$

where $\omega(\alpha) \geq 0$ for $\alpha \geq 0$ and $\int_0^\infty \omega(\alpha) d\alpha > 0$ for some $\delta > 0$, then the collocation matrix $A$ with entries $A_{ij} = \phi(||x_i - x_j||)$ is positive definite.

**Definition: Completely Monotonic Functions** A function $\phi(r) = \int_0^\infty e^{-ar^2} \omega(\alpha) d\alpha, r \geq 0$, where $\omega \geq 0$ is said to be completely monotonic on $[0, \infty)$ if, when considering

$$
\psi(r) = \phi(\sqrt{r}) = \int_0^\infty e^{-\alpha r} \omega(\alpha) d\alpha \quad (1.6)
$$

- $\psi(r) \geq 0$, and
- $(-1)^k \psi^{(k)}(r) \geq 0, r \geq 0$ for all positive integers $k$.

**Theorem 2** $\phi(r)$ can be expressed as $\int_0^\infty e^{-ar^2} \omega(\alpha) d\alpha$ *if and only if* $\psi(r) \geq 0, r \geq 0$ is completely monotonic.
Theorem 3 Let $\psi(r) = \phi(r^{1/2}) \in C^0[0, \infty)$, $\psi(r) > 0$ for $r > 0$, and $\psi'(r)$ completely monotone but not constant on $(0, \infty)$. Then for any set of $n$ distinct points $\{x_j\}_{j=1}^n$, the $n \times n$ matrix $A$ with entries $A_{ij} = \phi(||x_i - x_j||)$ is non-singular.

1.3.1 Accuracy

The accuracy of the radial basis function interpolation comes in two forms. In the case of piecewise continuous basis functions the accuracy is algebraic, i.e. of the form $O(h^k)$ where, in 1-D, $k$ is generally one greater than the level of continuity and $h$ is inversely proportional to the number of nodes and thus is a general measure of the average distance between two neighboring node points. For example, the cubic RBF $\phi(r) = r^3$, which produces the cubic spline interpolant, has accuracy of $O(h^4)$. In general, a basis function of the form $\phi(r) = r^{(2k-1)}$ will produce an interpolant with a jump in the $(2k-1)$st derivative and will therefore have accuracy of $O(h^{2k})$. We have to notice here that a basis function of the form $\phi(r) = r^{2k}$ cannot be used as it reduces to a polynomial of degree $2k$ and thus cannot be used to interpolate more than $(2k+1)$ nodes in one dimension. In higher dimensions convergence rates typically become greater.

The other form is infinitely smooth basis functions. These basis functions will have no jumps in any of their derivatives and will produce an interpolant which has spectral accuracy, or $O(e^{-c/h})$ convergence. The Gaussian basis function, which we use in this thesis for our numerically stable method described later, has an exceptional $O(e^{-c/h^2})$ convergence rate, denoted super-spectral convergence [4].

The shape parameter, $\epsilon$, also plays an important role for the convergence rate of infinitely smooth RBFs. For a fixed number of nodes, the interpolation error is found to decrease exponentially until some optimal value is reached, then generally a Runge-like phenomenon begins to gain influence and either increases the error or prevents it from decreasing any further.

It is interesting to note that though the value of the shape parameter $\epsilon = 0$ produces
Figure 1.2: A characterization of the uncertainty barrier in the small $\epsilon$ realm.
a singular collocation matrix, Fornberg et. al. found that the limit \( \epsilon \to 0 \) reproduces the Lagrange interpolation polynomial in one dimension [2]. The problem remains, however, that attempting to compute the interpolant for small values of \( \epsilon \) gives very poorly conditioned systems and thus many digits of accuracy are lost solely to the finite precision calculations used in the execution of the computation. Much work has been done to solve this problem, including the Contour-Padé method, the RBF-QR method, and the method discussed in this thesis [8, 7]. We note at this point that RBF interpolation for small \( \epsilon \) is a well conditioned process that traditionally has been approached in the numerically ill-conditioned manner of successfully using (1.2) and (1.3).

1.3.2 Computational Cost

Computational cost remains a major road block for the RBF interpolation method. Since most basis functions used are global, this leads to collocation matrixes which are dense. Because the direct method then requires a full matrix inversion, the computational cost of the algorithm is of \( O(n^3) \) complexity. There are ongoing efforts to improve the speed of the method, but to date there is still no fast and robust algorithm. A very promising approach in the area of solving PDE’s is the RBF-FD method (RBF generated finite difference stencils). This method uses global RBF’s to generate local finite difference stencils [18, 21, 6].

1.3.3 Stability

The numerical stability of the RBF method is the focus of this thesis. As mentioned earlier, the method guarantees a non-singular collocation matrix for many smooth basis functions, regardless of the number of nodes. However, the condition number of the system increases very rapidly both with the number of nodes and decreases in the shape parameter (for smooth basis functions). This phenomena lead to what Schaback called the ”Uncertainty Principle” for RBF’s, that there exists a lower limit for the value of \( \epsilon \) below which the interpolation becomes nonsensical [16]. This created a supposedly inescapable trade-off
between accuracy and conditioning.

The Contour-Padé and RBF-QR methods both show that this so-call uncertainty principle for RBF’s is breakable. They show that the interpolant can be computed for an arbitrarily small value of the shape parameter \( \epsilon \). Both methods, however, have their limitations. In this thesis we explore another method to defeat this barrier for the purpose of unlocking the small \( \epsilon \) limit. The original RBF-QR method defeated this barrier on the surface of a sphere by expanding the basis functions in terms of the spherical harmonics. In doing so it expressed the entries in the RBF matrix in a form that allowed powers of \( \epsilon \) to be factored out at the exact same rate as they appear in the eigenvalues of the matrix (as these eigenvalues are the source of the uncertainty principle).

\[
\begin{array}{c}
\text{2-D non-periodic case} \\
1 2 3 4 5 6 7 ...
\end{array}
\begin{array}{c}
\text{On the surface of a sphere} \\
1 3 5 7 9 11 13 ...
\end{array}
\begin{array}{c}
\text{3-D non-periodic case} \\
1 3 6 10 15 21 28 ...
\end{array}
\]

Table 1.2: The number of eigenvalues of order \( O(1) \), \( O(\epsilon^2) \), \( O(\epsilon^4) \), \( O(\epsilon^6) \), etc. in cases of the smooth RBF types listed in table 1.1 [9]. It is here assumed that the nodes are not located on any regular lattice.

In order to demonstrate the formation of the uncertainty barrier we look at the following example. Assume that we are using the Gaussian basis function in 2-D with \( N = 21 \) node locations.

\[
N = 21 = 1 + 2 + 3 + 4 + 5 + 6
\]

(1.7)

Thus our largest eigenvalue is order \( \epsilon^0 \) and our smallest is order \( \epsilon^{10} \) as follows from the data in the table above (Table 1.2). Since the condition number of the collocation matrix is the ratio of its largest to smallest eigenvalue, our condition number for this example would be \( \epsilon^{-10} \). If we were to then allow \( \epsilon \) to fall to \( 10^{-3} \), this would result in a condition number of
$10^{-30}$. Computationally speaking this means we would expect to lose 30 digits of accuracy in the inversion. Since standard double precision floating point holds only 16 digits, the direct inversion would loose many more digits than are available, introducing a very large amount of error. And this, again, is with a very small number of nodes. As the number of nodes increases, the condition number grows even more rapidly as $\epsilon$ decreases. For more details, see [9].
Chapter 2

Main Thesis

2.1 Problem Statement

The main purpose of this thesis is to look into the small $\epsilon$ problem. Two of the leading methods that have been developed to allow stable numerical calculations of the interpolant for small $\epsilon$ are the Contour-Padé method and the RBF-QR method [7, 8]. Contour-Padé allows stable calculation for any $\epsilon$ value (individually to $\epsilon \to 0$ limit) but has high computational cost as well as having certain restrictions involving identification of singularities in the complex plain.

The original RBF-QR method was computationally fast (relatively speaking), had no complex plane restrictions, but was limited to interpolation on the surface of a sphere [7]. Later versions were developed to remove this restriction, and RBF-QR can now be applied to arbitrary node configurations in 1-D, 2-D, and 3-D. [5]

2.2 Initial attempts using the IQ and MQ methods

The goal in the present study was to create a new stable method that would be usable in general 2-D (and 3-D) non-periodic space, without the computational limitations of the Contour-Padé method. RBF-QR on a sphere defeated the small $\epsilon$ problem by replacing the original basis functions with linear combinations of the spherical harmonics, in order to factor the problematic $\epsilon$ terms out of the system prior to numerical calculation. Thus we began the present work by exploring a similar approach when using the IQ and MQ basis
functions in 2-D. If we examine the IQ basis functions we find that these can be expressed in terms of truncated Taylor expansions with simple explicit remainder terms.

\[
\phi(||x - x_i||) = \frac{1}{1 + \epsilon^2((x - x_i)^2 + (y - y_i)^2)} 
= 1 - \epsilon^2((x - x_i)^2 + (y - y_i)^2) + \epsilon^4 \frac{((x - x_i)^2 + (y - y_i)^2)^2}{1 + \epsilon^2((x - x_i)^2 + (y - y_i)^2)} 
= 1 - \epsilon^2((x - x_i)^2 + (y - y_i)^2) \epsilon^4((x - x_i)^2 + (y - y_i)^2)^2 
- \epsilon^6 \frac{((x - x_i)^2 + (y - y_i)^2)^3}{1 + \epsilon^2((x - x_i)^2 + (y - y_i)^2)} 
= \ldots 
\] (2.1)

The idea then was to replace the original basis functions with linear combinations in the form

\[
\psi_1(x) = a_1 \phi(||x - x_1||) + a_2 \phi(||x - x_2||) + \ldots + a_7 \phi(||x - x_7||), 
\] (2.2)

etc., where \(x_2, \ldots, x_7\) are neighboring nodes to \(x_1\). Given this linear combination, we could choose the coefficients \(a_1, \ldots, a_7\) so that

\[
\sum a_i = \sum a_i x_i = \sum a_i y_i = \sum a_i x_i^2 = \sum a_i x_i y_i = \sum a_i y_i^2 = 0, 
\] (2.3)

which is the equivalent to finding a null vector of the matrix

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 & 2 & 2 & 1 \\
\end{bmatrix}
\] (2.4)

Given that this matrix has one more column than rows, we are assured to have at least one null vector. Also, because our new basis function is a linear combination of the original we
remain in exactly the same approximation space as said original RBF interpolant. Since we know analytically from equation 2.3 that
\[
\sum_{i=1}^{7} a_i (1 - \epsilon^2 ((x - x_i)^2 + (y - y_i)^2)) = 0 \quad (2.5)
\]
we can write our new basis function \( \psi_1(x) \) as
\[
\psi_1 = \epsilon^4 \sum_{i=1}^{7} \frac{a_i ((x - x_i)^2 + (y - y_i)^2)^2}{1 + \epsilon^2 ((x - x_i)^2 + (y - y_i)^2)}, \quad (2.6)
\]
thus allowing us to analytically factor out the \( \epsilon^4 \) term using a combination of 7 of the original basis functions.

Initial results showed that this exchange of basis functions could provide conditioning improvement in some cases. Using Halton node locations in two dimensions we plot the condition number for number of nodes \( N = 20, 22, 25, \) and 30. We let \( M \) represent the cutoff in node number where we switch from the original \( \phi(||x - x_i||) \) to the new basis function \( \psi(||x - x_i||) \) in that we use \( \phi(||x - x_i||) \) for \( i = 1, 2, ..., M \) and \( \psi(||x - x_i||) \) for \( i = M + 1, M + 2, ..., N \). The results are displayed below in figure 2.1

The improvements we did find showed that there may be potential for a method of this form. The lack of consistency however showed that something was missing. The missing piece was in the form of the counting problem in the number of terms of each power of \( \epsilon \) in an RBF interpolant. In this case we needed 7 terms in order to factor out a power of \( \epsilon^4 \) whereas from (Table 1.2) we see that in order to "keep up" with the terms we would need to be able to factor out the \( \epsilon^4 \) term from our expansion using only 4 original basis functions. Simply put, the counting was off. The answer, as it turned out, was to explore a different basis function altogether.

2.3 Counting Problem Solved

Here we propose another method for calculating the interpolant in two dimensions without either limitation from the previous methods. To begin, we must revisit the conditioning
Figure 2.1: A plot of the condition number as a function of $M$ for $N=20$, 22, 25, and 30, $\epsilon = 0.1$ for all cases. When $M = N$ here we get the condition number for the original RBF interpolant. Only in the cases $N=22$ and 25 do we see a substantial improvement in conditioning when $M = 6$ (thus the first 6 basis functions are the original RBF and the remainder use the new method presented in equation 2.2.)
issue which creates the small epsilon problem. We will focus our attention on the Gaussian basis function \( \phi(r, \epsilon) = e^{-(\epsilon r)^2} \). This basis function has the advantage of super-spectral accuracy. If we apply the RBF-Direct approach our system takes the form

\[
I(x, \epsilon) = \sum_{i=1}^{N} \lambda_i \phi(\|x - x_i\|) \quad (2.7)
\]

for scattered data \( \{x_i, f_i\}, i = 1 \) to \( N \)

Computation of the expansion coefficients \( \lambda_i \) consists of solving the linear system \( A \lambda = f \) where \( A \) has elements \( A_{ij} = \phi(\|x_i - x_j\|) \) and the vectors \( \lambda \) and \( f \) contain the individual values respectively, cf. equation (1.2). This \( A \) matrix is positive definite and contains one eigenvalue of \( O(1) \), two eigenvalues of \( O(\epsilon^2) \), three of \( O(\epsilon^4) \), four of \( O(\epsilon^6) \), etc until all \( N \) eigenvalues are accounted for. Thus the condition number of the \( A \) matrix grows rapidly with small values of \( \epsilon \) and can be expressed directly as:

\[
\text{cond}(A) = O(\epsilon^{-2[\sqrt{8N-7}-1]/2}) \quad (2.8)
\]

with \([x]\) being the integer part \([9]\).

In solving this linear system directly we can take two approaches. The first is high precision numerical calculation. This has the distinct drawback of having a very hefty computational cost. The second is through regularization of the \( A \)-matrix such as taking the SVD, suppressing the smallest singular values, and then calculating the \( \lambda \) elements. This second method, however, costs a great deal of accuracy in our interpolation as we are deliberately removing information from the system.

Simply put, as soon as we have formed the \( A \)-matrix using this direct method we have created the conditioning problem. The solution must thus be found in a way which bypasses the creation of this form of the collocation matrix entirely.

2.4 RBF-GA

The RBF-QR method for nodes on a sphere solved the small \( \epsilon \) problem by expanding the basis functions themselves in terms of the spherical harmonics. By doing so, the powers
of $\epsilon$ in the eigenvalues can be factored out of the system analytically as they appear, thus making the collocation matrix have all eigenvalues of $O(1)$ [5].

Fundamentally, the RBF-QR method creates a new set of basis functions which span the exact same space as the originals. Other examples of this concept include the monomials $1,x,x^2,x^3,...$ and the Chebyshev polynomials $T_0(x), T_1(x), T_2(x),...$. Both sets span the exact same space, but the Chebyshev polynomials have properties which make them more desirable in certain situations. In our case, the original basis functions are the Gaussian radial basis functions which, in the direct method, become increasingly linearly dependent as the value of $\epsilon$ goes to zero. To solve this we need to construct a new set of functions which will span the exact same space, and thus have all the valuable properties of the Gaussian, but which do not become increasingly linearly dependent as $\epsilon$ decreases. There are many numerical methods for doing this, such as the Gram-Schmidt algorithm, but these methods require large amounts of numerical cancelation, leading to the loss of digits and thus causing a similar problem. The new approach must allow all the cancelations to be done analytically and thus be removed from the problem before the finite precision numerical calculations are even performed.

The Gaussian radial function in two dimensions can be written as the product of three terms as follows:

$$
\phi_i^{(1)}(x) = e^{-\epsilon^2((x-x_i)^2+(y-y_i)^2)} \\
= e^{-\epsilon^2(x_i^2+y_i^2)} \cdot e^{-\epsilon^2(x^2+y^2)} \cdot e^{2\epsilon^2(x_i+y_i)} \tag{2.9}
$$

Here, the first term depends only on the location of the basis function and can be seen simply as a scalar multiplier and thus can be dropped from each basis function while remaining in the same spanned space. The second is independent of the node location, and thus a constant multiplier between each basis function, allowing it to be factored out. This final term is the only one which depends on both the basis function and position. It can be expanded as a Taylor series in 2-D as
\[
\phi_{1}^{(2)}(x) = e^{-\epsilon^2(x^2 + y^2)}.
\]

\[
\{1 + \frac{1}{1!}[2\epsilon^2(xx_i + yy_i)]^1 + \ldots + \frac{1}{(n-1)!}[2\epsilon^2(xx_i + yy_i)]^{n-1} + G_n(z)\}
\]

where \(z = 2\epsilon^2(xx_i + yy_i) = 2\epsilon^2 \cdot x_i\), and

\[
G_n(z) = e^z - \sum_{j=1}^{n-1} \frac{z^j}{j!}
\]

\[
= \frac{e^z}{(n-1)!} \int_0^z e^{-t} t^{n-1} dt
\]

\[
= e^z \cdot \text{gammainc}(z, n)
\]

where "gammainc" is the Matlab function for the incomplete gamma function. It is important to note that this provides us with a closed form expression for the remainder term in the Taylor expansion. This is a key property of the new algorithm.

We notice here the number of distinct functions of \(x\) and \(y\) associated with each order of \(\epsilon\): 1 of \(O(\epsilon^0)\), 2 of \(O(\epsilon^2)\), 3 of \(O(\epsilon^4)\), etc, exactly matching the rate at which the eigenvalues appear in the collocation matrix. This matchup provides a means of removing all the ill-conditioning. From this expansion we can form a new set of basis functions as linear combinations of the old ones (and thus remaining in exactly the same spanned space) in such a way that these leading terms can be canceled out analytically rather than numerically. Our new set of basis functions is constructed in the following way:

\[
[\psi_1(x)] = e^{-\epsilon^2(x^2 + y^2)} \cdot \frac{1}{\epsilon^0} [C_0][G_0(2\epsilon^2 \cdot x_1)],
\]

\[
\begin{bmatrix}
\psi_2(x) \\
\psi_3(x)
\end{bmatrix}
= e^{-\epsilon^2(x^2 + y^2)} \cdot \frac{1}{\epsilon^2} 
\begin{bmatrix}
C_1 & \cdots \\
\cdots & \cdots \\
G_1(2\epsilon^2 \cdot x_1) & \cdots \\
G_1(2\epsilon^2 \cdot x_2) & \cdots \\
G_1(2\epsilon^2 \cdot x_3)
\end{bmatrix},
\]

where

\[
C_0 = \frac{1}{(n-1)!} \int_0^z e^{-t} t^{n-1} dt
\]

\[
G_n(z) = e^z \cdot \text{gammainc}(z, n)
\]

\[
[\psi_1(x)] = e^{-\epsilon^2(x^2 + y^2)} \cdot \frac{1}{\epsilon^0} [C_0][G_0(2\epsilon^2 \cdot x_1)],
\]

\[
\begin{bmatrix}
\psi_2(x) \\
\psi_3(x)
\end{bmatrix}
= e^{-\epsilon^2(x^2 + y^2)} \cdot \frac{1}{\epsilon^2} 
\begin{bmatrix}
C_1 & \cdots \\
\cdots & \cdots \\
G_1(2\epsilon^2 \cdot x_1) & \cdots \\
G_1(2\epsilon^2 \cdot x_2) & \cdots \\
G_1(2\epsilon^2 \cdot x_3)
\end{bmatrix},
\]

where

\[
C_0 = \frac{1}{(n-1)!} \int_0^z e^{-t} t^{n-1} dt
\]

\[
G_n(z) = e^z \cdot \text{gammainc}(z, n)
\]
\[
\begin{bmatrix}
\psi_4(x) \\
\psi_5(x) \\
\psi_6(x)
\end{bmatrix} = e^{-\epsilon^2(x^2+y^2)} \cdot \frac{1}{\epsilon^4} \begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & C_2 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot 
\end{bmatrix}, \quad \epsilon 
\]

etc., where in these cases \( C_0 = [1], \ C_1 = \left[ \text{null} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \right]^T \),

\[
C_2 = \left[ \begin{array}{c}
\text{null} \\
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
y_1 & y_2 & y_3 & y_4 & y_5 & y_6
\end{array} \right]^T, \quad \text{etc.}
\]

Here the "null" represents the Matlab null command to find the null-space of the given matrix. Thus these "C" matrices ensure the analytical cancelations of the leading \( \epsilon \) terms from the Taylor series expansion. We know that they have at least one null vector since the number of columns is greater than the number of rows. Taking one more step in the algorithm shows the pattern still more clearly:

\[
C_3 = \left[ \begin{array}{c}
\text{null} \\
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} \\
y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 & y_{10} \\
x_1^2 & x_2^2 & x_3^2 & x_4^2 & x_5^2 & x_6^2 & x_7^2 & x_8^2 & x_9^2 & x_{10}^2 \\
x_1y_1 & x_2y_2 & x_3y_3 & x_4y_4 & x_5y_5 & x_6y_6 & x_7y_7 & x_8y_8 & x_9y_9 & x_{10}y_{10} \\
y_1^2 & y_2^2 & y_3^2 & y_4^2 & y_5^2 & y_6^2 & y_7^2 & y_8^2 & y_9^2 & y_{10}^2
\end{array} \right]^T
\]

Figures 2.2 and 2.3 show a comparison between the old and the new basis functions. As \( \epsilon \) decreases, the original Gaussian functions get flatter and flatter, becoming nearly constant
valued over a given domain. The new functions, however, look very much like a polynomial expansion set, one constant function, two linear, three quadratic, etc. We know analytically that they have the exact same spanning space as the Gaussian functions and thus they give the exact same desirable properties of that radial basis function, but with much better numerical conditioning on the system.

2.5 **Numerical Results for Interpolation**

Numerical calculations show a great amount of improvement over the RBF-Direct method. Figure 2.4 shows us a comparison between the standard matrix inversion RBF-Direct method and the new RBF-GA method. In these comparisons we use the test function:

\[ f(x, y) = \frac{59}{67 + (x + 1/7)^2 + (y - 1/11)^2} \]  

(2.16)

Nodes are chosen from the Halton sequence and are entirely contained within the unit circle. For these tests the nodes were listed in order of their distance from the origin. For the 91 node test we compare the results of RBF-GA to the results using high precision RBF-Direct in figure 2.5.

It is interesting to note that though the two interpolants appear to be near perfect matches for larger values of \( \epsilon \), the sup-norm difference of the two interpolants shows that there is an increasing difference as the value of \( \epsilon \) increases. There is evidence that the choice of node ordering can make a difference in the interpolation. We will discuss this in the next section.

2.6 **Polynomial vs. RBF-Generated Finite Difference Formulas**

The use of RBFs for the construction of finite difference weights, or stencils, has been clearly demonstrated to allow for the creation of alternative FD (Finite Difference) algorithms [19, 18, 21, 6]. Prior to the RBF techniques, FD had severe restrictions due to the requirement of structure grids, greatly complicating both approximations at curved boundaries, and
Figure 2.2: The Gaussian basis functions with $\epsilon = 10^{-4}$ centered at 15 nodes distributed over the domain $[-1, 1] \times [-1, 1]$. These basis functions are visually indistinguishable from each other, and from the constant function.
Figure 2.3: The basis functions generated using the RBF-GA algorithm using the same nodes as in figure 2.2. They are arranged into rows corresponding to the different levels from equations (2.12), (2.13), (2.14), and further. It should be noted that these functions look very similar to the 2-D polynomial basis functions of the same order.
Figure 2.4: Comparison of the RBF-GA to the RBF-Direct methods. On the left we have the absolute error values for the two methods using 21, 45, and 91 nodes. On the right we have sup-norm difference between the resulting interpolants the two methods produce.
Figure 2.5: The high precision RBF-Direct error in calculating the interpolant for the function defined in equation (1.3) using 91 Halton sequence generated node locations within the unit circle.
the use of local node refinements. Because the RBF interpolation technique requires no structured grid it has been used successfully to construct stencils that are similarly free of these restrictions.

The method we have developed in this research can also be used to improve the performance of finite difference stencil production in a similar way to its interpolation improvement. The creation of finite difference weights from a functional interpolation method is very similar to the direct interpolation itself. A finite difference stencil for some linear operator $L$ and for some set of node points $\{x_1, x_2, ..., x_n\}$ is a set of weights $\{w_1, w_2, ..., w_n\}$ such that for a given function $f$ at some point $x = x_c$,

$$Lf(x_c) \approx \sum_{i=1}^{n} w_i f(x_i) \quad (2.17)$$

These weights are generated by imposing that the interpolating function satisfy the equation for the point $x = x_c$. This can be achieved by solving the following linear system [6].

$$
\begin{bmatrix}
\phi_1(x_1) & \phi_1(x_2) & \cdots & \phi_1(x_n) \\
\phi_2(x_1) & \phi_2(x_2) & \cdots & \phi_2(x_n) \\
\vdots & \vdots & \ddots & \vdots \\
\phi_n(x_1) & \phi_n(x_2) & \cdots & \phi_n(x_n)
\end{bmatrix}
\begin{bmatrix}
w_1 \\
w_2 \\
\vdots \\
w_n
\end{bmatrix} =
\begin{bmatrix}
L\phi_1(x)|_{x=x_c} \\
L\phi_2(x)|_{x=x_c} \\
\vdots \\
L\phi_n(x)|_{x=x_c}
\end{bmatrix}
\quad (2.18)
$$

In this case the left hand side square matrix is exactly the transpose of the matrix used for the interpolation problem and thus can be generated in the exact same way. It is sometimes useful in calculating these weights to impose an additional constraint on the system in the form of a constant term or further constraints with additional polynomial terms. In this case the linear system takes the modified form of equation (2.19). Here we do not investigate the advantages of this as we are simply comparing the new method to RBF-Direct and so any
advantages given to it would carry over here.

\[
\begin{bmatrix}
\phi_1(x_1) & \phi_1(x_2) & \ldots & \phi_1(x_n) & 1 \\
\phi_2(x_1) & \phi_2(x_2) & \ldots & \phi_2(x_n) & 1 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\phi_n(x_1) & \phi_n(x_2) & \ldots & \phi_n(x_n) & 1 \\
1 & 1 & \ldots & 1 & 0
\end{bmatrix}
\begin{bmatrix}
w_1 \\
w_2 \\
\vdots \\
w_n \\
w_{n+1}
\end{bmatrix}
= 
\begin{bmatrix}
L\phi_1(x)|_{x=x_c} \\
L\phi_2(x)|_{x=x_c} \\
\vdots \\
L\phi_n(x)|_{x=x_c} \\
L1
\end{bmatrix}
\]

(2.19)

RBF-GA provides another significant advantage when used for calculating finite difference stencils. When calculating the values for the right hand side of equation (2.18) we find a very nice property of our RBF-GA basis functions. Equation (2.12) provides all derivatives of our new basis functions since it implies that

\[
\frac{d^p}{dz^p} G_n(z) = G_{\max(0,n-p)}(z), p = 0, 1, 2, \ldots
\]

(2.20)

We can calculate the weights generated by the method presented in this thesis compared to those generated using the finite precision RBF-Direct method and generated by a high precision RBF-Direct. All the following results are given for the linear operator \( L(f) = \partial f / \partial x \).

As can be seen from figures 2.6 - 2.9, the RBF-GA method generates a substantial improvement over standard precision RBF-Direct in accuracy. Figure 2.10 demonstrates that the RBF-GA method also gives a great improvement in computational speed over the high precision RBF-Direct.

It is important to note in figures 2.6 - 2.9 that, for large \( \epsilon \), the standard precision RBF-Direct actually generates better results than RBF-GA. The cause of this is can be identified when we examine the condition number for the system itself as a function of \( \epsilon \). Figure 2.11 shows the condition number for the 36 node stencil.

As this phenomena has not yet been fully explained we can continue to implement the RBF-GA method with a simple test of the system condition number compared to RBF-Direct prior to solving at only a small computational cost. However, as mentioned in the
Figure 2.6: The max norm error comparison between weights generated for $L$ using RBF-GA and standard precision RBF-Direct compared to high precision RBF-Direct using a 10 node stencil.
Figure 2.7: The max norm error comparison between weights generated for $L$ using RBF-GA and standard precision RBF-Direct compared to high precision RBF-Direct using a 15 node stencil.
Figure 2.8: The max norm error comparison between weights generated for $L$ using RBF-GA and standard precision RBF-Direct compared to high precision RBF-Direct using a 36 node stencil.
Figure 2.9: The max norm error comparison between weights generated for $L$ using RBF-GA and standard precision RBF-Direct compared to high precision RBF-Direct using a 105 node stencil.
Figure 2.10: The computation time in seconds for the three methods in generation of the finite difference stencils. Here RBF-VPA stands for Matlab’s Variable Precision Arithmetic using 200 digits of precision, the precision used for the previous error analysis plots.
Figure 2.11: The condition number for the 36 node stencil as a function of $\epsilon$. We immediately see that for large values of $\epsilon$ the condition number for the RBF-GA method grows out of control.
previous section, there is some evidence to the cause of this may lie partially in the ordering we choose for the nodes. Looking again at the 36 node stencil, the nodes were ordered based on their distance from the origin, a very structured ordering. The nodes themselves are located in an unstructured way using the Halton Sequence. Figure 2.12 shows the condition number for the same system if we are to order the nodes randomly (through the use of Matlab's "randperm" function). Here we can see that the condition number is unaffected for small $\epsilon$ but becomes significantly worse for larger $\epsilon$.

We have a near opposite effect when we change the node locations themselves. Performing the same condition number test using nodes located on a hexagonal lattice we see this difference in the larger $\epsilon$ realm. Figures 2.14 and 2.13 show that the random ordering produces improved conditioning over the distance ordering, the opposite of the results for the Halton nodes. In both of these cases, however, the $\epsilon$ realm where the ordering has any noticeable effect is in an area where RBF-Direct produces significantly better conditioning.

2.7 Concluding remarks

RBF-GA may turn out to not be an end all method. Computationally it is somewhat expensive, and it does not give a substantial improvement over other stable methods such as Contour Padé or RBF-QR. However, what we have obtained is an algorithm that is extremely simple to implement, and which has equally simple extensions to higher dimensions. The Taylor expansion of the Gaussian in 3-D, for example, leads to a similar counting matchup with eigenvalues of 3-D RBF’s as was demonstrated here with 2-D. Overall, as we continue to add more distinct algorithms that are simple, fast, robust, allow access to the small $\epsilon$ realm, etc., we gain further evidence of the existence of a possible "holy grail" algorithm for RBF interpolation. A newcomer in the field of interpolation, relatively speaking, RBF’s continue to show greater and greater potential for amazing improvement in many areas.
Figure 2.12: Condition number for the 36 node stencil system using a random ordering of nodes, again compared with RBF-Direct.
Figure 2.13: Condition number for the 36 hexagonally located node stencil system using a random ordering of nodes.
Figure 2.14: Condition number for the 36 hexagonally located node stencil system using a distance ordering of nodes.
References


