2009

Variable Shape Parameter Strategies in Radial Basis Function Methods

Derek Sturgill

Variable Shape Parameter Strategies in Radial Basis Function Methods

Derek Sturgill

Thesis submitted to the Graduate College of Marshall University in partial fulfillment of the requirements for the degree of

Master of Arts
in
Mathematics

Committee Chairman, Dr Scott Sarra
Committee member, Dr Ari Aluthge
Committee member, Dr Basant Karna

Department of Mathematics
Marshall University

Huntington, West Virginia
2009

Variable Shape Parameter Strategies in Radial Basis Function Methods

Derek Sturgill

ABSTRACT

The Radial Basis Function (RBF) method is an important tool in the interpolation of multidimensional scattered data. The method has several important properties. One is the ability to handle sparse and scattered data points. Another property is its ability to interpolate in more than one dimension. Furthermore, the Radial Basis Function method provides phenomenal accuracy which has made it very popular in many fields. Some examples of applications using the RBF method are numerical solutions to partial differential equations, image processing, and cartography. This thesis involves researching Radial Basis Functions using different shape parameter strategies. First, we introduce the Radial Basis Function method by stating its history and development in Chapter 1. Second, we explain how Radial Basis Functions work in Chapter 2. Chapter 3 compares RBF interpolation to polynomial interpolation. Chapters 4 and 5 introduce the idea of variable shape parameters. In these chapters we compare and analyze the variable shape parameters in one and two dimensions. In Chapter 6, we introduce the challenges in interpolations due to errors in boundary regions. Here, we try to reduce the error using different shape parameter

strategies. Chapter 7 lists the conclusions resulting from the research.

Throughout the thesis we use some abbreviations and acronyms, they are listed below in the Table 1.

Term	Meaning
ε	Shape parameter
$\phi(r)$	Radial Basis Function or RBF
κ, kappa, or $\kappa(B)$	Condition number of the system matrix
κ_C	Condition number of the system matrix for a constant ε
κ_E	Condition number of the system matrix for an exponential ε
κ_L	Condition number of the system matrix for a linear ε
κ_R	Condition number of the system matrix for a random ε
εAverage	Average value for the shape, ε
εMax	Maximum value for the variable shape for ε
εMin	Minimum value for the variable shape for ε
GA	Gaussian RBF
IMQ	Inverse multiquadric RBF
IQ	Inverse quadratic RBF
LI	Linear RBF
MQ	Multiquadratic RBF
TPS	Thin Plate Spline RBF

Table 1: Abbreviations and Acronyms.

The following notes are highlighted, in order to clarify how things are stated in the thesis:

Note: Matlab does not display scientific notation as we are used to seeing. For example, 1.2×10^2 is written by Matlab to be $1.2e^2$. Please be aware of this throughout the thesis. This different form of scientific notation is very common in numerical analysis papers. Do not mistake e to be $e \approx 2.71828183$.

Note: Starting at chapter 3, the reader will see the words

εMin, εMax, and εAverage. εMin, εMax, and εAverage correspond to the value associated with the shape parameter in the MQ RBF. The Multiquadric Radial Basis function **does not** use c as the shape parameter, but ε. The reader will see this notation (ε) in Table 3 that displays different radial basis functions.

Acknowledgement

It is a great privilege to thank certain individuals and groups that have helped me with my research.

First, let me thank Dr. Sarra for his knowledge and expertise in Radial Basis Functions, his support in time of need, correcting my errors, and his upmost drive to keep me on tract. I would, also, like to thank Marshall University for accepting me in the graduate program and helping me to develop into whom I am today. I would like to thank Mike Smith who helped me with a variety of problems.

I would like to thank Dr. Karna and Dr. Aluthge for being on my committee.

Finally, I would to thank my family. My Mom and Dad for providing me the opportunity to go to college and encouraging me to pursue my "passion" for learning. My Aunts and Uncles for providing a quiet place to write and research my thesis. I would like to thank my Nan for listening in times of stress and joy.

Thank you all for everything you have done.

Contents

List of Tables

List of Figures

Listings

1 Introduction to Radial Basis Functions

The Radial Basis Function approximation method is a generalized refinement of the multiquadric (MQ) method developed by R. L. Hardy in 1968, [18]. Hardy developed the multiquadric method for solving a problem in cartography. He needed a "satisfactory" continuous function, that given points few and far between could construct a topographic map with low errors. For example, he could use this method to draw a map of a rocky cliff side and guarantee its accuracy when compared to the true form. Why we used the term "satisfactory" was because it best approximated the function that could provide an exact fit of the data values. The method reproduced such abnormalities as valleys, drainage areas, hilltops, and cliffs. When Hardy started the creation of the multiquadric method, a skilled topographer would ultimately rely on his/her perspective to accurately decide how the surface should look like [19], even though there were available analytical methods that could determine the slope of the surface from one point to another. This reliance on perspective could lead to two very skilled topographers to envision and create two very different maps from the same data. The multiquadric method is an unbiased, automated method which could produce such a graph with very few errors.

The development of the multiquadric method was not used for topographic maps alone. The multiquadric method used in conjunction with analytic geometry and calculus to determine such things as vol-

umes of earth, valleys, mountaintops, and the distance along a curved line [18], [19].

The development of the topographic surface functions involved many types of interpolation functions. The first functions used polynomial interpolation and Fourier methods, although they were discovered to be unsatisfactory. Each method was found to have different faults. Polynomial interpolation was unsatisfactory because given few data points the method was unable to accurately represent the rapid variations in the topographic surface [18]. Fourier interpolation, also, had a problem with few data points. It was discovered that the Fourier series given few data points created a function that tended to oscillate too much between the data points [18]. Given these problems with limited data points, there was another problem with matrix singularity. Interpolation with the Fourier series and polynomials could be singular using Haar's Theorem. The Least-squares methods based on polynomial and Fourier series were also investigated for a topographic surface construction. It was later found out that the Least-squares method did not accurately provide a good fit of the data [18].

The collapse of the Fourier and polynomial series and the Least-squares methods for satisfactory functions that could represent a topographical surface helped lead Hardy to a new method which met the needs and surpassed the short comings of the other methods. Through trial and error, Hardy eventually discovered the multiquadric method.

The first step Hardy took was to investigate a one dimensional

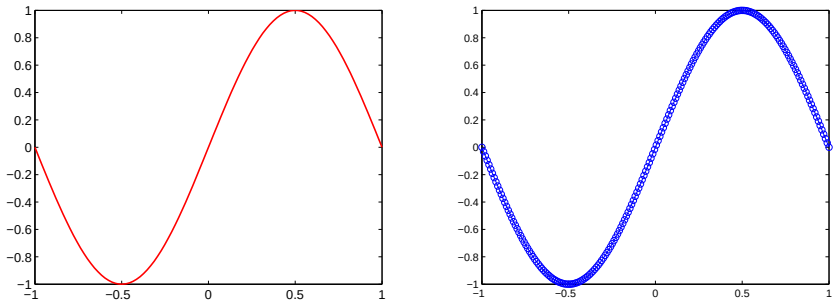

Figure 1: The figure on the left is $sin(\pi x)$, the figure on the right is the representation of $sin(\pi x)$ using the Multiquadric RBF method.

problem. Hardy wanted to create a satisfactory function that could represent a topographical curve. While studying this problem, Hardy discovered the data could be satisfactory represented by a piecewise linear interpolating function [19]. He proposed that given a set of n distinct scattered data points $\{x_j\}_{j=1}^n$ and corresponding measurements $\{f_j\}_{j=1}^n$, that the form of the interpolating function should look like:

$$s(x) = \sum_{j=1}^{n} \lambda_j |x - x_j|, \tag{1}$$

where the λ_j's are determined by the interpolation conditions; i.e. $s(x_j) = f_j,\ j = 1, 2, 3, \ldots, n$. Geometrically, this is interpolating the data by a linear combination of n translates of the absolute value function $|x|$, given that the vertex of $|x|$ is centered at one of the data points.

Hardy soon recognized the interpolating function, (1), accurately modeled a cliff. However, it did not find hilltops (maximums) and

valleys (minimums) since the function had a jump in the first derivative at each source point. Hardy figured out that this problem could be solved by removing the absolute value basis function in equation number (1) and replacing it with a function that is continuously differentiable. Hardy's function was $\sqrt{c^2 + x^2}$, where c is an arbitrary non-zero constant [19]. The new interpolation method became

$$s(x) = \sum_{j=1}^{n} \lambda_j \sqrt{c^2 + (x - x_j)^2}, \tag{2}$$

where again, the λ_j's are determined by the interpolation conditions. Note that for the above function, (2), if $c \neq 0$ this results in a continuously differentiable interpolant. However, if $c = 0$ function (2) is equivalent to the interpolating function (1). Relatively speaking, the same geometric interpretation held for the new method. The new method did not use the absolute value basis function; but instead, it uses a linear combination of translates of the hyperbolic basis function.

Hardy discovered the new method (2) could provide an accurate illustration of the topographic region, and the methods of calculus could be used. Using this new method, Hardy found the maximum and inflection points of the topographic region.

Hardy applied the new method to more than a one dimensional space. Note that the absolute value of the difference between two one-dimensional space is the Euclidean distance between the two points; for example, $|x - x_j| = \sqrt{(x - x_j)^2}$. Now he needed to create a in-

terpolating function of the form (2) to work in two dimensions. What Hardy created was an interpolating function based on translates of the Euclidean distance function in two dimensions. Hence, given n distinct scattered data points $\{x_j, y_j\}_{j=1}^n$ and corresponding topographic measurements $\{f_j\}_{j=1}^n$ for $\jmath = 1, 2, 3, \ldots, n$, Hardy proposed the following interpolation function

$$s(x, y) = \sum_{j=1}^{n} \lambda_j \sqrt{(x - x_j)^2 + (y - y_j)^2}, \tag{3}$$

where the λ_j's are again determined by interpolation; i.e. $s(x_j, y_j) = f_j$ for $\jmath = 1, 2, 3, \ldots, n$. Geometrically speaking, this method relates to interpolating the data by a linear combination of n number translates of a cone. To be exact $\phi(r) = \sqrt{x^2 + y^2}$. Also as previously described in one dimension, the vertex of each cone is centered at one of the data points. Again, Hardy ran into the same problem as his one-dimensional interpolating function. The problem was that function (3) suffered from being a piecewise continuous. He was unable to find a simple fix for this problem. Hardy proposed using a linear combination circular hyperboloid basis functions (rotated hyperbola basis functions $\sqrt{c^2 + x^2}$ translated to be centered at each source point). The new form of (3) is

$$s(x, y) = \sum_{j=1}^{n} \lambda_j \sqrt{c^2 + (x - x_j)^2 + (y - y_j)^2}. \tag{4}$$

We can apply the techniques in multivariable calculus for determining properties of the topographic surface, if $c \neq 0$.

Hardy discovered that the new function, (4), was an excellent method for approximating topographical information from sparse data points. Unlike the Fourier series, the new function did not suffer from large oscillations. Also, the function alleviated the problem associated with the polynomial series method (i.e. the polynomial series was unable to account for rapid variations of the topographical surface) [18]. Hardy named this new technique the "multiquadric method", because he considered the key features of this method to be a "superpositioning of quadric surfaces" [19].

Hardy first developed the the multiquadric method (4) only for solving two dimensional interpolation problems. He then realized he could extend the method to include interpolation for more than two dimensions. The RBF method, (4), depends on its Euclidean distance of the point (x, y) from its center (x_j, y_j). We could extend (4) to three dimensions by creating a basis function that depends on the Euclidean distance of the point (x, y, z) from its center (x_j, y_j, z_j). We can continue this "trend" to interpolate in any dimension. This lead to the following definition of the general multiquadric method for interpolation in any dimension:

Definition 1 "The Multiquadric(MQ) method"-Given a set of n distinct data points $\{x_j\}_{j=1}^{n}$ in R^d where each $x = x_1 + x_2 + \ldots + x_d$,

and corresponding data values $\{f_j\}_{j=1}^n$, the multiquadric interpolant of the distinct data points is given by the function,

$$s(x) = \sum_{j=1}^{n} \lambda_j \sqrt{c^2 + ||x - x_j||^2} \tag{5}$$

where $||\cdot||$ represents the Euclidean norm. The expansion coefficients, λ_j, are determined from the interpolation conditions $s(x_j) = f_j$, j $= 1, 2, 3, \ldots, n$, which leads to the following linear, symmetric system:

$$[B][\lambda] = [f] \tag{6}$$

where the entries of matrix B are given by the functions

$$b_{jk} = \sqrt{c^2 + ||x_j - x_k||^2}$$

.

After the publication of Hardy's paper [18], researchers of many different fields began using the multiquadric method. The MQ method is very successful in the fields of geology, geophysics, hydrology, geodesy photogrammetry and other fields [19]. There have been advances in fault classification in transmission lines and aero-engine designs using RBF's in references [1, 28]. Radial Basis Functions have also been used to estimate radiation and electromagnetic field problems in references [27, 30]. RBF's are found in the construction industry as well [2, 31, 40].

A very important advancement in the field of the multiquadric

method was done in 1974 by Richard Franke [14]. In this study, Franke compared a variety of methods that could be used for interpolating two dimensional scattered data. Franke compared 32 methods to see how they approximated 6 different test functions that were sampled on 3 different density levels [15]. The 6 different test functions Franke used generated surfaces for the comparison and were given the titles: "exponential test", "cliff test", "gentle test", "steep test", "saddle test", and "sphere test" [19]. The density levels included 25, 33, and 100 scattered data points on the 6 different test functions. The Multiquadric method provided the best approximations for 13 of 18 test, and was second best 3 times for the rest [15]. Table 2 shows a few results of Frank's experiment [15].

Method	Sensitivity	Complexity	Accuracy	Visual
Duchon's TPS	N	A	A	A
Franke's Method- 3	C	D	B	B
Franke's Method- TPS	C	D	B^+	B^+
Hardy's Multiquadrics	B	A	A	A
Hardy's Reciprocal MQ	B	A	A	A
Modified Linear Shepard	C	B	C	C

Table 2: Franke's Test of Interpolation Methods.

Though Franke provided numerical evidence that the MQ method was efficient for interpolation, he did not provide any mathematical foundation for it. It had not been shown that the method was uniquely solvable. Franke could not provide a sufficient proof that the matrix B in (6) was nonsingular. He could only show that the matrix B was

non-singular through extensive numerical experiments.

Later in 1986, Micchelli [29] provided a proof of Franke's conjecture. Micchelli provided sufficient conditions that could guarantee the nonsingularity of the method when other basis functions were used. For example, Micchelli's results guaranteed nonsingularity of Hardy and Göpfert basis functions [20]. Göpfert basis function is defined as $(c^2 + \|x\|)^{\frac{-1}{2}}$. Also Duchon's basis functions [8]: $\|x\|^2 \log \|x\|$ and $\|x\|^3$, fit Micchelli's conditions superbly.

Fifteen years after Hardy created the idea of fitting multidimensional scattered data with a linear combination of translates of the basis function $\sqrt{c^2 + \|x\|^2}$, Micchelli showed that the method was unconditionally nonsingular, and given certain conditions that the system matrix resulting from any RBF could be guaranteed to be nonsingular. Therefore, the multiquadric method was chosen to be the specific example from a variety of basis functions. The principle behind a more general method was to use translates of one basis functions that was dependent on the Euclidean distance from its center in order to create a multidimensional interpolant. Given this dependence, it implied that the function is "radial" symmetric about its center. Hence, became known as "radial basis functions" and this method became known as the RBF method [32].

1.1 Radial Basis Function Method

As we have stated before, the RBF method is one of the important tools for the interpolation of sparse, scattered, data points in multi-dimensions. Using this method, we have the ability to approximate an underlining function, and it has become very popular in a diversity of fields. In this thesis we used a "simple" form of the Radial Basis Function. By "simple" we mean the method is not augment with polynomials. Let it be known, that otherwise stated that x represents x_j in R^d where d is a positive integer, $||\cdot||$ represents the Euclidean norm, and f_j is a scalar value.

Definition 2 "Simple RBF Method"- Given a set of n distinct data points $\{x_j\}_{j=1}^n$ and corresponding data values $\{f_j\}_{j=1}^n$, the simple Radial Basis Function interpolant is produced by,

$$s(x) = \sum_{j=1}^{n} \lambda_j \phi(\|x - x_j\|) \tag{7}$$

where $\phi(\|x-x_j\|) = \phi(r)$, $r \geq 0$, is a radial function. λ_j, the expansion coefficients, are determined by the interpolant conditions $s(x_j) = f_j$ for j $= 1, 2, 3, \ldots, n$. The above produces the symmetric linear system,

$$[B][\lambda] = [f]. \tag{8}$$

The entries in matrix B are given by the formula $b_{ij} = \phi(\|x_i - x_j\|)$.

Micchelli [29] gave adequate conditions for $\phi(r)$ in (7) to ensure that the matrix B in (8) would be nonsingular; therefore, the Radial Basis Function method is uniquely solvable. Some examples of Radial Basis Functions are listed in Table 3 and Figure 2.

Name of RBF	Abbreviations	$\phi(r),\ r \geq 0$	Smoothness
Gaussian	GA	$e^{-(\varepsilon r)^2}$	Infinite
Generalized Multiquadric	GMQ	$(1+(\varepsilon r)^2)^{\beta}$	Infinite
Inverse Multiquadric	IMQ	$\frac{1}{\sqrt{1+(\varepsilon r)^2}}$	Infinite
Inverse Quadratic	IQ	$\frac{1}{1+(\varepsilon r)^2}$	Infinite
Multiquadric	MQ	$\sqrt{1+(\varepsilon r)^2}$	Infinite
Cubic	CU	r^3	Piecewise
Linear	LI	r	Piecewise
Monomial	MN	r^{2k-1}	Piecewise
Thin Plate Spline	TPS	$r^2 \log(r)$	Piecewise

Table 3: Radial Basis Function Table.

The variable ε in the infinitely smooth Radial Basis Functions is known as the shape parameter, which as it suggests, controls the shape of the functions. In the experiments and thesis, ε, is a non-zero real value. Notice how the Multiquadric differ from Hardy's definition (5), we have replaced c with $\frac{1}{\varepsilon}$. The replacement of c with $\frac{1}{\varepsilon}$ has become common in recent years [36].

Now we discuss the conditions for $\phi(r)$ that would guarantee the matrix B in (5) to be non-singular. The conditions are attributed to Schoenberg [37] in 1938. Micchelli later showed that Schoenberg's conditions could be relaxed to include many more functions.

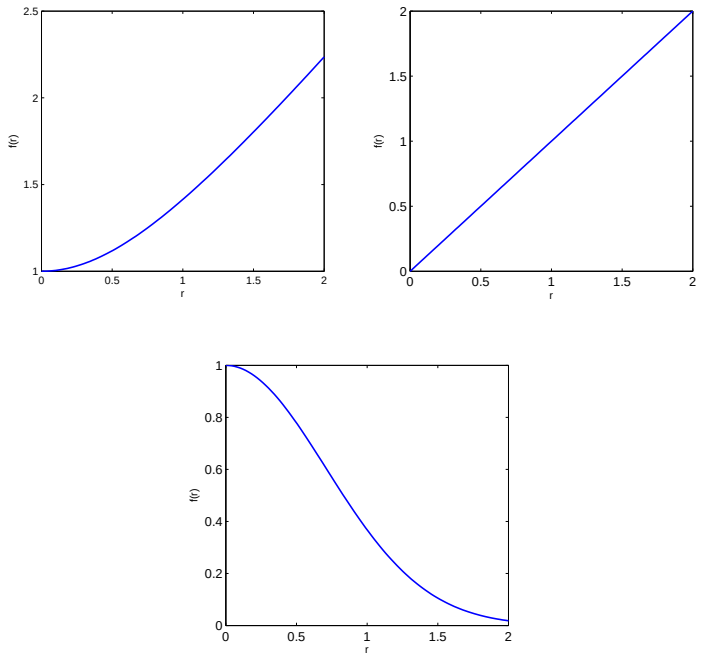

Figure 2: Graphs of some Radial Basis Functions (left to right: MQ, LI, GA) with $\varepsilon = 1$.

Theorem 1 Schoenberg [37]- If $\psi(r) = \phi(\sqrt{r})$ is 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 B with entries $b_{ij} = \phi(\|x_i - x_j\|)$ is positive definite and thus non-singular.

Definition 3 "Completely Monotone Function"- A function ψ, it is defined to be completely monotone on $[0, \infty)$ if the following conditions hold,

(1) $\psi \in C[0, \infty)$

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

(3) $(-1)^l \psi^l(r) \geq 0$ for $r > 0$ and $l = 0, 1, 2, 3, \ldots$

Later on Micchelli [29] extended Schoenberg's theorem.

Theorem 2 Micchelli [29]- Let $\psi(r) = \phi(\sqrt{r}) \in C[0,\infty)$, $\psi(r) > 0$ for $r > 0$, and $\psi'(r)$ be 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 B with entries $b_{ij} = \phi(\|x_i - x_j\|)$ is non-singular. Furthermore, for $n \geq 2$, the matrix B has $n-1$ negative eigenvalues and one positive eigenvalue.

Next we explore the Multiquadric radial basis function to see if it fits Micchelli's theorem above. For the Multiquadric radial basis function $\psi(r) = \phi(\sqrt{r})$ is defined to be $\sqrt{1+(\varepsilon r)^2}$ for $r \geq 0$. Since r is defined to be greater than or equal to zero, we write the above equation to be $\psi(r) = \sqrt{1+\varepsilon^2 r}$.

$$\begin{aligned}
\psi(r) &= \sqrt{1+\varepsilon^2 r} \\
\psi'(r) &= \frac{\varepsilon^2}{2\sqrt{1+\varepsilon^2 r}} \\
\psi^{(2)}(r) &= \frac{-\varepsilon^4}{4(1+\varepsilon^2 r)^{\frac{2}{3}}} \\
\psi^{(3)}(r) &= \frac{3\varepsilon^6}{8(1+\varepsilon^2 r)^{\frac{5}{2}}} \\
\vdots &= \vdots
\end{aligned}$$

$$\psi^{(\ell)}(r) \;=\; \frac{\Gamma(\ell-\frac{1}{2})\varepsilon^{2\ell}(-1)^{\ell-1}}{2\sqrt{\pi}(1+\varepsilon^2 r)^{\ell-\frac{1}{2}}} \qquad \text{for } \ell = 1,\ 2,\ 3, \ldots$$

and thus we have,

$$(-1)^l\psi^l(r) \;=\; \frac{\Gamma(\ell-\frac{1}{2})\varepsilon^{2\ell}}{2\sqrt{\pi}(1+\varepsilon^2 r)^{\ell-\frac{1}{2}}} \qquad \text{for } \ell = 1,\ 2,\ 3, \ldots \text{ [34]}$$

The Gamma function, Γ, is non-negative since it is a generalization of factorials to the non-integer values. Hence, $\psi'(r)$ is completely monotone and fits Theorem 2 of Micchelli. Thus the MQ system matrix is invertible. Likewise, the above can be shown for the IQ and IMQ basis functions. For the IQ and IMQ basis functions $\psi(r) = e^{-(\varepsilon^2 r)}$ and $\psi(r) = \frac{1}{1+\varepsilon^2 r}$ respectively thus,

$\psi(r) = e^{-(\varepsilon^2 r)}$	$\psi(r) = \frac{1}{1+\varepsilon^2 r}$
$\psi'(r) = -\varepsilon^2 e^{-(\varepsilon^2 r)}$	$\psi'(r) = \frac{-\varepsilon^2}{(1+\varepsilon^2 r)^2}$
$\psi^{(2)}(r) = \varepsilon^4 e^{-(\varepsilon^2 r)}$	$\psi^{(2)}(r) = \frac{2\varepsilon^4}{(1+\varepsilon^2 r)^3}$
$\psi^{(3)}(r) = -\varepsilon^6 e^{-(\varepsilon^2 r)}$	$\psi^{(3)}(r) = \frac{-6\varepsilon^6}{(1+\varepsilon^2 r)^4}$
$\psi^{(4)}(r) = \varepsilon^8 e^{-(\varepsilon^2 r)}$	$\psi^{(4)}(r) = \frac{24\varepsilon^8}{(1+\varepsilon^2 r)^5}$
$\vdots = \vdots$	$\vdots = \vdots$
$\psi^{(\ell)}(r) = \varepsilon^{(2\ell)} e^{-(\varepsilon^2 r)}(-1)^\ell$	$\psi^{(\ell)}(r) = \frac{(-1)^\ell \varepsilon^{(2\ell)}(\ell!)}{(1+\varepsilon^2 r)(\ell+1)}$

$$\text{for } \ell = 0\ ,1\ ,2\ ,3\ , \ldots$$

and thus we have,

$$(-1)^\ell\psi^{(\ell)}(r) = \varepsilon^{2(\ell)}e^{-(\varepsilon^2 r)} \quad (-1)^\ell\psi^{(\ell)}(r) = \frac{\varepsilon^{2(\ell)}(\ell!)}{(1+\varepsilon^2 r)^{(\ell+1)}}$$

We see that for the IQ and IMQ basis functions, the above quantities are greater than or equal to 0 for $\ell = 0,\ 1,\ 2,\ 3, \ldots$ Applying Micchelli's theorem, we see that the IQ and IMQ system matrices are invertible.

In conclusion, Theorems 1 and 2 state the needed conditions which guarantees that the "simple" Radial Basis Function method is uniquely solvable. In this thesis we use the Multiquadric method, unless we have have stated otherwise.

2 How Radial Basis Function Approximation works

2.1 Radial Basis Function Interpolation

First, we rephrase the definition of the Multiquadric Interpolation.

Definition 4 "Multiquadric Interpolation of Scattered Data"- Given scattered data (x_j, f_j) for $j = 1,\ 2,\ 3,\ \ldots,\ n$ and where $x_j \in R^d$ and $f_j \in R$, find an infinitely differentiable function, $s(x)$, such that $s(x_j) = f_j$ for $j = 1,\ 2,\ 3,\ \ldots,\ n$ [35]

The MQ method produces the function $s(x)$ by using linear combinations of translations of one of the infinitely differentiable functions $\phi(r)$. Given a set of centers $x_1^c, x_2^c, x_3^c, \ldots, x_n^c$ in R^d, the MQ interpolant takes the form of

$$s(x) = \sum_{j=1}^{n} \lambda_j \phi(\|x - x_j^c\|_2, \varepsilon) \tag{9}$$

where

$$r = \|x\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_d^2}. \tag{10}$$

The expansion coefficents λ_j's are determined by the interpolation condition

$$s(x_j) = f_j \text{ for } j = 1,\ 2,\ 3\ \ldots,\ n \tag{11}$$

at a set of nodes that usually coincide with the centers, $x_1^c,\ x_2^c,\ x_3^c,\ \ldots,\ x_n^c$.

After applying the interpolation conditions we have the following linear system,

$$B\lambda = f \tag{12}$$

where again the λ's need to be determined. Matrix B in (12) is an $n \times n$ matrix with entries, (**Note:** For simplicity reasons in the later Matlab code and further text **n** is now referred to as ***N***.)

$$b_{ij} = \phi(\|x_i^c - x_j^c\|_2) \text{ where } i, j = 1,\ 2,\ 3,\ \ldots,\ N \tag{13}$$

The $N \times N$ matrix B is called the interpolation matrix or the system matrix [35]. The b_{ij} entries of the system matrix are made up of functions which approximate the space evaluated between the center locations. To evaluate the interpolant at M number of points, x_i, we use (9) to create a $M \times N$ evaluation matrix. The evaluation matrix [35] is denoted by H. The matrix H contains elements of form

$$h_{ij} = \phi(\|x_i - x_j^c\|_2) \text{ where } i = 1, \ldots, M \text{ and } j = 1, \ldots, N. \tag{14}$$

The interpolant evaluated a M number of points is,

$$f_{approximate} = HB^{-1} f_{exact} = H\lambda. \tag{15}$$

To better understand the MQ RBF method, we explore it through some examples. The Matlab code used to produce the result for Examples 1 and 2 is listed in Listing 1. For Example 2, we only modified

the code from Example 1.

Listing 1: InterpolationExample1Dimensional.m

```
%MQ RBF interpolation.
%Modified version of Dr. Scott Sarra code
%Can be further modified to meet different requirements below
clear, home

   xc = [0; 0.5; 1];% vector of centers
    x = [0; 1/5; 2/5; 3/5; 4/5; 1];% vector of evaluation points
   xp = linspace(0,1);% 100 evenly spaced points for 0 to 1

        f = exp(sin(pi*xc));
   fExact = exp(sin(pi*x));

    shape = 3;% shape parameter

    B = systemMatrix(xc,shape);% creates system matrix
    H = evaluationMatrix(xc,x,shape);% creates evaluation matric

    lambda = B\f;% solves for lambda

    fApprox = H*lambda;% approximation of the true function

    format long, format compact
    pointWiseErrors = abs(fApprox - fExact)% error calculation
    fApproximation = fApprox% list the values of the approximate
                            % function
    fExactTrue = fExact% list exact function values
    format % reset the formats to default
    %plotting is below
    plot(xp,exp(sin(pi*xp)),'g',xc,f,'r*',x,fApprox,'ko')
    xlabel 'x', ylabel 'f(x)'
```

Example 1 MQ RBF Interpolation: Let $f(x) = e^{sin(\pi x)}$ be the function that we are interpolating. We interpolate $f(x)$ on the interval $[0, 1]$ using the center locations: $x_1^c = 0,\ x_2^c = .5,\ x_3^c = 1$. We evaluate the interpolant at the following points: $x_1 = 0,\ x_2 = \frac{1}{5},\ x_3 = \frac{2}{5},\ x_4 = \frac{3}{5},\ x_5 = \frac{4}{5},\ x_6 = 1$. The shape parameter is taken to be 3.

From here we need to determine the unknown RBF expansion coefficients denoted λ, the system matrix B, and the evaluation matrix H. Here, $f = [f(x_1^c)\ \ f(x_2^c)\ \ f(x_3^c)]^T = [f(0)\ \ f(.5)\ \ f(1)]^T = [0\ \ 2.25\ \ 4]^T$. The 3×3 system matrix was [35],

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

After we have obtained B, we can solve the expansion coefficients λ and get $\lambda = [1.668483972787157\ \ -3.297522687714909$ $1.668483972787156]^T$. After we found the expansion coefficients, we need to form the evaluation matrix . For this example, H is a 6×3 matrix of the form below [35].

$$H = \begin{bmatrix} \phi(\|x_1 - x_1^c\|_2) & \phi(\|x_1 - x_2^c\|_2) & \phi(\|x_1 - x_3^c\|_2) \\ \phi(\|x_2 - x_1^c\|_2) & \phi(\|x_2 - x_2^c\|_2) & \phi(\|x_2 - x_3^c\|_2) \\ \phi(\|x_3 - x_1^c\|_2) & \phi(\|x_3 - x_2^c\|_2) & \phi(\|x_3 - x_3^c\|_2) \\ \phi(\|x_4 - x_1^c\|_2) & \phi(\|x_4 - x_2^c\|_2) & \phi(\|x_4 - x_3^c\|_2) \\ \phi(\|x_5 - x_1^c\|_2) & \phi(\|x_5 - x_2^c\|_2) & \phi(\|x_5 - x_3^c\|_2) \\ \phi(\|x_6 - x_1^c\|_2) & \phi(\|x_6 - x_2^c\|_2) & \phi(\|x_6 - x_3^c\|_2) \end{bmatrix}$$

After we form H, we will evaluate the interpolant as $H\lambda$. Output from this example is listed in Table 4.

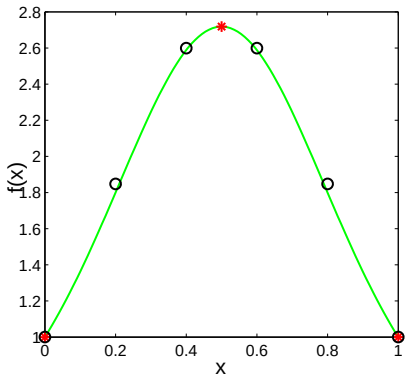

Figure 3: Graph of Example 1. The red asterisks are the exact function value at the center locations, the open black circles are the approximate function values at the evaluation points, and the green line is $f(x) = e^{sin(\pi x)}$ where $x \in [0, 1]$.

Clearly the numerical evidence listed in Table 4 and the Graph 3, using the MQ RBF to interpolate $f(x) = e^{sin(\pi x)}$ was a satisfactory choice. The highest point-wise error was $4.7468e^{-2}$ and the lowest

$f(x)_{exact}$	$f(x)_{approximate}$	Point-wise errors
1.000000000000000	1.000000000000002	0.000000000000002
1.799997457304433	1.847465233053820	0.047467775749387
2.588442947332867	2.599159299556487	0.010716352223620
2.588442947332867	2.599159299556487	0.010716352223620
1.799997457304434	1.847465233053819	0.047467775749386
1.000000000000000	1.000000000000002	0.000000000000002

Table 4: Data of MQ RBF Interpolation Example 2.1.1.

point-wise error was $2e^{-15}$ for the given six evaluation points. We see from Table 4 that the error in the approximation for the first and last point is zero. The reason is that the first and last center and evaluation points coincide. From the interpolation conditions $s(x_i^c) = f(x_i)$ should be exact. We try another example of interpolating with the MQ RBF to further understand the mechanics of its use.

Example 2 MQ RBF Interpolation: Let $f(x) = x^2+2x+1$ be the function that we are interpolating. We interpolate $f(x)$ on the interval $[-1, 1]$ using the center locations: $x_1^c = -1,\ x_2^c = -\frac{1}{3},\ x_3^c = \frac{1}{3},\ x_4^c = 1.$ We evaluate the interpolant at the following points: $x_1 = -1,\ x_2 = -\frac{1}{2},\ x_3 = 0,\ x_4 = \frac{1}{2},\ x_5 = 1.$ The shape parameter, ε, is again taken to be 3.

We have

$$f = [f(x_1^c)f(x_2^c)f(x_3^c)f(x_4^c)]^T = [f(-1)f(-\frac{1}{3})f(\frac{1}{3})f(1)]^T = [0\ \frac{4}{9}\ \frac{16}{9}4]^T$$

, and the 4×4 system matrix, B, is

$$B = \begin{bmatrix} 1 & 2.23606797749979 & 4.12310562561766 & 6.08276253029822 \\ 2.23606797749979 & 1 & 2.23606797749979 & 4.12310562561766 \\ 4.12310562561766 & 2.23606797749979 & 1 & 2.23606797749979 \\ 6.08276253029822 & 4.12310562561766 & 2.23606797749979 & 1 \end{bmatrix}$$

After we have obtained B, we solve for the expansion coefficients λ in equation (12). Solving equation (12) for λ, we have

$$\lambda = [0.48\ 0.1351\ 0.4184\ -0.4122]^T$$

. After creating the system matrix and solving for the expansion coefficients, our next step is to create the evaluation matrix H. H for this example was a 5×4 matrix. The numerical form of the evaluation matrix is,

$$H = \begin{bmatrix} 1 & 2.23606797749979 & 4.12310562561766 & 6.08276253029822 \\ 1.80277563773199 & 1.11803398874990 & 2.69258240356725 & 4.60977222864644 \\ 3.16227766016838 & 1.41421356237310 & 1.41421356237310 & 3.16227766016838 \\ 4.60977222864644 & 2.69258240356725 & 1.11803398874990 & 1.80277563773199 \\ 6.08276253029822 & 4.12310562561766 & 2.23606797749979 & 1 \end{bmatrix}$$

After we have formed H, we interpolate the given function. The approximation of the function is simply $H\lambda$. Data for this interpolation is in Table 5.

Again we see that using a MQ RBF interpolant to interpolate $f(x) = x^2 + 2x + 1$ where $x \in [-1, 1]$ produced very satisfactory results. The smallest point-wise error is 0 at x^c and the largest is approximately

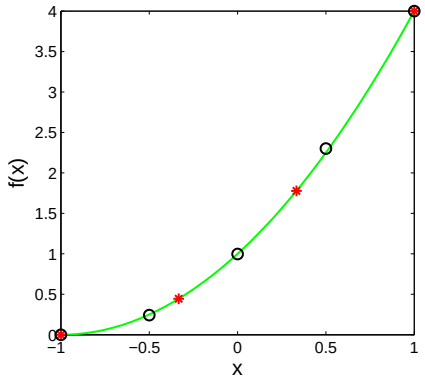

Figure 4: Graph of Example 2. The red asterisks are the exact function value at the center locations, the open black circles are the approximate function values at the evaluation points, and the green line is $f(x) = x^2 + 2x + 1$ where $x \in [-1, 1]$.

$f(x)_{exact}$	$f(x)_{approximate}$	Point-wise errors
0	0	0
0.25	0.242842685092608	0.007157314907392
1	0.997116820836758	0.002883179163242
2.25	2.301027980537274	0.051027980537274
4	3.999999999999999	0.000000000000001

Table 5: Data of MQ RBF Interpolation Example 2.1.2.

0.05103. Notice in Graph 4 that if the center locations and the evaluation points did not directly coincide that this did not drastically increase the point-wise errors. However, if all the evaluation points and center points coincide the point-wise error is 0.

2.2 Properties of the System Matrix

For the solution to exist in equation (15), the system matrix B must be invertible. Micchelli [29], provided a theorem that guarantees that

B is invertible. The theorem and justification of MQ system matrix being invertible is in Section 1.1.

2.3 Shape Parameter ε

The shape parameter, ε, in the MQ RBF is not a variable that can be selected arbitrarily. The shape parameter is crucial to the interpolation of a given function. There are consequences one needs to consider when choosing a shape parameter. A large shape parameter results in a well conditioned system matrix; however, the approximation using the RBF is poor. If one chooses to use a small shape parameter this results in a very accurate RBF approximation, but now the system matrix is ill-conditioned. Using a large shape parameter produces accuracy in approximations that is very similar to local methods such as the cubic spline and the finite difference methods [35]. If the shape parameter becomes too small the ill-conditioned matrix causes errors in floating point arithmetic, which results in very poor accuracy. This trade-off between accuracy and conditioning is known as the uncertainty principle [35].

Definition 5 "flat limit"- The term, flat limit, describes the limit as the shape parameter approaches 0 [35].

The "flat limit" RBF approximation has been shown to be equivalent to global polynomial approximation in one dimension and higher dimensions [35]. In global polynomial methods the interpolation sites

must be chosen carefully, because it results in ill-conditioning of the system matrix and the Runge Phenomenon [13]. If the center locations

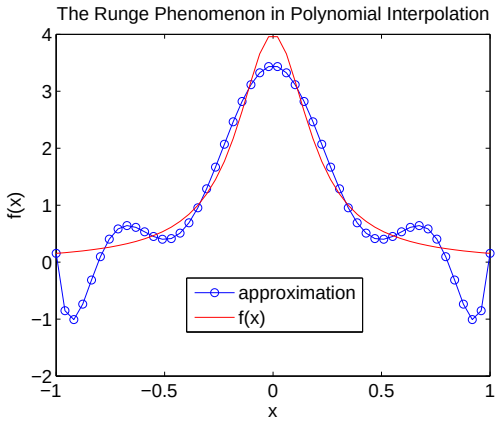

Figure 5: An example of the Runge phenomenon. For low number of centers(N) we see an oscillation in the approximation of $f(x)$.

coincide with the "carefully" chosen interpolation sites, then it can be said that the two methods are equivalent. Given this fact, the MQ methods are considered to be a generalized global polynomial methods. The "flat limit" is well known and there is much information [5, 7] concerning it, however this cannot be said about $\varepsilon > 0$. Shape parameters that are greater than zero produce much more versatile interpolants than global polynomial methods [35]. These versatile interpolation methods have many new features, which are not completely understood. For this reason, research is very active in the RBF area.

2.4 Condition Number and The Uncertainty Principle

Definition 6 "Uncertainty Principle- Phrase used to describe the fact that a RBF approximant can not at the same time be accurate and well conditioned.

In a perfect world we would want the condition number to be small, but many times this is very unlikely. Even the best written numerical algorithms will have difficulty approximating an ill-conditioned problem. Errors in input data or in the early stages of a computation may make the interpolation error grow uncontrollably.

The condition number in the 2-norm of a linear system $B\lambda = f$ is defined as follows,

$$\kappa(B) = \|B\|_2 \|B^{-1}\|_2 = \frac{\sigma_{max}}{\sigma_{min}} \tag{16}$$

where σ denotes the singular values of B. The system matrix in the MQ method is symmetric, hence $\sigma = |\lambda|$ and all eigenvalues λ are real. By Micchelli [29]; we know there exists $N-1$ negative eigenvalues and 1 positive eigenvalues for $N \geq 2$.

3 Comparison of Interpolation

3.1 RBF and Polynomial Interpolation Comparison

Let us start with an example involving Matlab, so we can further understand polynomial interpolation.

Example 3 Polynomial Interpolation: Use a global polynomial to interpolate the function $f(x) = \frac{1}{1+25x^2}$, where $x \in [-1, 1]$. Here we use the Matlab code **polyInter.m** where it requires the inputs N and M. N represents the number of centers and M is number of evenly spaced evaluation points. The open blue circles represent M in Figure 6.

Figure 6 shows the exact value of *f(x)*, displayed by the red line. The polynomial interpolation of *f(x)* is the blue line. The number of centers are 20 and the number of evaluation points are 50.

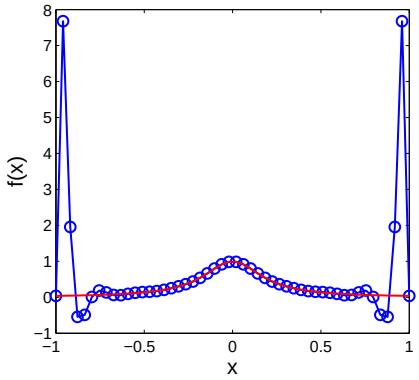

Figure 6: Polynomial interpolation of $f(x) = \frac{1}{1+25x^2}$.

From Figure 6 we see that polynomial interpolation does an acceptable job interpolating *f(x)* for $\frac{-1}{2} \leq x \leq \frac{1}{2}$. However, it can be seen that as the interpolant moves away from the center, problems occur. The oscillation of values toward the ends of the interval is known as the Runge Phenomenon [13, 26]. This oscillations of values is seen in Figure 5. There are several papers written about the Runge phenomenon in which they try to reduce the error in interpolation. In [26], the author suggests the use of Padé-Gegenbauer interpolants to resolve the oscillations.

There is also the Gibbs Phenomenon [12, 23] where the interpolant values oscillate when the underlying function is discontinuous. The Gibbs Phenomenon is seen in Figure 7. It has been found that there are many different ways to alleviate the Gibbs phenomenon [6, 17], such as Gegenbauer Reconstruction and filtering. One example of filtering is the Filtered Chebyshev Approximation.

Next we try the MQ Radial Basis Function for interpolating the function $f(x) = \frac{1}{1+25x^2}$ as seen in Figure 8. Again, the blue line represents the MQ interpolation of *f(x)* with the open blue circles representing N number of centers. The red line is the exact value of *f(x)*.

In Figure 8, we see that the Runge Phenomenon has disappeared. Below is Table 6 displaying the point-wise errors, the condition number of the system matrix, and different constant shape parameters. Table 6 is to further justify the MQ's ability to interpolate *f(x)*.

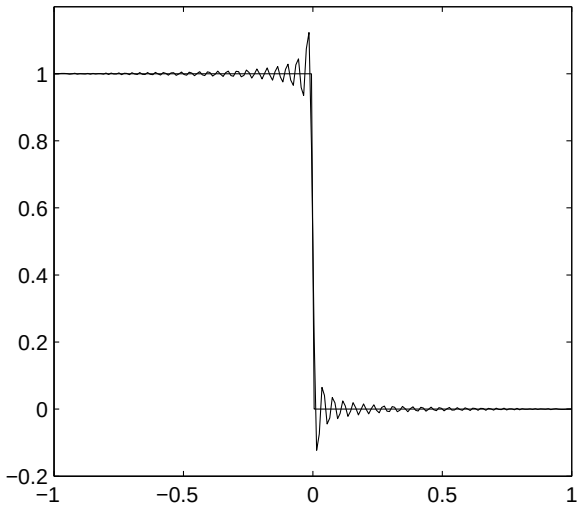

Figure 7: Interpolation of a step function, displaying the Gibbs Phenomenon.

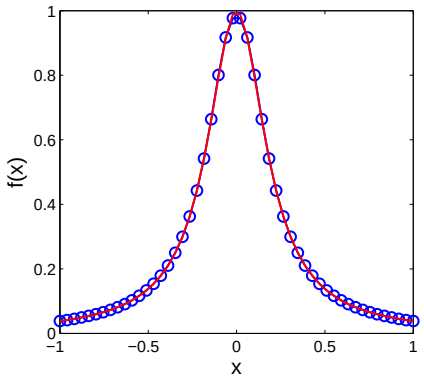

Figure 8: MQ RBF interpolation of $f(x) = \frac{1}{1+25x^2}$, $\varepsilon = 15$.

Definition 7 "Maximum Error"- Let $f(x)$ be the function that one is interpolating. Let $g(x)$ be the approximation of $f(x)$ produced by interpolation. Then the Maximum (Max) Error is defined as $\|g(x) - f(x)\|_\infty$ in g and f at any x.

Maximum Error	$\kappa(B)$	ε
0.0356	$7.3062e^{13}$	1
0.0124	$1.1991e^{3}$	15
0.0452	$3.1329e^{2}$	150
0.0535	$2.7893e^{2}$	1500

Table 6: Data of MQ interpolant of Figure 8.

Maximum Error	$\kappa(B)$
7.6368	$2.7224e^{8}$

Table 7: Data of Polynomial interpolant of Figure 6.

We conclude from Figure 8 and Table 6 that the MQ RBF interpolant produced better accuracy over that of a polynomial interpolation, in certain situations and is less susceptible to the Runge and Gibbs phenomenon.

4 Shape Parameter Strategies

In this chapter we apply numerical comparisons of the variable shape parameter to that of the constant shape parameter. The conclusions of the "best" shape parameter are drawn from numerical experiments. First we introduce the constant shape parameter, then follow it with the comparisons of the different variable shape parameters. The variable shape parameter strategy uses a different shape value at each of the centers. In Micchelli's Theorem 2, B being invertible applied only to the constant shape. It is conjectured that it is impossible to prove similar results in the more general setting of variable shape parameter. Despite firm theoretical underpinning, there are numerous results from a large collection of applications [4, 22, 24, 25] indicating advantages in using variable shape parameter strategies. For this reason, we preform numerical trials to justify our conclusions.

4.1 Constant Shape

Many scientists and mathematicians use the constant shape parameter for interpolation of data [14, 16, 18, 21]. There are many methods for choosing the "best" value of the shape parameter. The most obvious one is the brute force method. In the brute force method, one plots ε vs. the $|error|$, and then picks ε with least average $|error|$. Some other methods for finding the optimal constant shape parameter were produced by Hardy [18], Franke [14], and the leave-one-out cross-validation

(LOOCV) [16] algorithm referenced in [33]. The first two formulas for a good ε are defined below. Note, see the introduction to the thesis for refreshment of notation used.

Definition 8 "Hardy's c" -

$$c = .815d \text{ given } d = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{17}$$

where d_i is the distance from the i^{th} center to the nearest neighbor and N is the number of centers.

Definition 9 "Franke's c"-

$$c = \frac{1.25D}{\sqrt{N}} \tag{18}$$

where D is the diameter of the smallest circle encompassing all the center locations and N is the number of centers.

Many other people have tried to construct a satisfactory formula for the shape parameter in MQ interpolation as noted in [9, 38]. When creating an "optimal" shape parameter, one must combat the uncertainty principle. The goal of a shape parameter formula is for the interpolant to provide good accuracy with not too high of a condition number.

4.2 Why a Variable Shape Parameter

Theoretically speaking, the MQ RBF with a constant shape parameter is very hard to explain. When the theory is established for radial basis functions, a constant shape parameter was used. If one uses a variable shape parameter, the complexity of theory becomes extremely difficult to explain. In [4] there are somewhat restrictive sufficient conditions that show the system matrix B is non-singular with a variable shape parameter. One positive aspect of a variable shape parameter is that it creates distinct entries in the RBF matrices which lead to lower condition numbers [39]. The downside to a variable shape parameter is that it caused B to be nonsymmetric. Recall that if one uses a constant shape parameter, the system matrix B will be symmetric. There are additional papers that used the variable shape parameter if one would like to further explore this subject [4, 22, 24, 25].

4.3 Variable Linear Shape

The variable linear shape is a $1 \times N$ matrix that contains shape parameters generated by the formula,

$$\varepsilon_j = \varepsilon_{min} + (\frac{\varepsilon_{max} - \varepsilon_{min}}{N-1})j, \text{ for } j = 0,\ 1,\ \ldots, N-1 \qquad (19)$$

where ε_{min}, ε_{max}, and N are given in the script. The Matlab code used to generate the variable linear shape parameters is in Listing 2.

Listing 2: LinearVariableShapeFunction.m

```
%returns shape parameters in a 1xN matrix
function c = LinearVariableShapeFunction(eMax,eMin,N)

%creates a 1xN matrix of shape parameters
c = eMin + ((eMax-eMin)/(N-1)).*(0:N-1);
```

A graph of (19) with the following constraints: εMax=40, εMin=1, and $N = 20$ is shown in Figure 9.

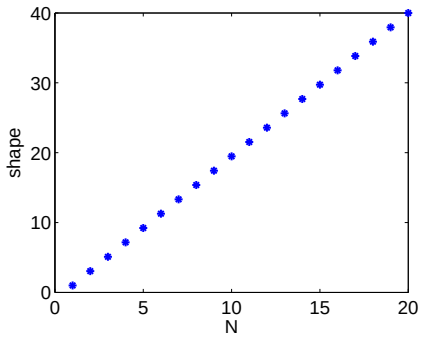

Figure 9: Plot of (19) with εMax=40, εMin=1, and $N = 20$.

Now we compare MQ radial basis interpolating with a constant and variable linear shape parameter. We use (19) to generate the shape parameters. The function for interpolation is $f(x) = e^{sin(\pi x)}$. We use a modified version of the Matlab code **stationaryInterpolationFP.m** in Listing 5, with εMax=7.6 and εMin=2.1. Figure 10 displays the absolute value of the point-wise error of interpolation for constant shape parameter of 3 and the variable linear shape parameter. Like in Figure 18, the blue line represents the variable shape parameter, and the red line represents the constant shape parameter. Additional informa-

tion from the example is in Table 8.

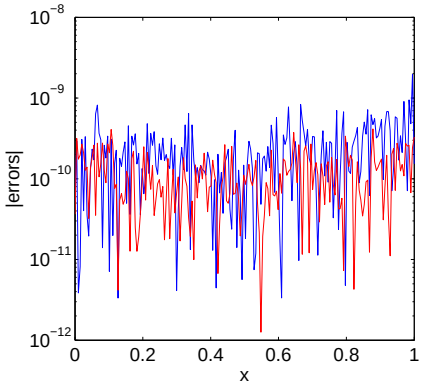

Figure 10: Plot of point-wise errors for a constant shape and variable linear shape parameter for the interpolation of $f(x) = e^{sin(\pi x)}$.

ε	$\kappa(B)$	Maximum Error	Avg. of $\|error\|$
Constant(3)	1.3856×10^{19}	4.1471×10^{-10}	1.185×10^{-10}
Variable Linear	1.5776×10^{19}	1.9557×10^{-9}	2.433×10^{-10}

Table 8: Data of constant vs. variable linear shape parameter.

Unlike the interpolation using the variable random shape parameter, the interpolation using the constant shape parameter achieved better results. In Table 10, we noted that interpolating using the constant shape parameter of 3 lead to a lower measure in error. The trend of increased error near the boundaries of the interval is more noticeable in Figure 10 than in Figure 18.

We now change to a different function and discuss the interpolation of it using a MQ RBF with a constant shape parameter and a variable linear shape parameter. The new function for interpolation is $g(x) =$

x^2+2x+1, a quadratic function. Again we use stationary interpolation from the modified Matlab script **stationaryInterpolationFP.m** in Listing 5. The values in the script have the same M and N like the previous interpolation of $f(x) = e^{sin(\pi x)}$. However, the values of εMin and εMax have changed. The values are εMin is 3.1 and εMax is 7.45. In Figure 11, the absolute values of the point-wise errors are shown. Table 9 displays data representing interpolation of $g(x) = x^2 + 2x + 1$. Again, the blue line represents the interpolation with a variable linear shape parameter and the red line represents the interpolation using a constant shape parameter.

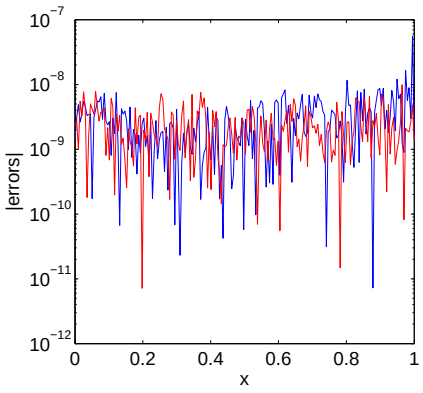

Figure 11: Pot of point-wise errors for a constant shape and variable linear shape parameter for the interpolation of $g(x) = x^2 + 2x + 1$.

ε	$\kappa(B)$	Maximum Error	Avg. of $\|error\|$
Constant(3)	$1.3857e^{19}$	$9.8922e^{-9}$	$2.501e^{-9}$
Variable Linear	$1.1497e^{19}$	$5.5578e^{-8}$	$3.3352e^{-9}$

Table 9: Data of constant vs. variable linear shape parameter 2.

In conclusion, we validated that interpolating with a constant shape parameter yielded better results than with a variable linear shape parameter.

So far, we have compared the variable linear shape parameter to a constant shape using stationary interpolation. We have looked at the graph of the absolute values of the point-wise errors and other data to conclude which shape parameter, constant or variable linear, is the best choice.

Now we need to determine if the variable linear shape parameter is the better choice in non-stationary interpolation. We use a modified version of the Matlab program **nonStationaryInterpolationFP.m** in Listing 6. Recall that in non-stationary interpolation N is allowed to vary while ε remains constant. For this example, N took on the values of 10 to 250 in intervals of 10. The values of εMin and εMax for variable linear shape generator (19) are 1.9909 and 9, respectivley. We interpolate $f(x) = e^{sin(\pi x)}$ on the interval [-1,1]. Figure 12 displays the absolute values of the point-wise errors. The blue line refers to $|error|$ in the interpolation using the variable linear shape parameter. The red line refers to $|error|$ in the interpolation using a constant shape parameter. Table 12 contains more information about this interpolation using a constant and variable linear shape parameter.

Using a variable linear shape parameter in the non-stationary interpolation of $f(x) = e^{sin(\pi x)}$ did not reduce the errors when compared to a constant shape parameter. Using a constant shape parameter did

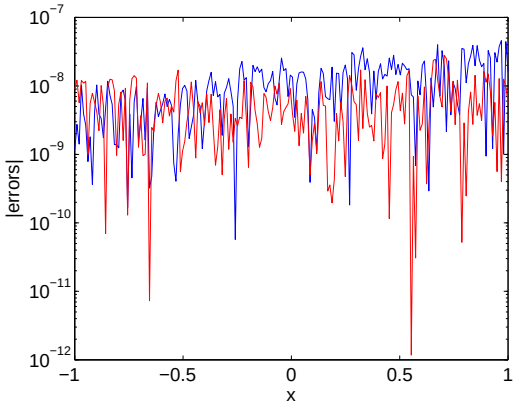

Figure 12: Plot of point-wise errors for a constant shape and variable linear shape parameter for the non-stationary interpolation of $f(x) = e^{sin(\pi x)}$.

ε	$\kappa(B)$	Maximum Error	Avg. of $\|error\|$
Constant(5)	$1.8771e^{19}$	$2.8068e^{-8}$	$5.597e^{-9}$
Variable Linear	$1.6078e^{19}$	$4.5933e^{-8}$	$1.168e^{-8}$

Table 10: Data of constant vs. variable linear shape parameter in non-stationary interpolation.

reduce the maximum error approximately one half the value of the variable linear shape parameter. The constant shape parameter also had the least absolute average of error.

4.4 Exponentially Varying Shape

The exponentially varying shape is a $1 \times N$ matrix that contains shape parameters generated by the formula,

$$\varepsilon_j = \left[\varepsilon_{min}^2 \left(\frac{\varepsilon_{max}^2}{\varepsilon_{min}^2} \right)^{\frac{j-1}{N-1}} \right]^{1/2}, \text{ for } j = 1,\ 2,\ \ldots,\ N \tag{20}$$

where ε_{min}, ε_{max}, and N are given in the script. The Matlab code used to generate the exponentially varying shape parameters is in Listing 3.

Listing 3: expShapeFunction.m

```
%returns shape parameters in a 1xN matrix
function c = expShapeFunction(eMax,eMin,N)

%creates a 1xN matrix of shape paramters
c = eMin^2.*(eMax^2 eMin^2).^(((1:N)-1)/(N-1)).^(1/2);
```

Next we show a plot of (20) with the following constraints: εMax=40, εMin=1, and $N = 20$ in Figure 13.

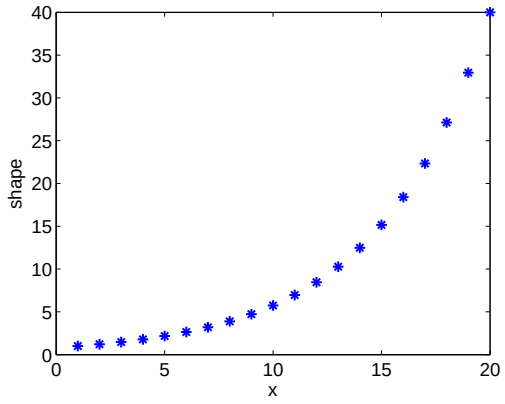

Figure 13: Plot of (20) with εMax=40, εMin=1, and $N = 20$.

Now we compare, as we have in previous sections, interpolating $f(x) = e^{sin(\pi x)}$ using a MQ radial basis function with constant shape parameter and a variable shape parameter. This variable shape parameter is exponential and produced by equation (20). The values for N, M, εMax, and εMin are 100, 198, 7.6, and 2.1, respectively. The program we use to produce Figure 14 and Table 11 is again a modi-

fied version of **stationaryInterpolationFP.m**. The constant shape parameter is set to 3.

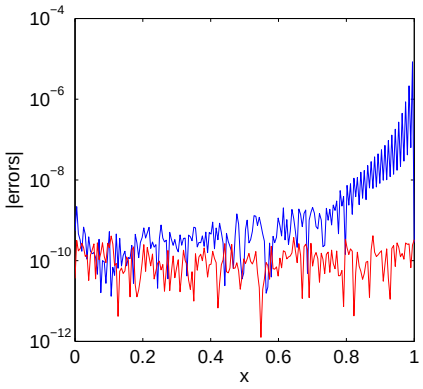

Figure 14: Plot of point-wise errors for a constant shape and exponentially varying shape parameter for the interpolation of $f(x) = e^{sin(\pi x)}$.

ε	$\kappa(B)$	Maximum Error	Avg. of $\|error\|$
Constant(3)	$1.3856e^{19}$	$4.1471e^{-10}$	$1.185e^{-10}$
Exponentially Varying	$1.2481e^{19}$	$8.4253e^{-6}$	$6.685e^{-8}$

Table 11: Information of constant vs. exponentially varying shape parameter.

Changing the constant shape parameter to an exponentially varying shape parameter did not reduce the interpolant error. The constant shape parameter of 3 is clearly the best. Also, note how the absolute error increases as one moves away from the center of the interval. This problem is more severe with the exponentially varying shape parameter than the constant shape parameter.

Now we change the function that we are interpolating. The function is $g(x) = x^2 + 2x + 1$, where $x \in [0, 1]$. For this interpolation, we use a

modified version of the Matlab script **stationaryInterpolationFP.m**. The constant shape parameter is 3. We use the exponentially varying shape parameters produced by equation (20) with εMin equal to 2.7 and εMax equal to 7.999. Figure 15 shows the absolute point-wise errors in interpolation using a constant shape parameter, the red line, and the exponentially varying shape parameter, the blue line. Table 12 shows further information from this example.

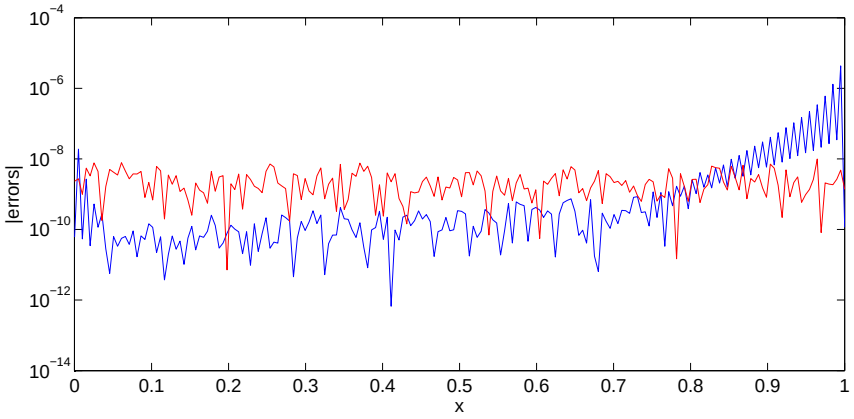

Figure 15: Plot of point-wise errors for a constant shape and exponentially varying shape parameter for the interpolation of $g(x) = x^2 + 2x + 1$.

ε	$\kappa(B)$	Maximum Error	Avg. of $\|error\|$
Constant(3)	$1.3857e^{19}$	$9.8922e^{-9}$	$2.501e^{-9}$
Exponentially Varying	$1.8413e^{19}$	$4.3185e^{-6}$	$3.818e^{-8}$

Table 12: Information of constant versus exponentially varying shape parameter 2.

As we have previously seen in the interpolation of $f(x) = e^{sin(\pi x)}$, using a constant shape parameter in the MQ RBF interpolation yielded

the best results.

We will compare a constant shape parameter to that of an exponentially varying shape parameter in non-stationary interpolation. Non-stationary interpolation was explained in Section 4.2. For this interpolation we use $M = 198$, a constant shape parameter of 5, and N varying from 10 to 250 in intervals of 10. We use non-stationary interpolation on function $f(x) = e^{sin(\pi x)}$ where $x \in [-1, 1]$. The script we use is a modified version of **nonStationaryInterpolationFP.m** Matlab file.

Graph 16 depicts the absolute values of the point-wise error for the different non-stationary interpolation. The blue line corresponds to the exponentially varying shape parameter. The red line corresponds to the constant shape parameter. Table 13 lists further details from the example. We used the values of 1.9595 and 9 for εMin and εMax, respectively in (20).

ε	$\kappa(B)$	Maximum Error	Avg. of $\lvert error \rvert$
Constant(5)	$1.8771e^{19}$	$2.8068e^{-8}$	$5.957e^{-9}$
Exponentially Varying	$1.4885e^{19}$	$9.6342e^{-7}$	$2.237e^{-8}$

Table 13: Information of constant vs. exponentially non-stationary.

In conclusion of the non-stationary interpolation of $f(x) = e^{sin(\pi x)}$, we documented that the constant shape parameter has once again yielded better results than an exponentially varying shape parameter. Using the constant shape parameter produced less maximum error and had a lower absolute mean error. Even if we reduced the error in inter-

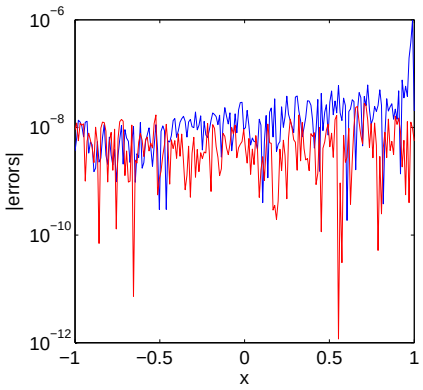

Figure 16: Plot of point-wise errors for a constant shape and exponentially varying shape parameter for the non-stationary interpolation of $f(x) = e^{sin(\pi x)}$.

polation near the boundaries, the constant shape parameter would still interpolate with fewer errors. This is illustrated in Figure 16, where the absolute values of the point-wise errors using a exponentially varying shape (blue line) lies in general above point-wise errors for a constant shape non-stationary interpolation (red line).

4.5 Variable Random Shape

The variable random shape is a $1 \times N$ matrix that contains shape parameters generated by the formula,

$$\varepsilon_j = \epsilon_{min} + (\varepsilon_{max} - \varepsilon_{min}) rand(1, N) \tag{21}$$

where ε_{min}, ε_{max}, and N are given in the script in Listing 4. The rand(1,N) command is a Matlab function that creates a $1 \times N$ matrix of randomly generated numbers from a unit interval.

Listing 4: randomVariableShapeFunction.m

```
%returns shape parameters in a 1xN matrix
function c = RandomVariableShapeFunction(eMax,eMin,N)

%creates a 1xN matrix of shape parameters
%rand(x,y) is a random number generator created by Matlab
c = eMin + (eMax-eMin).*rand(1,N);
```

In Figure 17 the graph of (21) with the following constraints: εMax=40, εMin=1, and $N = 20$.

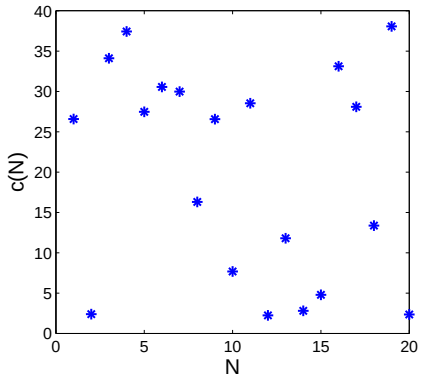

Figure 17: Graph of (21) with εMax=40, εMin=1, and $N = 20$.

Now we compare the random variable shape parameters in a MQ radial basis function interpolant to that of a constant shape parameter. In order to compare the two interpolants, we need to make sure the condition number of the system matrix is relatively close. The condition number of the system matrix is noted as *kappa* or $\kappa(B)$. We use stationary interpolation where N, the number of centers, and M, the number of evaluation points are fixed. For this comparison we use the Matlab program **stationaryInterpolationFP.m**, found in Listing 5.

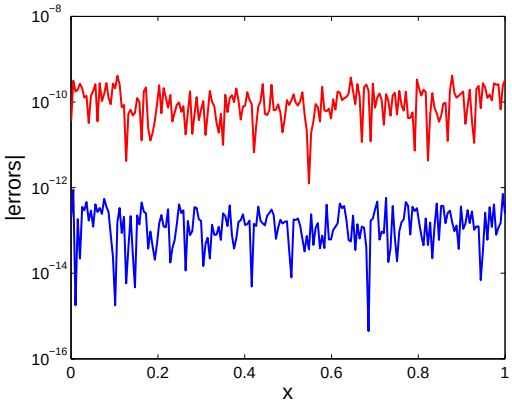

Figure 18: Plot of point-wise errors for a constant shape and variable random shape parameter for the interpolation of $f(x) = e^{sin(\pi x)}$.

The function being interpolated is $f(x) = e^{sin(\pi x)}$, where $x \in [0, 1]$. For this comparison we have chosen a constant shape parameter of 3, and variable random shape parameter generated by (21) with εMax=7.6 and εMin=2.1. **Note:** When we use stationary interpolation the values for M and N will be 198 and 100, unless otherwise noted. Figure 18 shows the point-wise errors in the approximation of $f(x) = e^{sin(\pi x)}$ using both a constant shape parameter and a variable random shape parameter. The blue line in Figure 18 represents the point-wise errors for the variable random shape parameter. The red line represents the point-wise errors for the constant shape parameter. Data from the example is shown in Table 14.

We see from the data that interpolating with a variable random shape parameter does a far better job, than using a constant shape parameter. In Table 14, the variable random shape parameter had a

ε	$\kappa(B)$	Maximum Error	Average of $\lvert error \rvert$
Constant(3)	1.3857×10^{19}	4.1471×10^{-10}	1.185×10^{-10}
Variable Random	1.6287×10^{19}	8.8574×10^{-13}	1.731×10^{-13}

Table 14: Data of constant vs. variable random shape parameter.

lower value in both the maximum error and the mean of the absolute error. Also, notice as one moves away from the center of the interval that the error increases. This is a common problem in interpolating with a radial basis function, because errors occurr more frequently near the bounds [3, 10, 11]. This problem is more evident in further interpolations that we explore in more detail in Chapter 6.

As we have seen previously, using a variable random shape parameter for the MQ radial basis function delivered better accuracy in interpolating $f(x) = e^{sin(\pi x)}$ than using a constant shape parameter. Now we experiment with a different function. The function we interpolate is a quadratic and given by $g(x) = x^2+2x+1$. We use stationary interpolation with εMax=7.6 and εMin=2.7 . The Matlab program we use is a modified version of **stationaryInterpolationFP.m**. The constant shape parameter is 3. Note: It was difficult to obtain relatively close condition numbers for the two system matrices since the variable random shape parameters is generated by equation (21), which contains a random number generator. Therefore, we had to run the program quite a few times to obtain a $\kappa(B)$ that was relatively close to the condition number of the system matrix with a constant shape parameter. If the $\kappa(B)$'s of the two different system matrix were not the same, our

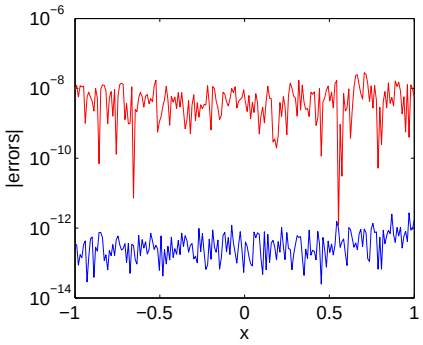

Figure 19: Plot of point-wise errors for a constant shape and variable random shape parameter for the interpolation of $g(x) = x^2 + 2x + 1$.

comparison of shape parameter strategies would be flawed. It would be "numerically insignificant" to compare the shape parameter strategies if their $\kappa(B)$'s were different. Figure 19 shows the absolute point-wise errors of the interpolation. The red line represents the MQ RBF interpolant with a constant shape parameter. The blue line represents the MQ RBF interpolant with the variable random shape parameter. Data of Figure 19 is shown in Table 15.

ε	Kappa	Maximum Error	Average of $\|error\|$
Constant(3)	$1.3856e^{19}$	$9.8921e^{-9}$	$2.501e^{-9}$
Variable Random	$1.6958e^{19}$	$3.2751e^{-12}$	$3.574e^{-13}$

Table 15: Data of figure 19.

In conclusion, interpolating $g(x) = x^2 + 2x + 1$ using a MQ RBF with a variable random shape parameter produced less error than using a constant shape parameter.

So far we have explored stationary interpolation with the MQ RBF

with a constant and variable shape parameter. Next we explore non-stationary interpolation with a constant and variable shape parameters, and discuss the results. The Matlab code we use is **nonStationary-InterpolationFP.m** in Listing 6. The first function for interpolation is $f(x) = e^{sin(\pi x)}$ where $x \in [-1, 1]$. In non-stationary interpolation N, the number of centers, does not stay constant. In **nonStationaryInterpolationFP.m**, N takes on the value of 10 to 250 in intervals of 10. The value of M stayed the same, and the constant shape parameter is set at 3. The algorithm, **nonStationaryInterpolationFP.m**, can be found in Listing 6.

Figure 20 represents the absolute values of the point-wise errors for the non-stationary interpolation of the function, $f(x) = e^{sin(\pi x)}$ where $x \in [-1, 1]$. The blue line represents the random variable shape parameter and the red line represents the constant shape parameter. Here the constant shape parameter is 5. Note: We have changed the shape parameter from 3 to 5 as in the previous interpolations. The reason for this was to reduce the condition number of the system matrix. Table 16 displays more information about the example.

ε	Kappa	Maximum Error	Average of $\|error\|$
Constant(5)	$1.8771e^{19}$	$2.8068e^{-8}$	$5.9570e^{-9}$
Variable Random	$1.9105e^{19}$	$2.7276e^{-12}$	$4.2190e^{-13}$

Table 16: Data of constant versus variable random shape parameter using non-stationary interpolation.

Just as before in stationary interpolation of $f(x) = e^{sin(\pi x)}$, we had

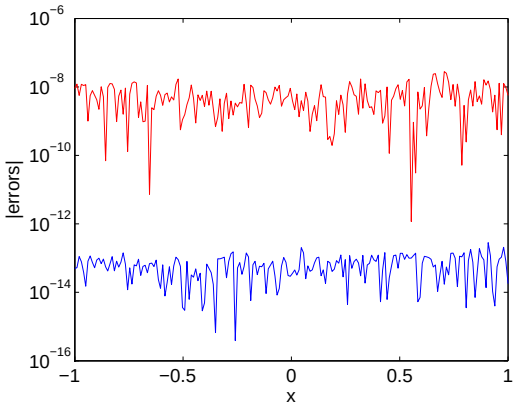

Figure 20: Plot of point-wise errors for a constant shape and variable random shape parameter for the non-stationary interpolation of $f(x) = e^{sin(\pi x)}$.

to run the program several times to compare interpolations. The values we use for the variable random shape generator (21) were εMax=9 and εMin=2. As before for stationary interpolation, using a variable random shape parameter yielded far better accuracy in the non-stationary interpolation of $f(x) = e^{sin(\pi x)}$. We see in Table 16 that using a variable random shape parameter decreased both the maximum $|error|$ and the average of the absolute point-wise errors.

4.6 Best Shape Parameters on Different Functions in 1D?

In this section, we experiment numerically to find out what type of shape parameters works best with different functions. As stated before, given the complexity of variable shape parameter these results are based on numerical data and not proofs. This subsection is broken down further for each function as we compare shape parameter

strategies. While working with these functions, we use different values of (M, N). We produced graphs that plot the max $|error|$ against the average values of εMin and εMax denoted CAverage. We provided some numerical results to further justify our conclusions. The numerical results include the following: $\kappa(B)$, the average of the max $|error|$ over the range of CAverage, M, and N. For the comparison in one dimension, we use the Matlab code **errorLoop.m**. This code is found in Listing 7.

4.6.1 Sinusoid function

For the first comparison we use a smooth geometric function. The function is $f(x) = \sin(\pi x)$ where $x \in [-1, 1]$. We collect the data values for the tables at z=50 points. The values of εMin and εMax are 2 and 4. We increment εMin and εMax by .05. The comparison is run for different values of M and N. Recall that, M is the number of evaluation points and N is the number of centers. The values of (M, N) are (15,9), (33,20), (198,100), and (300,250). One might ask why there exist such an "odd" pairing of M and N. The reason is that we do not want the evaluation points and center locations to coincide with each other. If the evaluation points and center locations do coincide, the error in interpolation would be zero. Figure 21 displays the $|error|$ versus the average shape parameter.

It is clearly documented in the bottom two images of Figure 21, that interpolating a sinusoid function using a variable shape parameter

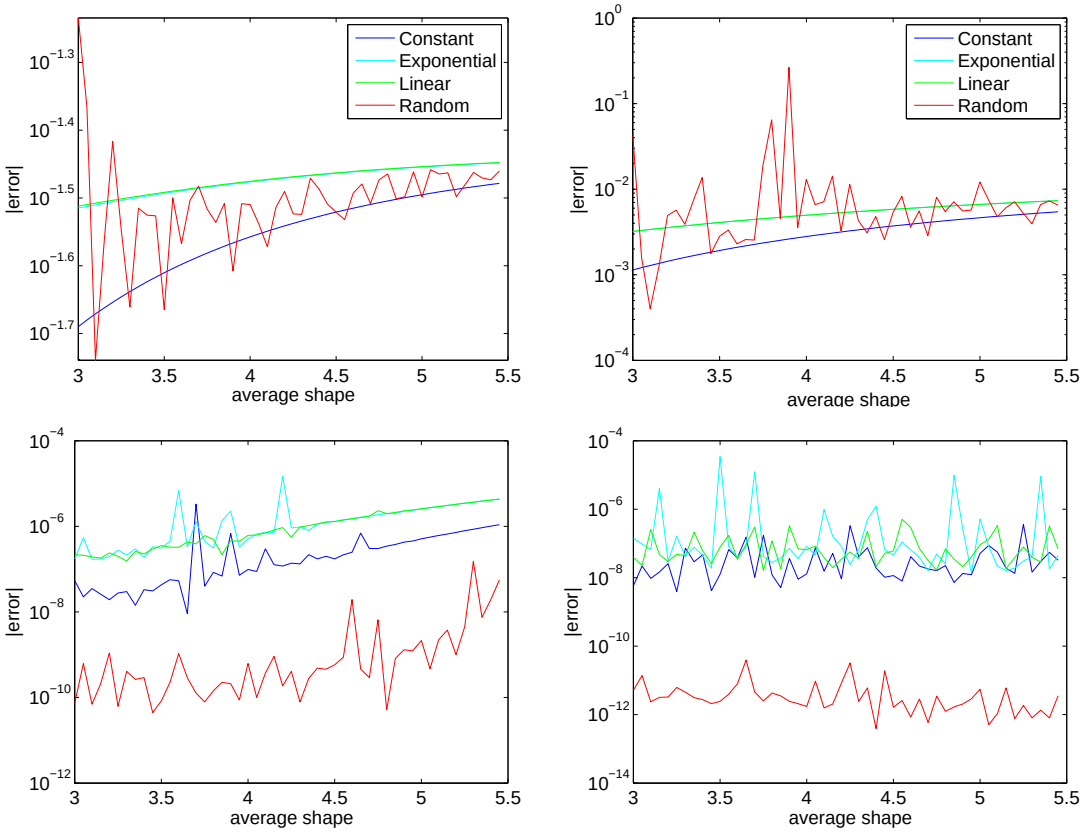

Figure 21: A comparison of shape parameters involving interpolating the function,$f(x) = \sin(\pi x)$ with different values of (M,N). Top left: (15,9), Top Right: (33,20), Bottom Left: (198,100), Bottom Right: (300,250).

with large (M, N) ($M \geq 198$ and $N \geq 100$) provides the best accuracy. Listed in Table 17 we see that using $M = 300$ and $N = 250$, the MQ RBF had $\kappa(B) = 1.7475e^{19}$ and an average $|error|$ of $4.9043e^{-12}$. In conclusion, using a variable random shape parameter with a MQ RBF to interpolate $sin(\pi x)$ produced better results than the other shape parameter strategies with M, N large. However, if one uses low values

Name of Data	$(M,N) = (15,9)$	$(M,N) = (33,20)$	$(M,N) = (198,100)$	$(M,N) = (300,250)$
κ_C	$2.9789e^{2}$	$2.5829e^{4}$	$3.1282e^{15}$	$1.5259e^{19}$
avg. $\lvert error \rvert$ for κ_C	0.0282	0.0032	$3.5135e^{-7}$	$4.5595e^{-8}$
κ_E	$3.5613e^{2}$	$4.4916e^{4}$	$1.4584e^{17}$	$3.6191e^{19}$
avg. $\lvert error \rvert$ for κ_E	0.0336	0.0053	$1.8222e^{-6}$	$1.5149e^{-6}$
κ_L	$3.4479e^{2}$	$4.2059e^{4}$	$1.1121e^{17}$	$3.2177e^{19}$
avg. $\lvert error \rvert$ for κ_L	0.0337	0.0053	$1.3515e^{-6}$	$9.901e^{-8}$
κ_R	$2.8447e^{2}$	$2.2516e^{4}$	$5.9495e^{15}$	$1.7475e^{19}$
avg. $\lvert error \rvert$ for κ_R	0.0316	0.013	$5.7665e^{-9}$	$4.9043e^{-12}$

Table 17: Numerical results from $\sin(\pi x)$.

of M and N, the constant shape parameter is the best choice. This choice of using low values of M and N is ill-advised.

Next, we increase the peaks and valleys of $sin(\pi x)$, to see if our conclusion held above. Let $f(x) = sin(4\pi x)$, $f(x)$ is shown in Figure 22, where $x \in [-1, 1]$. Note: The only thing that was different from the previous comparison of $f(x) = sin(\pi x)$ is that the function has now changed to $f(x) = sin(4\pi x)$. Everything else stays the same. For a more concise comparison of the shape parameter strategies, we only provide their plots of the $|error|$ vs. average shape in Figure 23 for different values of (M, N) and the data involving $(M, N) = (300, 250)$.

Name of Data	Trial $(M,N) = (300,250)$
κ_R	$2.2119e^{19}$
avg. $\lvert error \rvert$ for κ_R	$1.59e^{-7}$

Table 18: Numerical information of Figure 23, case $(M, N) = (300, 250)$.

For the 300 evaluation points and 250 centers, the best accuracy overall was achieved using a variable random shape parameter. With the other three cases of (M, N) the constant shape in Listing 7 worked

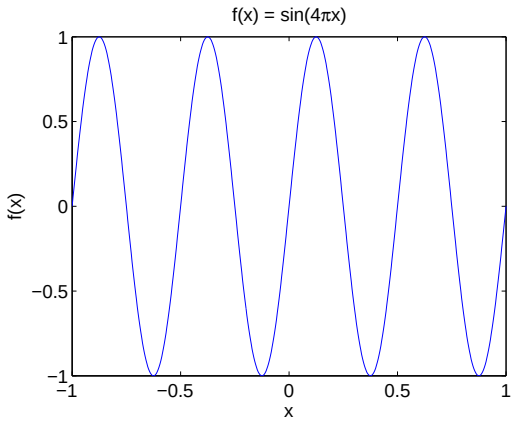

Figure 22: The function, $f(x) = sin(4\pi x)$ where $x \in [-1, 1]$.

the best. Data of the interpolation for $(M, N) = (300, 250)$ is shown in Table 18. Thus we can conclude from Figure 23 and Table 23, that for (M, N) large the random shape parameter yielded the best results.

For the last comparison involving a sinusoid function we multiply π by $\frac{1}{4}$, and then compare the different shape strategies with (M, N) varying. Let $f(x) = sin(\frac{1}{4}\pi)$ where $x \in [-1, 1]$. We mainly use graphs of $|error|$ vs. εAvgerage to reinforce our conclusions of the most accurate shape strategy to use. The graphs of the varying (M, N) values are shown in Figure 24. Also, data from the largest (M, N) graph is in Table 19 for reference.

Name of Data	Trial $(M, N) = (300, 250)$
κ_R	$1.5137e^{19}$
avg. $\|error\|$ for κ_R	$6.3397e^{-13}$

Table 19: Numerical information of Figure 24, case (M,N)=(300,250).

The best accuracy for the interpolation of $f(x) = sin(\frac{1}{4}\pi)$ where

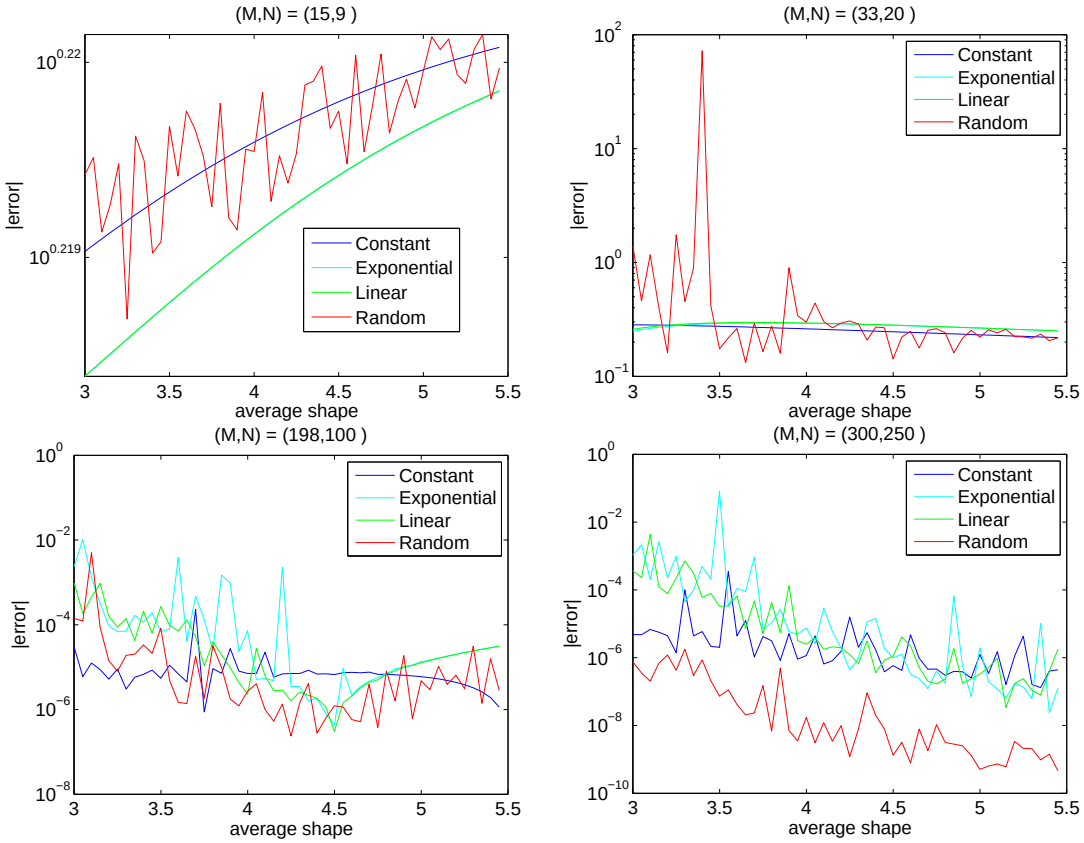

Figure 23: A comparison of shape parameters interpolating $f(x) = sin(4\pi x)$ for different values of M and N.

$x \in [-1, 1]$ is provided by the variable random shape with the $M = 300$ and $N = 200$. The approximation of $f(x)$ has a mean absolute error of $6.3397e^{-13}$ and $\kappa_R = 1.5137e^{19}$. The constant shape parameter did produce better accuracy than the exponential, linear, or random shape parameter when (M, N) was small (i.e.(15,9) and (33,20)). The variable exponential and linear shape parameter did not provide the lowest absolute error in any of the cases (M, N).

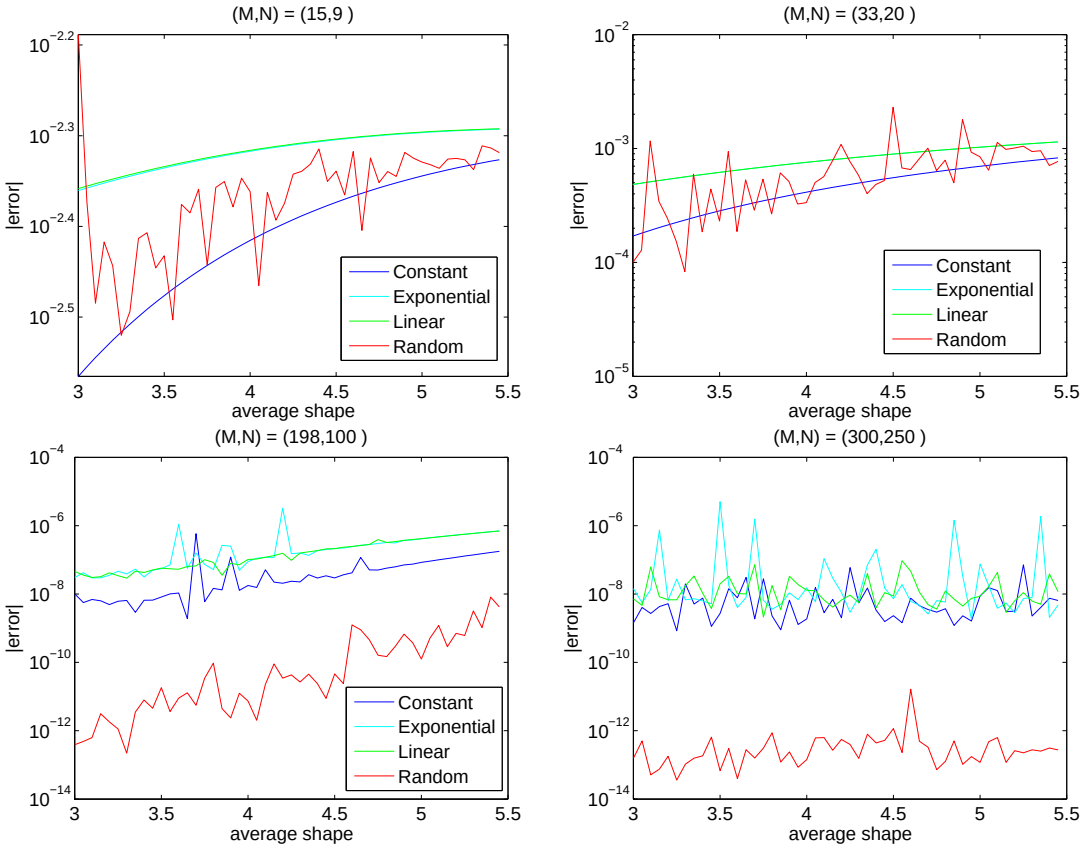

Figure 24: A comparison of shape parameters interpolating $f(x) = sin(\frac{1}{4}\pi x)$ for different values of M and N.

In conclusion of the subsection, we have found that a variable random parameter provided the least absolute error when interpolating a sinusoid graph.

4.6.2 Constant function

Several researchers have reported that the RBF method has difficulty accurately approximating flat functions [24]. For the second test, we

use a constant function to see which shape parameter and values of (M, N) produce the least absolute error. We iterate (M, N) as in the previous subsubsection. The constant function for interpolation is $f(x) = 1$. The interpolation is on the interval [-1,1]. Figure 25 displays the graphs of the $|error|$ vs. the average shape for various values of (M, N). Table 20 contains the data of each of the graphs listed in Figure 25.

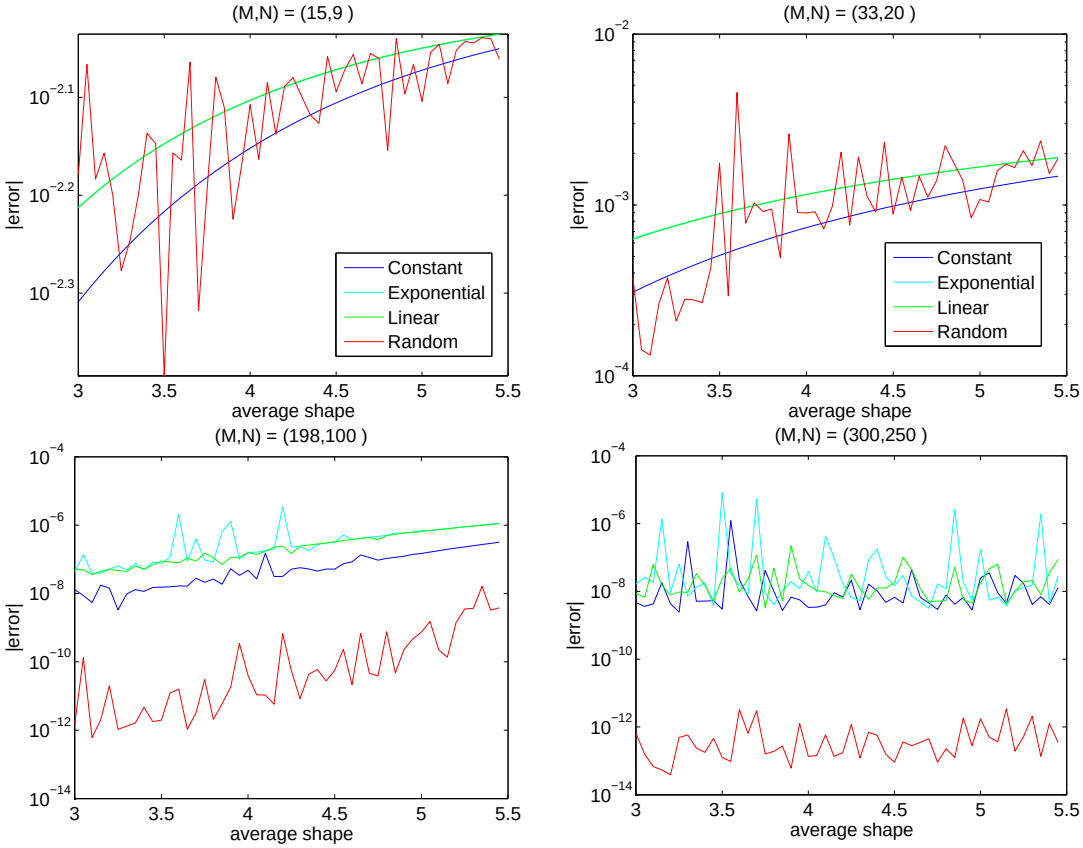

Figure 25: A comparison of shape parameters interpolating $f(x) = 1$ for different values of M and N.

Name of Data	$(M,N)=(15,9)$	$(M,N)=(33,20)$	$(M,N)=(198,100)$	$(M,N)=(300,250)$
κ_C	$2.9789e^{2}$	$2.5829e^{4}$	$3.1282e^{15}$	$1.5259e^{19}$
avg. $\lvert error\rvert$ for κ_C	0.0072	$8.6203e^{-4}$	$8.394e^{-8}$	$3.9805e^{-8}$
κ_E	$3.5613e^{2}$	$4.4916e^{4}$	$1.4584e^{17}$	$3.6191e^{19}$
avg. $\lvert error\rvert$ for κ_E	0.008	0.0013	$4.903e^{-7}$	$4.1709e^{-7}$
κ_L	$3.448e^{2}$	$4.2059e^{4}$	$1.1121e^{17}$	$3.2177e^{19}$
avg. $\lvert error\rvert$ for κ_L	0.008	0.0013	$3.3835e^{-7}$	$2.7828e^{-8}$
κ_R	$3.0565e^{2}$	$2.7032e^{4}$	$9.3258e^{15}$	$1.1787e^{19}$
avg. $\lvert error\rvert$ for κ_R	0.0077	0.0012	$7.7322e^{-10}$	$6.1279e^{-13}$

Table 20: Numerical information from Figure 25.

From Table 20 and Figure 25, we conclude that using a constant shape parameter for (M,N) equalling (15,9) and (33,20) yielded the least average $|error|$ for interpolation of a constant function. However, interpolating a function on a given set of data requires a more stringent bound on error than 0.0072 or $8.6203e^{-4}$. To achieve better accuracy, we need to increase the number of evaluation points and center locations in our RBF interpolation. One sees in Table 20 that once (M,N) is increased to (198,100) and (300,250) interpolating the constant function with a variable random shape parameter produced the best results. The best reduced average $|error|$ is achieved by setting (M,N) to (300,250) producing an average $|error|$ of $6.1279e^{-13}$ with a condition number of $1.1787e^{19}$.

4.6.3 Polynomial

We have already documented in the two previous subsubsections that using a variable random shape parameter combined with different number of evaluation points and center locations that we can achieve a

relatively close approximation to the function. Now we look further into this concept and experiment with a polynomial. The polynomial for interpolation is $f(x) = x^2 + 2x + 1$ where $x \in [-1, 1]$. Again like the previous comparisons, we use different shape parameter strategies combined with different pairs of (M, N). The error analysis of the different cases of (M, N) are in Figure 26, while the information of this figure is in Table 21.

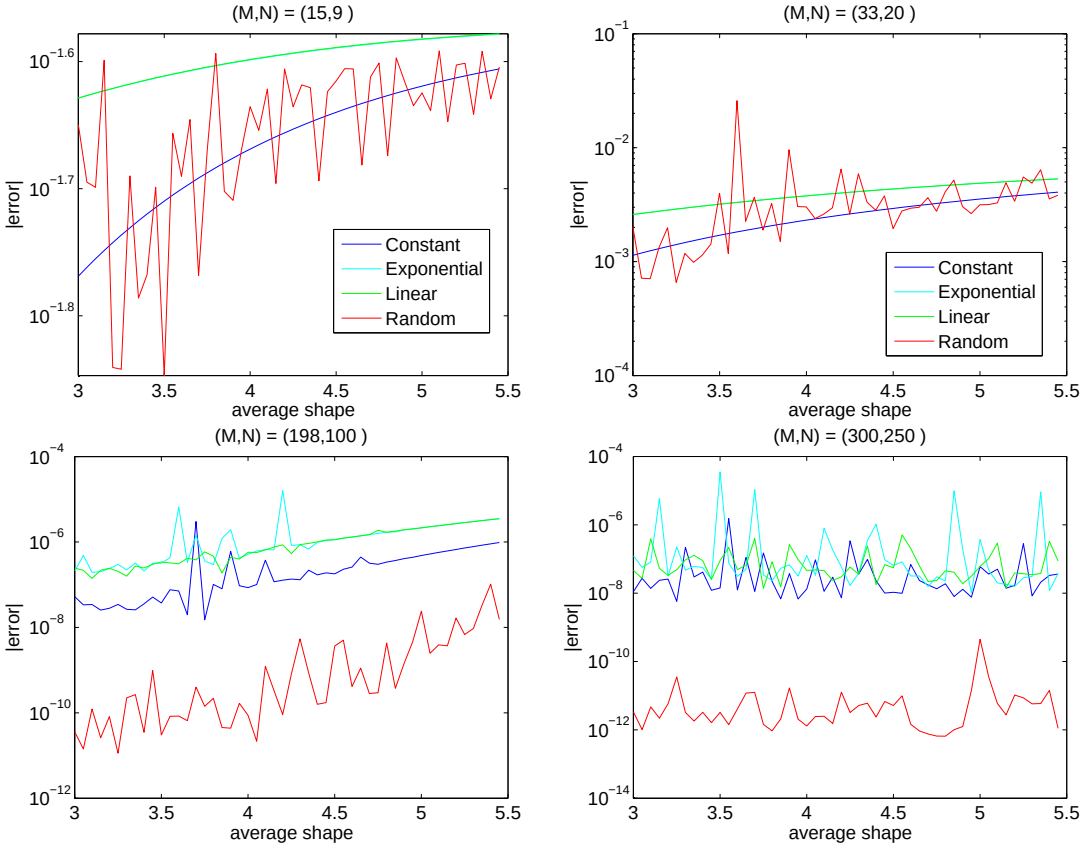

Figure 26: A comparison of shape parameters interpolating $f(x) = x^2 + 2x + 1$ for different values of M and N.

Name of Data	$(M,N)=(15,9)$	$(M,N)=(33,20)$	$(M,N)=(198,100)$	$(M,N)=(300,250)$
κ_C	$2.9789e^{2}$	$2.5829e^{4}$	$3.1282e^{15}$	$1.5259e^{19}$
avg. $\lvert error\rvert$ for κ_C	0.0217	0.0026	$3.2978e^{-7}$	$7.4456e^{-8}$
κ_E	$3.5613e^{2}$	$4.4916e^{4}$	$1.4584e^{17}$	$3.6191e^{19}$
avg. $\lvert error\rvert$ for κ_E	0.0253	0.004	$1.6498e^{-6}$	$1.5149e^{-6}$
κ_L	$3.4479e^{2}$	$4.2059e^{4}$	$1.1121e^{17}$	$3.2177e^{19}$
avg. $\lvert error\rvert$ for κ_L	0.0253	0.004	$1.1542e^{-6}$	$9.7714e^{-8}$
κ_R	$3.0564e^{2}$	$2.7032e^{4}$	$1.7331e^{16}$	$1.7288e^{19}$
avg. $\lvert error\rvert$ for κ_R	0.022	0.0036	$5.03978e^{-9}$	$1.4695e^{-11}$

Table 21: Numerical information from Figure 26.

We conclude from Figure 26 and Table 21, involving a large number of centers (here $N \geq 100$) that interpolating a polynomial with a random variable shape parameter produces the least error. When a system matrix becomes ill-conditioned, we also documented that using a variable random shape parameter produces the highest accuracy. Note: When (M,N) were low the constant shape parameter produced the best accuracy, but when we increased (M,N) large the random variable shape parameter produced the best accuracy. Using RBF interpolation with a variable random shape parameter to interpolate $f(x) = x^2 + 2x + 1$ where $x \in [-1,1]$ produced an average absolute error of $1.4695e^{-11}$ with $\kappa_R = 1.7288e^{19}$. This high of κ_R, signals that B is ill-conditioned.

4.6.4 -Arctan function

In this test, we use the function $f(x) = -\arctan(5(x+\frac{1}{2}))$ where $x \in [-1,1]$ to determine which is the best shape parameter to use with different values of (M,N). The graphs of the $|error|$ vs. the average

shape are shown in Figure 27, while the data for the different case of (M,N) is in Table 22.

Note: The data in Table 22, and hereafter in this section is shown approximated for space issues for the thesis.

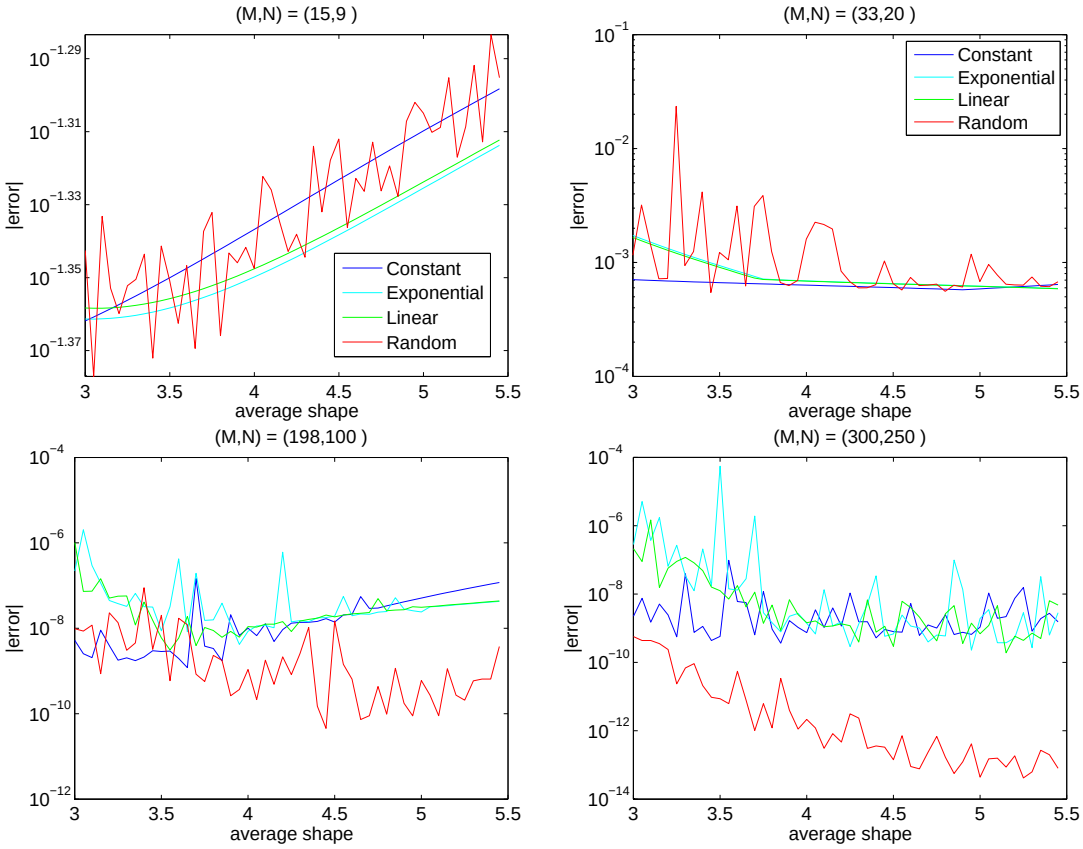

Figure 27: A comparison of shape parameters interpolating $f(x) = -\arctan(5(x+\frac{1}{2}))$ for different values of M and N.

Again, we see the same trend as the previous comparisons of shape parameter strategies. For large (M, N) and an ill-conditioned system matrix, the variable random shape parameter yields the best interpo-

Name of Data	$(M,N)=(15,9)$	$(M,N)=(33,20)$	$(M,N)=(198,100)$	$(M,N)=(300,250)$
κ_C	$2.9788e^{2}$	$2.5828e^{4}$	$3.1282e^{15}$	$1.5259e^{19}$
avg. $\lvert error\rvert$ for κ_C	0.0467	$6.3121e^{-4}$	$2.9658e^{-8}$	$5.4636e^{-9}$
κ_E	$3.5613e^{2}$	$4.4916e^{4}$	$1.4584e^{17}$	$3.6191e^{19}$
avg. $\lvert error\rvert$ for κ_E	0.0454	$8.0134e^{-4}$	$1.0065e^{-7}$	$1.3112e^{-6}$
κ_L	$3.4479e^{2}$	$4.2058e^{4}$	$1.1121e^{17}$	$3.2177e^{19}$
avg. $\lvert error\rvert$ for κ_L	0.04578	$7.8647e^{-4}$	$4.7891e^{-8}$	$4.669e^{-8}$
κ_R	$3.0564e^{2}$	$2.9473e^{4}$	$1.5986e^{16}$	$1.8768e^{19}$
avg. $\lvert error\rvert$ for κ_R	0.0467007	0.0016	$5.4167e^{-9}$	$4.7801e^{-11}$

Table 22: Numerical information from Figure 27.

lation of the inverse trigonometric function $f(x) = -\arctan(5(x+\frac{1}{2}))$. For 300 evaluation points and 250 center locations, the variable random shape parameter produce an average $|error|$ of $4.7801e^{-11}$.

4.6.5 Exponential function

Next, we experiment with an exponential function to see which shape parameter combined with different number of evaluation points and center locations produces the most accurate interpolation. The exponential function for interpolation is $f(x) = e^{\sin(\pi x)}$ where $x \in [-1,1]$. Figure 28 displays the $|error|$ vs. the average shape for the four different cases of (M,N) and each shape parameter strategy. The condition number and the average $|error|$ for the cases is in Table 23.

We see something very different in this comparison than the previous comparisons. Note: The previous comparisons for low values of (M,N)$\leq$(33,20), the constant shape parameter produce the best interpolation. However, in this test this is not true. For low values of (M,N), the linear shape parameter is the best choice for interpolating

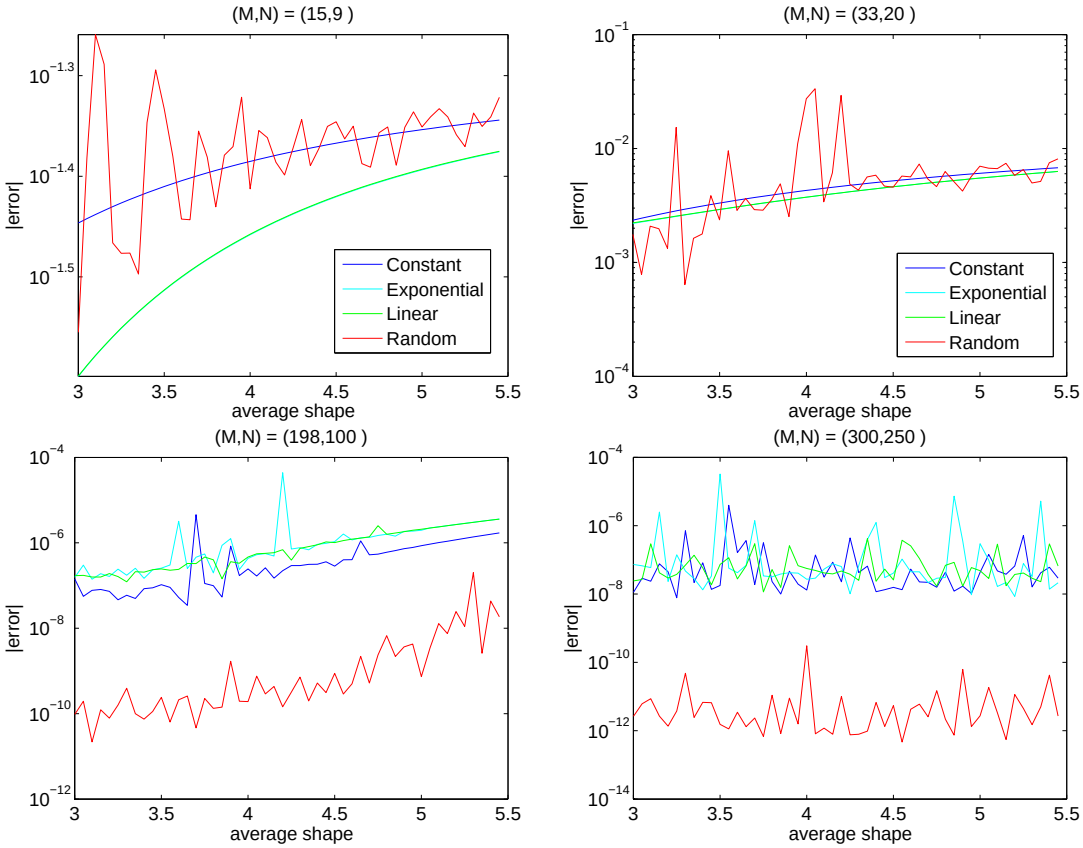

Figure 28: A comparison of shape parameters interpolating $f(x) = \exp^{\sin(\pi x)}$ for different values of M and N.

Name of Data	$(M,N) = (15,9)$	$(M,N) = (33,20)$	$(M,N) = (198,100)$	$(M,N) = (300,250)$
κ_C	297.8863	$2.5829e^{4}$	$3.1282e^{15}$	$1.5259e^{19}$
avg. $\lvert error \rvert$ for κ_C	0.0415	0.0047	$5.6040e^{-7}$	$1.6125e^{-7}$
κ_E	356.1310	$4.4916e^{4}$	$1.4584e^{17}$	$3.6191e^{19}$
avg. $\lvert error \rvert$ for κ_E	0.0354	0.0042	$2.0448e^{-6}$	$1.0591e^{-6}$
κ_L	344.7936	$4.2059e^{4}$	$1.1121e^{17}$	$3.2177e^{19}$
avg. $\lvert error \rvert$ for κ_L	0.0355	0.0042	$1.1023e^{-6}$	$1.2881e^{-11}$
κ_R	304.0434	$2.6946e^{4}$	$2.2868e^{16}$	$1.4529e^{19}$
avg. $\lvert error \rvert$ for κ_R	0.0426	0.0065	$7.1573e^{-9}$	$1.2881e^{-11}$

Table 23: Numerical information from Figure 28.

$f(x) = e^{\sin(\pi x)}$ on the interval $[-1, 1]$. For large (M, N) our theory of the variable random shape parameter still held for interpolating $f(x)$. Note: The system matrix is ill-conditioned. Given 300 evaluation points and 250 center locations, we achieve an accuracy with an average $|error|$ of $1.2881e^{-11}$

The next function we test with different shape parameters and values of (M, N) is $f(x) = \sin(50\pi x)e^{(-100(x-\frac{1}{2})^2)}$ where $x \in [0, 1]$. The graph of $f(x)$ is shown in Figure 29. We follow the same procedure as the previous comparison on this function. Figure 30 displays the different cases for the number of evaluation points and center locations. The data for the last three cases of (M, N) is seen in Table 24.

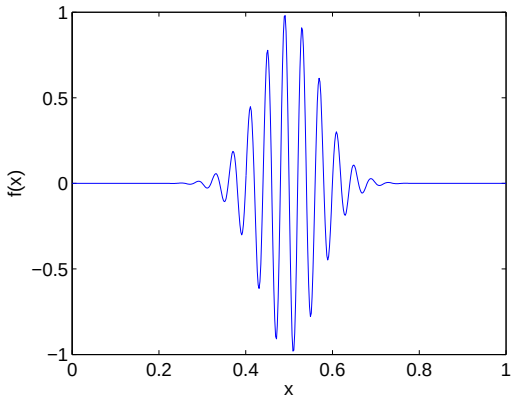

Figure 29: Graph of $f(x) = \sin(50\pi x)e^{(-100(x-\frac{1}{2})^2)}$ where $x \in [0, 1]$.

Therefore, we conclude from Table 24 and Figure 30 for large number of evaluation points and center locations, the variable random shape

Name of Data	$(M,N)=(198,100)$	$(M,N)=(398,300)$	$(M,N)=(1000,800)$
avg. κ_C	$2.5437e^{19}$	$1.1407e^{20}$	$2.7877e^{20}$
avg. $\|error\|$ for κ_C	12.1602	13.43	48.58
avg. κ_E	$1.1255e^{02}$	$1.5182e^{20}$	$2.8245e^{20}$
avg. $\|error\|$ for κ_E	66.1601	65.9403	$8.3406e^{2}$
avg. κ_L	$1.9813e^{20}$	$4.1849e^{19}$	$5.4340e^{20}$
avg. $\|error\|$ for κ_L	80.5482	$4.6426e^{2}$	57.5255
avg. κ_R	$6.1702e^{19}$	$5.61251e^{19}$	$2.1939e^{20}$
avg. $\|error\|$ for κ_R	37.2348	0.2323	0.1995

Table 24: Numerical information from Figure 30.

parameter is the best choice when interpolating

$$f(x) = \sin(50\pi x)e^{(-100(x-\frac{1}{2})^2)}.$$

We notice as the system matrix becomes more and more ill-conditioned, the value of using the variable random shape parameter for interpolation became more apparent.

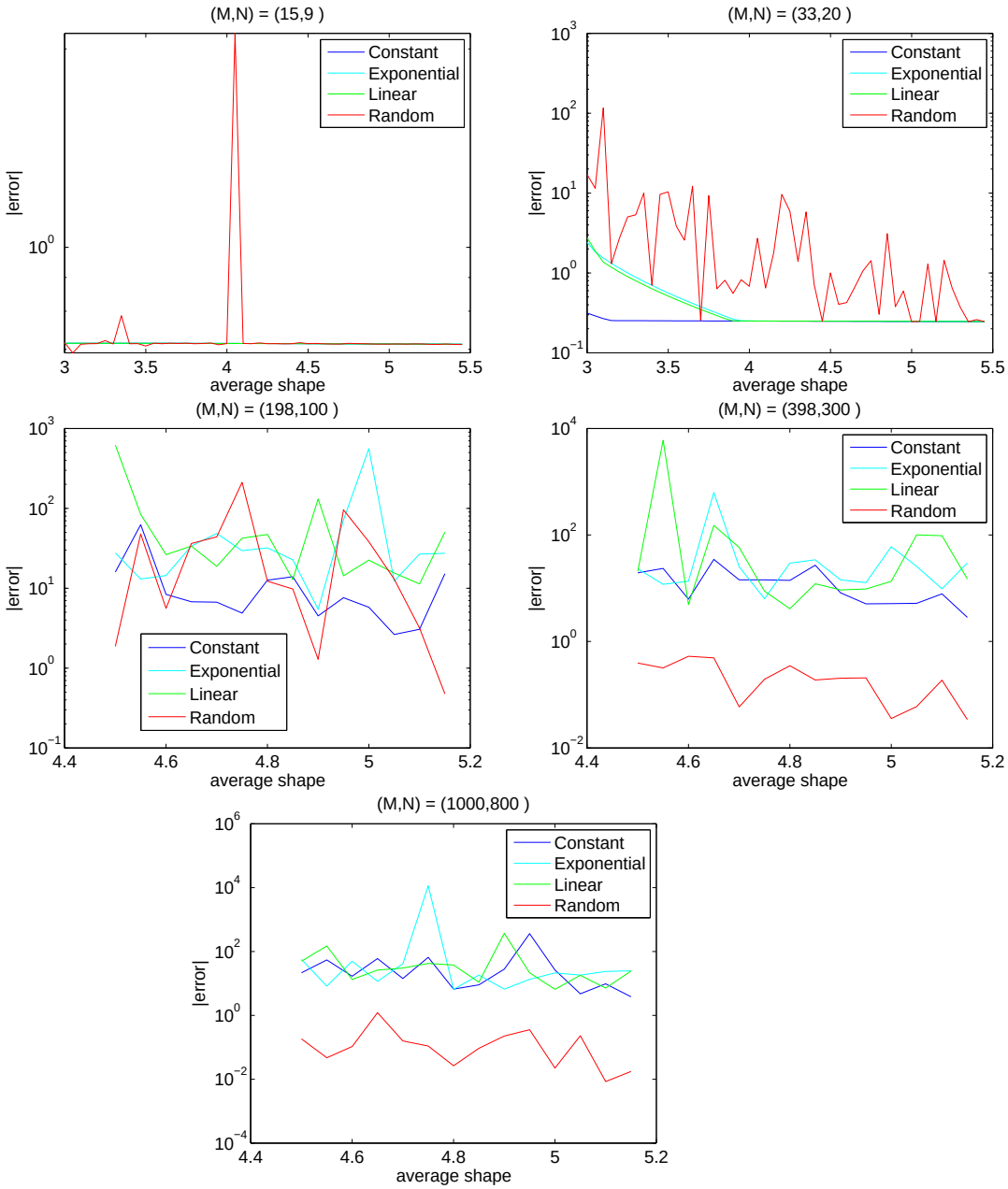

Figure 30: A comparison of shape parameters interpolating $f(x) = \sin(50\pi x)e^{(-100(x-\frac{1}{2})^2)}$ where $x \in [0, 1]$ for different values of M and N.

5 Shape Parameter Strategies in 2 Dimensions

5.1 Introduction

We have already seen by numerical examples that choosing the "optimal" shape parameter strategy does indeed affect the accuracy of interpolating various functions. In the next subsection we apply the different shape parameter strategies to an interpolation problem in two dimensions. The different shape parameter strategies that we compare are the following: constant, variable exponential, variable linear, and variable random. We use the generators for each different variable shape parameter in Listings 2, 4, and 3. **Recall**, as in Chapter 4, in order for judgement to be unbiased, it is imperative that the condition numbers of each shape parameter strategy be as close to one another as possible. After we obtain sufficiently close condition numbers for each shape parameter strategy, we evaluate them mainly for the smallest average of the $|errors|$ in interpolation. We, also, record the minimum and maximum error that each shape parameter strategy produces.

5.2 Franke Function

The Franke Function was created by Richard Franke for comparing different methods of interpolation over different surfaces [14]. The Franke Function can be described as a surface consisting of two Gaussian peaks and one Gaussian valley, where the surface slopes toward the first quadrant. The Franke function is seen in Figures 31 and 32. The definition

of the function is listed below.

Definition 10 "Franke (bivariate test) Function"- Given data in the form of (x,y), the function returns the sum of the four exponentials. The function is defined as,

$$\begin{aligned} f(x,y) &= \frac{3}{4}e^{-((9x-2)^2+(9y-2)^2)/4} + \frac{3}{4}e^{-((9x+1)^2/49-(9y+1)/10)} \\ &\quad + \frac{1}{2}e^{-((9x-7)^2+(9y-3)^2/4)} - \frac{1}{5}e^{-((9x-4)^2-(9y-7)2)} \end{aligned} \tag{22}$$

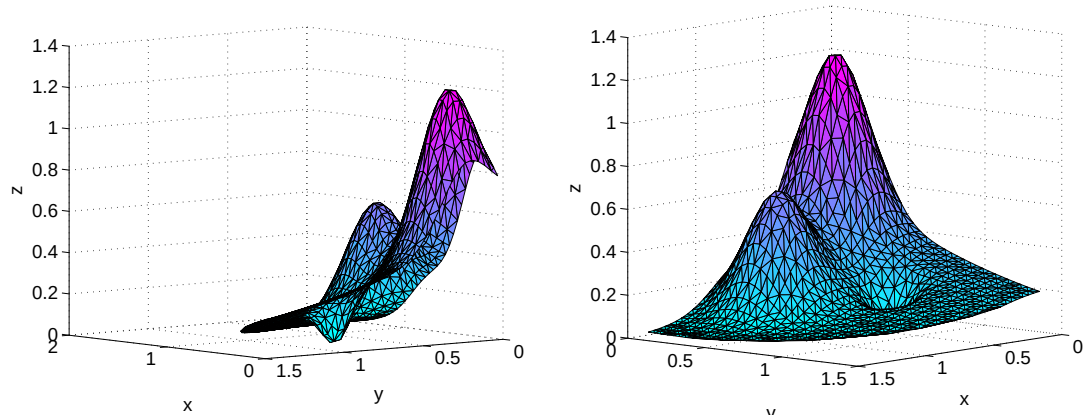

Figure 31: Some side views of the Franke Function.

For this test in two dimensions, we use the Franke Function. Matlab has a built in function for the Franke Function. The function call for Matlab, version R2008a, is **franke(x,y)**, where x and y are same size matrices. The Matlab script we use for exploring which shape parameter strategy is the sufficient is in Listing 9. The data for each

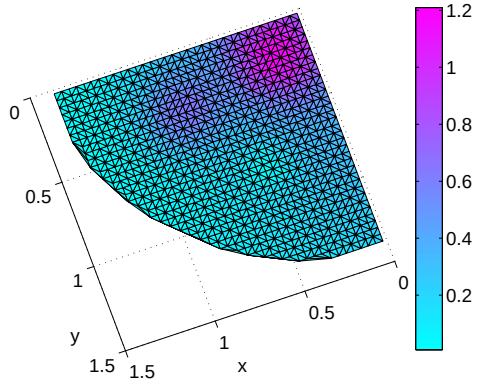

Figure 32: The top view of the Franke Function.

RBF interpolation with each different shape parameter is in Table 25.

Type of ε	(εMin,εMax)	$\kappa(B)$	Mean $\lvert error \rvert$	Max $\lvert error \rvert$	Min $\lvert error \rvert$
Constant	$\varepsilon = 3$	$1.1059e^{16}$	$2.8099e^{-7}$	$9.2320e^{-6}$	$2.2500e^{-10}$
Variable Exponential	(2.7,9)	$1.4293e^{16}$	$2.3028e^{-5}$	$7.5861e^{-4}$	$3.7478e^{-10}$
Variable Linear	(2.7,6.36)	$1.1054e^{16}$	$1.3494e^{-5}$	$3.2138e^{-4}$	$6.2487e^{-9}$
Variable Random	(2,7)	$1.1361e^{16}$	$9.1431e^{-5}$	0.00377	$4.5203e^{-8}$

Table 25: Numerical information from the interpolation of the Franke Function with different shape strategies.

We see from Table 25 that using a constant shape parameter to interpret the Franke Function produce the best results. The constant shape parameter of $\varepsilon = 3$ has the least average $|error|$ and least maximum $|error|$. Also, the constant shape parameter has the most desirable minimum $|error|$ of all the shape parameter strategies. Thus, we conclude that the constant shape is the best choice when interpolating the Franke Function.

5.3 Exponential function

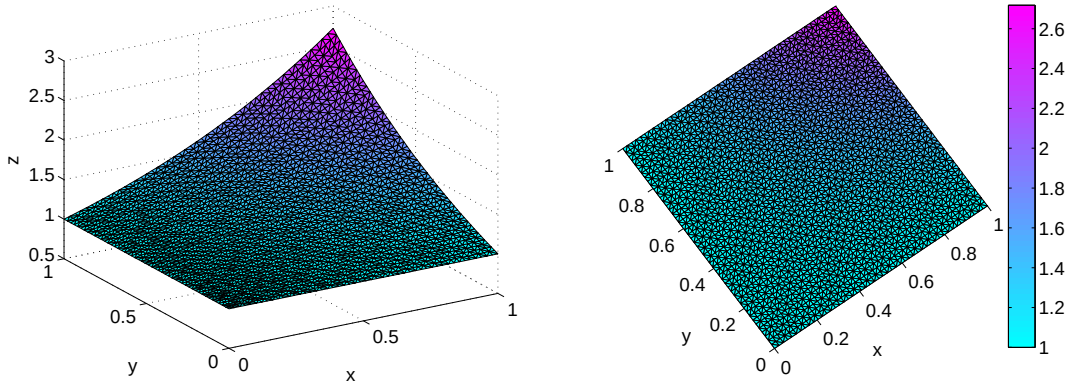

Figure 33: The top and side views for $f(x, y) = e^{xy}$ where $x \in [0, 1]$ and $y \in [0, 1]$.

In this subsection we numerically determine which shape parameter strategy is the best use when interpolating an exponential function. In this comparison we vary the number of evaluation points and centers to see which shape parameter strategy is the best. The two dimensional function for interpolation is $f(x, y) = e^{xy}$ where $x \in [0, 1]$ and $y \in [0, 1]$. The graph of $f(x, y)$ is seen in Figure 33.

The data we gather from the numerical experiments is shown in a variety of different ways. Tables 26, 27, and 28 display the condition number of the system matrix, the average $|error|$, and (M, N) for each of the different shape parameter strategies. Figure 34 is the plot of the maximum error that occurs for each shape parameter strategy for an average value shape parameter. That is, the average value of ε

denoted by $\varepsilon_{avg.} = \frac{\varepsilon Min}{\varepsilon Max}$. The Matlab code that produced Figure 34 is very similar to Listing 7, so we do not include the Matlab code in this thesis. However, we include the Matlab code used for data collection of Tables 26, 27, and 28. The code is in Listing 8.

Shape(εMin,εMax)	$\kappa(B)$	Max $\lvert error \rvert$	Avg. $\lvert error \rvert$
Constant, $\varepsilon = 3$	$1.1271e^{14}$	$8.6064e^{-5}$	$1.5832e^{-6}$
Exponential(7,2.5)	$1.9477e^{14}$	0.0023	$2.6131e^{-5}$
Linear(7,2.462)	$1.1124e^{14}$	0.0023	$2.6939e^{-5}$
Random(7,2.2)	$1.5586e^{14}$	0.0096	$2.0058e^{-4}$

Table 26: Numerical information from the interpolation of $f(x, y) = e^{xy}$ for $(M, N) = (25, 19)$.

Shape(εMin,εMax)	$\kappa(B)$	Max $\lvert error \rvert$	Avg. $\lvert error \rvert$
Constant, $\varepsilon = 3$	$1.2090e^{20}$	$2.1704e^{-6}$	$3.1507e^{-8}$
Exponential(7,2.5)	$1.6779e^{20}$	$6.8161e^{-5}$	$3.9212e^{-7}$
Linear(7,2.3)	$1.0630e^{20}$	$9.9538e^{-5}$	$4.7203e^{-7}$
Random(7,2.3)	$1.5193e^{20}$	$7.3561e^{-7}$	$4.3251e^{-9}$

Table 27: Numerical information from the interpolation of $f(x, y) = e^{xy}$ for $(M, N) = (50, 38)$.

Shape(εMin,εMax)	$\kappa(B)$	Max $\lvert error \rvert$	Avg. $\lvert error \rvert$
Constant, $\varepsilon = 3$	$4.1930e^{20}$	$1.9986e^{-6}$	$1.9521e^{-8}$
Exponential(8,2.3)	$3.5305e^{20}$	$6.7571e^{-5}$	$2.6674e^{-7}$
Linear(8,2.2)	$3.5978e^{20}$	$5.8040e^{-4}$	$1.5083e^{-6}$
Random(7,2.3)	$4.4497e^{20}$	$5.1810e^{-7}$	$1.6963e^{-9}$

Table 28: Numerical information from the interpolation of $f(x, y) = e^{xy}$ for $(M, N) = (100, 76)$.

We deduce from Tables 27 and 28 and Figure 34 for (M, N) large ((M,N) $\geq (50, 38)$) that the variable random shape parameter strategy produce the least $\lvert error \rvert$ and minimized the maximum error. Also note, when the system matrix becomes severely ill-conditioned that the ran-

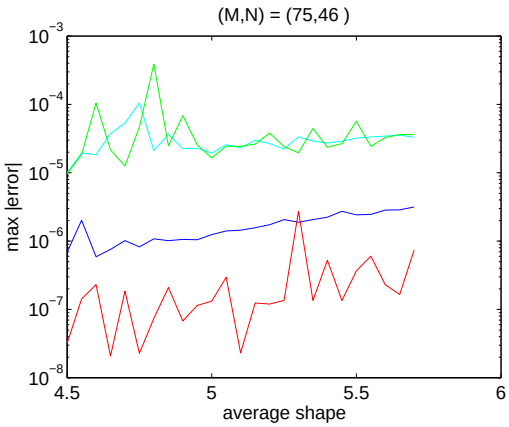

Figure 34: This graphs plots the $\varepsilon_{avg.}$ vs. the maximum $|error|$ for interpolation of $f(x, y) = e^{xy}$ with different shape parameter strategies.

dom shape produces the best interpolation. Therefore, we conclude the random variable shape parameter is the best choice when interpolating (N large) for an exponential function.

5.4 Polynomial Function

Next we compare the variable shape parameter strategies in another two dimensional function. The function that we use in this test is the polynomial function $f(x, y) = x^2 + y^3$ where $x \in [0, 1]$ and $y \in [0, 1]$. For this test we proceed as follows: first, we only compare a small and large case of (M, N); and secondly, we will plot the large case of (M, N) Figure 34. The first part of the test is supported by Tables 29 and 30, while the second part by Figure 36. For Tables 29 and 30, the constant shape parameter $\varepsilon = 3$ is used and (M, N) equal to (25,19) and (50,38). The data for the tables is produced by the code in Listing 8.

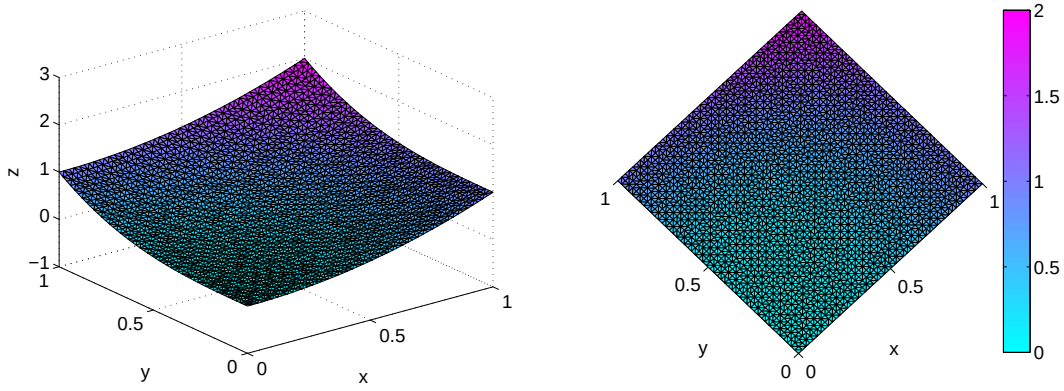

Figure 35: The top and side views for $f(x,y) = x^2 + y^3$ where $x \in [0,1]$ and $y \in [0,1]$.

Shape(εMin,εMax)	$\kappa(B)$	Max $\lvert error \rvert$	Avg. $\lvert error \rvert$
Constant, $\varepsilon = 3$	$1.1271e^{14}$	$8.6064e^{-5}$	$1.5832e^{-6}$
Exponential(7,2.56)	$1.1501e^{14}$	0.0012	$1.6796e^{-5}$
Linear(7,2.46)	$1.1312e^{14}$	0.0013	$1.8511e^{-5}$
Random(7.1,2)	$1.0867e^{14}$	$3.1223e^{-4}$	$6.1470e^{-6}$

Table 29: Numerical information from the interpolation of $f(x,y) = x^2 + y^3$ for $(M,N) = (25,19)$.

Shape(εMin,εMax)	$\kappa(B)$	Max $\lvert error \rvert$	Avg. $\lvert error \rvert$
Constant, $\varepsilon = 3$	$1.2090e^{20}$	$1.3693e^{-6}$	$1.9353e^{-8}$
Exponential(7,2.1)	$1.1824e^{20}$	$5.7170e^{-5}$	$5.8625e^{-7}$
Linear(7,2.3)	$1.0630e^{20}$	$3.8094e^{-5}$	$1.7803e^{-7}$
Random(7,2.3)	$3.6073e^{20}$	$2.0069e^{-7}$	$1.1799e^{-9}$

Table 30: Numerical information from the interpolation of $f(x,y) = x^2 + y^3$ for $(M,N) = (50,38)$.

We conclude from Table 30 and Figure 36 that as (M,N) gets larger, the variable random shape parameter produces the best accuracy and has the least maximum error. We also note as the system matrix

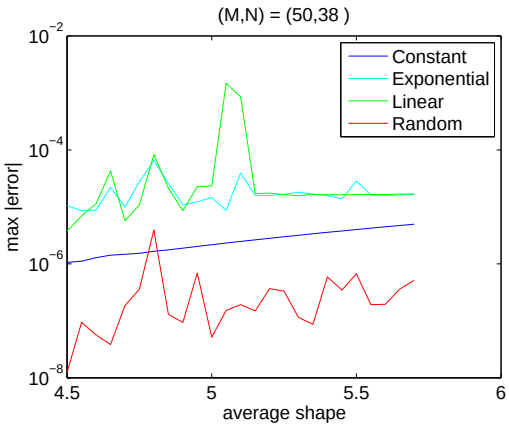

Figure 36: This graphs plots the $\varepsilon_{avg.}$ vs. the maximum $|error|$ for interpolation of $f(x, y) = x^2 + y^3$ with different shape parameter strategies.

Shape Parameter Strategy	$\kappa(B)$	Avg. $\lvert error \rvert$
Constant	1.031e+016	4.1611e-006
Exponential	$1.5136e^{20}$	$4.1702e^{-5}$
Linear	$7.9902e^{19}$	$1.2059e^{-4}$
Random	$2.8001e^{19}$	$8.6921e^{-7}$

Table 31: Information from Figure 36.

becomes ill-conditioned, using the variable random shape parameter to interpolate produces the best results.

5.5 Sinusoidal Function

The next two dimensional function we compare different shape parameter strategies is a sinusoidal function. The function is defined by $f(x, y) = sin(.25\pi x) + sin(e^{\pi y})$ where $(x, y) \in [0, 1]$. We use this sinusoidal function for comparison of shape parameter since it varies drastically in elevation throughout the Figure 37. The value of the

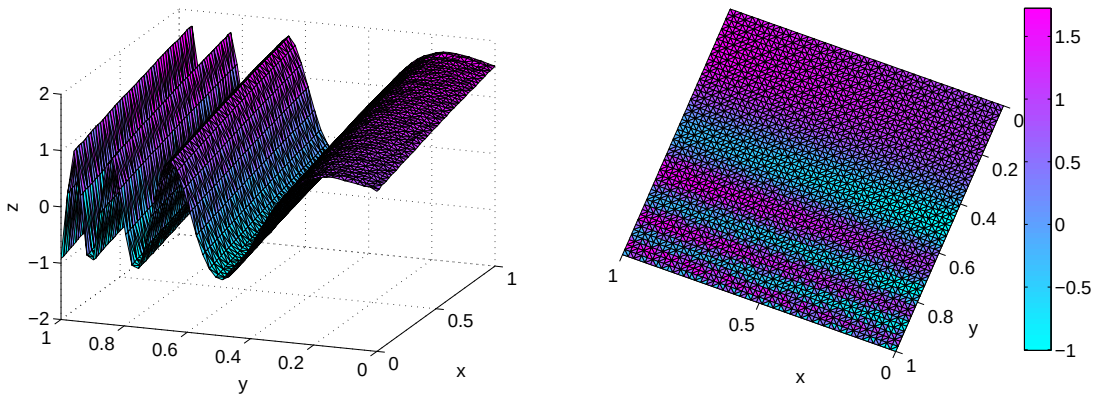

Figure 37: The top and side views for $f(x,y) = sin(.25\pi x) + sin(e^{\pi y})$ where $x \in [0,1]$ and $y \in [0,1]$.

constant shape parameter is 3, and the values of (M,N) are as follows: (25,19), (50,38), and (60,47). The data from the interpolations of $f(x,y)$ for different values of (M,N) are shown in Tables 32, 33 and 34. We also interpret Figure 38 to further determine which shape parameter is the best for this type of function. Data of this figure is in Table 35.

Shape(εMin,εMax)	$\kappa(B)$	Max $\lvert error \rvert$	Avg. $\lvert error \rvert$
Constant, $\varepsilon = 3$	$1.1271e^{14}$	3.3138	0.0443
Exponential(7,2.56)	$1.1501e^{14}$	2.5836	0.0452
Linear(7,2.459)	$1.1416e^{14}$	2.5296	0.0454
Random(8,2)	$2.4488e^{14}$	25.3314	0.3589

Table 32: Numerical information from the interpolation of $f(x,y) = sin(.25\pi x) + sin(e^{\pi y})$ for $(M,N) = (25,19)$.

We conclude from Tables 34 and 35 and Figure 38, that for large number of centers, the variable random shape parameter is the best

Shape(εMin,εMax)	$\kappa(B)$	Max $\lvert error \rvert$	Avg. $\lvert error \rvert$
Constant, $\varepsilon = 3$	$1.2090e^{20}$	0.4250	0.0047
Exponential(7,2.1)	$1.1824e^{20}$	2.3759	0.0197
Linear(7,1.7)	$1.7760e^{20}$	54.9480	0.2341
Random(7,1.7)	$1.0781e^{20}$	7.3745	0.0494

Table 33: Numerical information from the interpolation of $f(x, y) = sin(.25\pi x) + sin(e^{\pi y})$ for $(M, N) = (50, 38)$.

Shape(εMin,εMax)	$\kappa(B)$	Max $\lvert error \rvert$	Avg. $\lvert error \rvert$
Constant, $\varepsilon = 3$	$6.334e^{20}$	1.9277	0.0251
Exponential(7,2)	$6.0793e^{20}$	567.7671	4.7354
Linear(7,1.7)	$1.8723e^{20}$	21.1374	0.1017
Random(7,1.7)	$3.1530e^{20}$	0.3000	0.0014

Table 34: Numerical information from the interpolation of $f(x, y) = sin(.25\pi x) + sin(e^{\pi y})$ for $(M, N) = (60, 47)$.

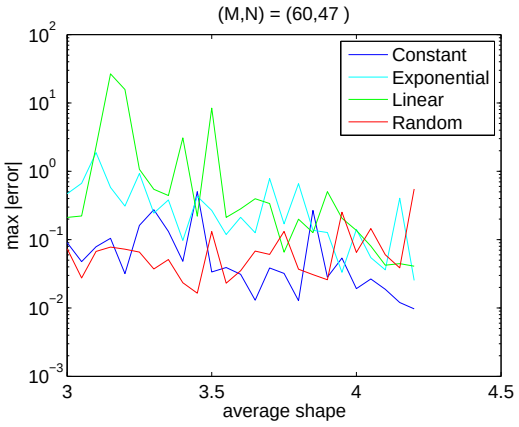

Figure 38: This graphs plots the $\varepsilon_{avg.}$ vs. the maximum $\lvert error \rvert$ for interpolation of $f(x, y) = sin(.25\pi x) + sin(e^{\pi y})$ for $(M, N) = (60, 47)$ with different shape parameter strategies.

choice when interpolating this sinusoidal function.

Shape Parameter Strategy	$\kappa(B)$	Avg. $\lvert error\rvert$
Constant	1.746802547574497e+020	0.084361074062004
Exponential	2.127456115364558e+020	0.371312591642159
Linear	1.977274000119941e+020	2.462959991315436
Random	4.872398891594948e+020	0.086761541572835

Table 35: Information from Figure 38.

5.6 Constant function

The last function we use to test shape parameters in two dimensions is the constant function. The function is $f(x, y) = 1$ where $(x, y) \in [0, 1]$. We first show the type of shape parameter, condition number, maximum $\lvert error\rvert$, and average $\lvert error\rvert$ for different values of M and N. This data is shown in the tables below. Next, we will plot the $\lvert error\rvert$ versus the average shape for the last case of M and N in Figure 39. Figure 39 is used to reinforce our conclusion of which shape parameter is the best in this comparison. The constant shape parameter value $\varepsilon = 3$ is used. The values for εMin and εMax is in the tables below.

Shape(εMin,εMax)	$\kappa(B)$	Max $\lvert error\rvert$	Avg. $\lvert error\rvert$
Constant, $\varepsilon = 3$	$1.1271e^{14}$	$1.8491e^{-5}$	$4.1921e{-}7$
Exponential(8,2.545)	$1.1231e^{14}$	$5.1664e^{-4}$	$7.9732e^{-6}$
Linear(8,2.4294)	$1.0974e^{14}$	$5.0566e^{-4}$	$7.9263e^{-6}$
Random(7.5,2)	$1.2386e^{14}$	$8.8182e^{-5}$	$1.6024e^{-6}$

Table 36: Numerical information from the interpolation of $f(x, y) = 1$ for $(M, N) = (25, 19)$.

Documentation from Tables 37, 38 and Figure 39, we see that for large (M, N) and an ill-conditioned system matrix support that the variable random shape parameter produces the best results.

Shape(εMin,εMax)	$\kappa(B)$	Max $\lvert error\rvert$	Avg. $\lvert error\rvert$
Constant, $\varepsilon = 3$	$1.2090e^{20}$	$4.1590e^{-7}$	$6.1654e^{-9}$
Exponential(7.5,2.1)	$1.6962e^{20}$	$1.9169e^{-6}$	$1.4117e^{-8}$
Linear(8,2.4294)	$1.8831e^{20}$	$6.4694e^{-7}$	$3.2763e^{-9}$
Random(7.5,2)	$3.0109e^{20}$	$1.9573e^{-9}$	$1.3109e^{-11}$

Table 37: Numerical information from the interpolation of $f(x, y) = 1$ for $(M, N) = (50, 38)$.

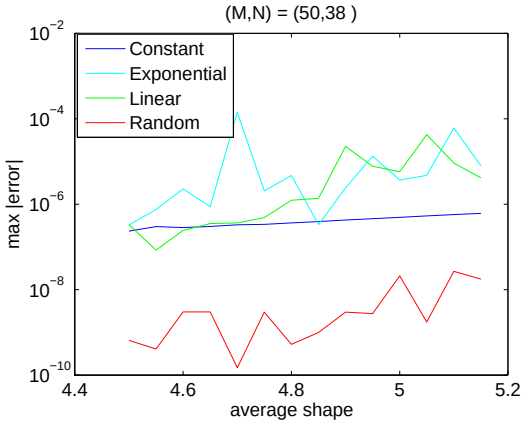

Figure 39: Plots of the $\varepsilon_{avg.}$ vs. the maximum $\lvert error\rvert$ for interpolation of $f(x, y) = 1$ for $(M, N) = (50, 38)$ with different shape parameter strategies.

Shape Parameter Strategy	$\kappa(B)$	Avg. $\lvert error\rvert$
Constant	$2.94371e^{17}$	$4.01982e^{-7}$
Exponential	$1.9282e^{20}$	$1.7579e^{-5}$
Linear	$1.0749e^{20}$	$6.8759e^{-6}$
Random	$4.8545e^{20}$	$6.03334e^{-9}$

Table 38: Information from Figure 39.

5.7 Conclusions of 2-D Interpolation

We conclude from the previous subsections that when interpolating a two dimensional function involving large number of centers and an ill-conditioned system matrix that the variable random shape parameter

produces the best results. It is documented, numerically, that when the variable random shape parameter is not the best choice one can only increase the number of centers to alleviate this situation. In conjunction with the Franke Function, the author suggests a drastic increase in the number of evaluation points and centers to make the variable random shape the best choice.

6 Reducing Errors Near Boundaries

6.1 Introduction to Boundary Error

The Radial Basis Function approximation method is a "satisfactory" method to interpolate sparse and scattered data. As we look back in Chapter 4, we see that the MQ RBF method displays very low pointwise errors in the center of the interval; however, near the boundaries the errors were larger. Techniques for reducing boundary errors are relatively new. There is some research that addresses the boundary error issue to date [3, 10, 11]. In reference [10] the authors suggests many intriguing methods. Some methods that were suggested in their papers were the following: adding polynomial terms to RBF expansion, Super Not-a-Knot (modified version of Not-a-Knot), boundary treatment, and clustering the center locations close to the boundary. In reference [11], the authors suggest initiating a series of conditions that are imposed on the interpolation that connect the initial and boundary data. It has been shown in [36] that clustering the center locations too much near the boundaries will reduce the accuracy of the interpolation in the interior of the region.

In this section we apply a variable mixed shape parameter strategy with the goal of reducing errors in boundary regions.

6.2 Variable Random Numerical Trial 1

In our first numerical trial, we experiment with using a variable shape parameter on a percentage of the area near the endpoints of the interval with a constant shape in between. We compare this new shape parameter strategy to that of a constant shape parameter and a variable random shape parameter. We look for a reduction of the point-wise errors near the boundary of the interval. After the initial trial, we introduce different percentages of boundary region coverage to see if we can improve on the reduction of error.

The first function we try to reduce boundary errors is $f(x) = -arctan(5(x+\frac{1}{2}))$. The constant shape parameter that is used throughout this section is, ε=5. We use the MQ method to interpolate $f(x)$ with $N = 100$ centers and $M = 198$ evaluation points. First, we use a variable random shape parameter on 30 percentage of the interval near the left and right boundaries. Thus, 60 shape parameters are generated by (21) and the rest will be constant.

The Figure 40 shows the absolute values of the point-wise errors of interpolation of $f(x)$. The red line corresponds to the constant shape interpolation, while the blue line corresponds to the mixed shape parameter. The values for εMin and εMax are 3 and 8. Table 39 displays further information about the experiment. The table also includes different values for the percentage of the interval that contains the random shape parameter.

From Table 39 and Figure 40, we see that if we use the variable

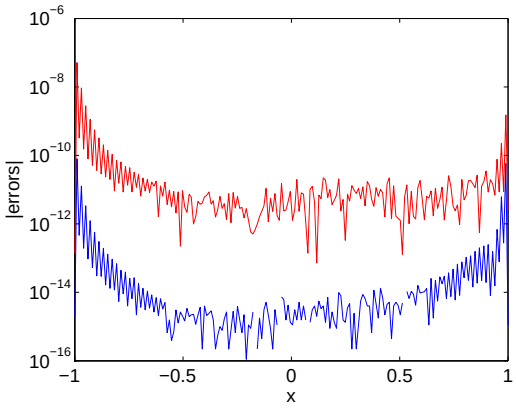

Figure 40: Reducing boundary errors with a mixed random shape parameter for $f(x) = -arctan(5(x + \frac{1}{2}))$.

ε	Percent of Interval	Kappa	Maximum Error	Average of $\lvert error \rvert$
5	NA	$4.1018e^{16}$	$5.1080e^{-8}$	$3.5241e^{-10}$
Mixed$(3, 10), \varepsilon = 5$	1^{th} left and right	$3.9266e^{16}$	$1.3845e^{-8}$	$1.6086e^{-10}$
Mixed$(3, 10), \varepsilon = 5$	2.5^{th} left and right	$3.9624e^{16}$	$2.9097e^{-8}$	$1.8812e^{-10}$
Mixed$(3, 8), \varepsilon = 5$	5^{th} left and right	$4.2044e^{16}$	$3.4125e^{-9}$	$2.5308e^{-11}$
Mixed$(3, 10)$, $\varepsilon = 5$	10^{th} left and right	$3.1834e^{16}$	$2.1704e^{-9}$	$2.4132e^{-11}$
Mixed$(3, 8), \varepsilon = 5$	20^{th} left and right	$3.4507e^{16}$	$6.8518e^{-9}$	$5.8918e^{-11}$
Mixed$(3, 10)$, $\varepsilon = 5$	30^{th} left and right	$5.3094e^{16}$	$7.9814e^{-11}$	$8.4109e^{-13}$
Mixed$(3, 10), \varepsilon = 5$	40^{th} left and right	$4.2543e^{16}$	$1.124e^{-9}$	$6.8183e^{-12}$
Mixed$(3, 10), \varepsilon = 5$	45^{th} left and right	$4.5402e^{16}$	$2.0075e^{-9}$	$1.5204e^{-11}$
Mixed$(3, 10), \varepsilon = 5$	47.25^{th} left and right	$4.0309e^{16}$	$8.51e^{-10}$	$9.2456e^{-12}$
"Full" Random$(3, 10)$	100^{th}	$4.5402e^{16}$	$2.0075e^{-9}$	$1.5204e^{-11}$

Table 39: Decreasing boundary errors using a mixed random shape parameter numerical experiment 1.

random shape parameter on the 30^{th} left and right percent of the interval we are able to reduce the maximum error and reduce the average $|error|$ the best. Notice, that using a "full" variable random shape to interpolant $f(x)$ did reduce the maximal and average $|error|$ compared to that of the constant shape parameter. Also, it didn't do as well as the mixed variable shape parameter.

6.3 Variable Random Numerical Trial 2

The next experiment involves the same procedure as in the subsection **Variable Random Numerical Trial 1**. The function that we try to reduce boundary errors is $f(x) = e^{sin(4\pi x)}$ on the interval [-1,1]. We use the constant shape parameter, $\varepsilon = 3$. Since we used (21) to generate the different shape parameter, the values of εMin and εMax did change from trial to trial. We could not keep εMin and εMax the same throughout the mixed shape parameter, since we needed to keep the condition numbers of each system matrix relatively close. The different values for εMin and εMax are in Table 40.

Figure 41 shows the absolute values of the point-wise errors of the interpolation. The red line corresponds to the constant shape parameter, and the blue line is the mixed random shape parameter. In Figure 41, the constant shape is $\varepsilon = 3$ and the percentage mixed random shape parameter that contained a random variable shape parameter was 10 percent left and 10 percent right. The number of centers, N, and the number of evaluation points, M, did not change from the previous experiment. Table 40 displays additional experiments with different percentages of the interval covered with the variable random shape parameter. For this numerical trial, the values of εMin and εMax are closer to the constant shape parameter, than in that of **Variable Random Numerical Experiment 1**.

In Table 40, we illustrate yet again that the mixed variable/random shape strategy has the best accuracy. Here, the 40^{th} percentage left

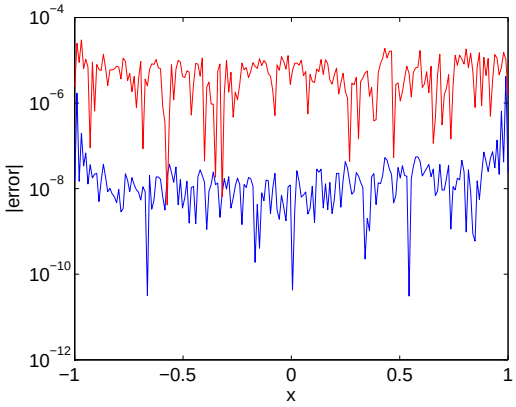

Figure 41: Reducing boundary errors with a mixed random shape parameter for $f(x) = e^{sin(4\pi x)}$.

ε	Percent of Interval	Kappa	Max Error	Avg. of $\lvert error \rvert$
3	NA	$1.8927e^{19}$	$2.9805e^{-5}$	$5.8462e^{-6}$
Mixed$(2,5), \varepsilon = 3$	10^{th} left and right	$1.9146e^{19}$	$4.1908e^{-6}$	$5.0504e^{-8}$
Mixed$(2,6), \varepsilon = 3$	20^{th} left and right	$1.9041e^{19}$	$5.0003e^{-6}$	$4.6138e^{-8}$
Mixed$(2,7), \varepsilon = 3$	30^{th} left and right	$1.8276e^{19}$	$2.8276e^{-5}$	$2.9871e^{-7}$
Mixed$(2,7), \varepsilon = 3$	40^{th} left and right	$1.7791e^{19}$	$1.2502e^{-6}$	$1.6056e^{-8}$
Mixed$(2,7), \varepsilon = 3$	45^{th} left and right	$1.7791e^{19}$	$1.2501e^{-6}$	$1.6056e^{-8}$
Mixed$(2,7), \varepsilon = 3$	47.5^{th} left and right	$1.8276e^{19}$	$2.8276e^{-5}$	$2.9871e^{-7}$
Full Random$(2,7)$	100^{th}	$1.7650e^{19}$	$1.1344e^{-4}$	$8.0002e^{-7}$

Table 40: Decreasing boundary errors us a mixed random shape parameter Numerical Trial 2.

and right mixed random variable shape parameter produced the least average and maximal $|error|$.

6.4 Conclusions

We see in the two numerical experiments that using a mixed random/-constant shape parameter did significantly better than either of the variable random or constant shape parameter alone. We see that given large number of centers and evaluation points that the variable random

shape parameter produced the best approximation for a function given evenly spaced centers. Now we have an "insight" into what this obscure shape parameter strategy can accomplish. In future research, we will further investigate reducing boundary errors with a mixed random/-constant shape parameter for evenly and not evenly spaced centers.

7 Conclusions

We found in many cases where using a variable shape parameter did produce a better interpolation than using a constant shape parameter. In Chapter 3, we introduced the MQ RBF. We illustrated the MQ RBF produced better results than using polynomial interpolation. Chapter 4 introduced us to the topic of variable shape parameters. The first section of Chapter 4, we demonstrated for all the different shape parameter strategies, the variable random shape parameter produced the best accuracy. The second part of Chapter 4, we further solidified the idea of the variable random shape parameter by comparing it with different shape parameter strategies involving different functions. Chapter 4 concluded with the idea of using the variable random shape parameter for large (M, N) and evenly spaced evaluation points produced the best results when interpolating data or a function. The next chapter, we expanded the idea of shape parameter into two dimensions. In this chapter, the variable random shape parameter again produced the best interpolation of all comparisons, but one. For that one comparison, the author suggested a drastic increase in the number of centers and evaluation points, resulting in the variable random shape parameter to be the best choice. The last chapter, Chapter 6, we explored the topic of reducing boundary errors. We concluded from Chapter 6 with the idea of using a mixed variable random shape parameter reduced boundary error in interpolation. We saw from Sections 6.2 and 6.3

that the variable mixed random shape reduced boundary errors, compared to the interpolations using a constant or variable random shape parameter. From this thesis one concluded from numerical trials, the concept of a variable random shape parameter to interpolate data with (M, N) large is the best choice over other shape parameter strategies.

The author would like to further explore the topic of variable shape strategies where the number of centers and evaluation points are not evenly spaced. The author suggests using the variable random shape for differentiation of functions might produce better accuracy than a constant shape parameter. The topic of using variable shapes is a excellent tool for interpolation that needs to be further explored by others, as well as this author.

Matlab Code

This appendix is devoted to the Matlab code that was used in this thesis. The code will be ordered by their listing number. We removed some of the listings in the main body of the thesis in order to make the reading flow better.

Listing 5: stationaryInterpolationFP.m algorithm

```
%general Matlab code for the stationary interpolation
%can be modified to produce given result

   N = --;%N stands for centers
   M = --;%M stands for number of evaluation point
  xc = linspace(0,1,N)';%creates N number of equal spaced points
   x = linspace(0,1,M)';%between 0 and 1
CMax = --;%CMax and CMin declared for variable shape parameters
CMin = --;

%--Declare your function for interpolating and call the desired shape
%--function
        f = exp(sin(pi*xc));%define function for interpolating
   fExact = exp(sin(pi*x));%for error analysis
    shape = --%call the desired variable shape function



%--Reshape the given shape matrix above so it can be used in the system
%--and evaluation matrix
    shapetemp1 = reshape shape matrix for system matrix
    shapetemp2 = reshape shape matrix for evaluation matrix

%--Create system and evaluation matrices and then interpolate
    B = systemMatrix(xc, shapetemp1);
    H = evaluationMatrix(xc, x, shapetemp2);
```

```
        lambda = B\f;
       fApprox = H*lambda;

%---Repeat the above process of reshaping your constant shape parameter,
%---create the system and evaluation matrices, and then solve

%---From here you could plot error bounds, plot fApprox vs fExact, find
%---point-wise errors, and so forth
```

Listing 6: nonStationaryInterpolationFP.m algorithm

```
% Name of file :nonStationaryInterpolationFP.m
% Used for non-stationary interpolation of a given function with the
%   MQ RBF as the interpolant.
% Can be modified to give desired results.
% This is the GENERAL algorithm.

      M = --;%Declare your desired number of evaluation points
      x = linspace(-1,1,M)';%column vector of M evenly spaced points
                            %evaluation points from -1 to 1
      n = 10:10:250;% This differs from stationary interpolation. Here
                    %n takes on the values of 10 to 250 in intervals
                    %of 10, for use of N later. N represents the number
                    %of centers
   CMin = --;%CMin and Cmax values used for the variable shape
   CMax = --;%parameter

%---Declare the function you wish to interpolate in terms of x
 fExact = --;%Declared your exact function in terms of x

%---Create for loop, from i=1 to i=length(n). This section is used for
%---the variable shape parameter interpolation. The interpolation using
%---a constant shape is listed below
for i = 1:length(n)
        N = n(i);%ith element of the row vector n
```

```
    shape = --;%call desired variable shape generator
       xc = linspace(-1,1,N)';%column vector of N evenly spaced points
                               %from -1 to 1
        f = --;%Just fExact now in terms of xc

%--Reshape the given shape matrix above, so it can used in the system
%--matrix and evaluation matrix
shapetemp1 = reshape shape matrix for system matrix
shapetemp2 = reshape shape matrix for evaluation matrix

%--Create system and evaluation matrices and then solve
         B = systemMatrix(xc,shapetemp1);
         H = evaluationMatric(xc,x,shapetemp2);
    lambda = B\f;
   fApprox = H*lambda;

--END OF FOR LOOP--

%--Now use a constant shape parameter. Need to use another for loop
%--from i=1 to i=length(n)
shapeConstant = --;%constant shape parameter

for i = 1:length(n)
         N = n(i);%ith element of the row vector n
        xc = linspace(-1,1,N)';%column vector of N evenly spaced points
                               %from -1 to 1
         f = --;%Just fExact now in terms of xc

%--Create system and evaluation matrices and then solve
      B_Constant = systemMatrix(xc,shapeConstant);
      H_Constant = evaluationMatrix(xc,x,shapeConstant);
          lambda = B_Constant\f;
fApprox_Constant = H_Constant*lambda;

--END OF FOR LOOP--
```

```
%—From here you could plot error bounds, plot fApprox vs fExact, find
%—point-wise errors, and so forth
```

Listing 7: errorLoop.m

```
% For error analysis of shape parameters.
% color explanation: blue is constant, exponential is cyan,
% green is linear, and random is red in the graph
% named : errorLoop
% Input: M, N, CMin, CMax, ItterateLower, and ItterateHigher
% Output: Average error for the different shape parameters, condition
% numbers, and plots |error| vs the average shape.
% This function displays the error analysis for the diffent shape
% parameter strategies.

function  errorLoop(z, M, N, CMin, CMax, IterateLower, IterateHigher)
warning off;
format long;

%preallocating space for the error matrices below.
    shapeAvg = ones([1,z]);
    maxErrorConstant = ones([1,z]);
    maxErrorExponential = ones([1,z]);
    maxErrorLinear = ones([1,z]);
    maxErrorRandom = ones([1,z]);

%linspace divides the interval [-1,1] into x amount of evenly spaces
%numbers
        xc = linspace(-1,1,N)';
        x = linspace(-1,1,M)';

        f = —%Declare your function in terms of xc
        fExact = —%Declare your function in terms of x
```

```
  for j = 1:z

%CAverage is used for plotting the errors and as the value of the
%constant shape parameter
       CAverage = (CMin + CMax)/2;
       shapeAvg(j) = CAverage;

       %-------------------------------------
       %| FOR CONSTANT INTERPOLATION
       %-------------------------------------
       shapeConstant=ones([1 N]).*CAverage;
       ctemp1 = shapeConstant;
       ctemp2=shapeConstant;

       cstarConstant=repmat(ctemp1,N,1);
       B_Constant = systemMatrix(xc,cstarConstant);
       cstar2Constant=repmat(ctemp2,M,1);
       H_Constant = evaluationMatrix(xc,x,cstar2Constant);
       lambda = B_Constant\f;
       fApprox_Constant = H_Constant*lambda;
       %store the error in an array
       maxErrorConstant(j) = norm(fApprox_Constant - fExact,inf);

       %-------------------------------------
       %| FOR EXPONENTIAL INTERPOLATION
       %-------------------------------------
        shapeExponential = expShapeFunction(CMax,CMin,N);
        ctemp3 = shapeExponential;
        ctemp4 = shapeExponential;

        cstar3=repmat(ctemp3,N,1);
        B_Exponential = systemMatrix(xc,cstar3);
        cstar4=repmat(ctemp4,M,1);
       H = evaluationMatrix(xc,x,cstar4);
        lambda = B_Exponential\f;
```

```
 fApprox_Exponential = H*lambda;
 %store the error in an array
 maxErrorExponential(j) = norm(fApprox_Exponential - fExact,inf);

%-------------------------------------
%| FOR LINEAR INTERPOLATION
%-------------------------------------
 shapeLinear = LinearVariableShapeFunction(CMax,CMin,N);
 ctemp5 = shapeLinear;
 ctemp6 = shapeLinear;

 cstar5=repmat(ctemp5,N,1);
 B_Linear = systemMatrix(xc,cstar5);
 cstar6=repmat(ctemp6,M,1);
 H = evaluationMatrix(xc,x,cstar6);
 lambda = B_Linear\f;
 fApprox_Linear = H*lambda;
 %store the error in an array
 maxErrorLinear(j) = norm(fApprox_Linear - fExact,inf);

%-------------------------------------
%| FOR RANDOM INTERPOLATION
%-------------------------------------
 shapeRandom = RandomVariableShapeFunction(CMax,CMin,N);
 ctemp7 = shapeRandom;
 ctemp8 = shapeRandom;

 cstar7 = repmat(ctemp7,N,1);
 B_Random = systemMatrix(xc,cstar7);
 cstar8 = repmat(ctemp8,M,1);
 H = evaluationMatrix(xc,x,cstar8);
 lambda = B_Random\f;
 fApprox_Random = H*lambda;
 %store the error in an array
 maxErrorRandom(j) = norm(fApprox_Random - fExact,inf);
```

```
        %-------------------------------------

        CMin = CMin + IterateLower;
        CMax = CMax + IterateHigher;
    end

%call desired functions
```

Listing 8: stationary interpolation in 2-dimensions

```
function stationaryInterpolation2dFPFunction(M,N, CMax, CMin, shapeC,
    ansr, call)

warning off
format compact

%Matlab code for stationary interpolation in 2-dimensions
%involving different shape parameter strategies

%For center locations, need N, interval [0,1]
[xc,yc] = meshgrid(linspace(0,1,N),linspace(0,1,N));
xc = reshape(xc,N^2,1);
yc = reshape(yc,N^2,1);
NStar = length(xc);%could use yc instead

%For evaluation points, Need M, interval [0,1]
[x,y] = meshgrid(linspace(0,1,M),linspace(0,1,M));
x = reshape(x,M^2,1);
y = reshape(y,M^2,1);
MStar = length(x);%could use y instead

%Declare f and FExact in terms of evaluation points and
%centers--
        f = --;
```

```
        fExact = --;

%define r for the center locations and evaluation points
%so one can use the definition of the MQRBF
o = ones(1, length(xc));
r_B = sqrt((xc*o - (xc*o)').^2 + (yc*o - (yc*o)').^2);
r_H = sqrt((x*ones(1,NStar) - ones(MStar,1)*xc').^2 + ...
      (y*ones(1,NStar) - ones(MStar,1)*yc').^2);



%Different variable shape parameters-------------------------
if ansr == 2
   if call ==1
   shape = expShapeFunction(CMax,CMin,NStar);% exponential
   end
   if call ==2
   shape = LinearVariableShapeFunction(CMax,CMin,NStar);% linear
   end
   if call ==3
   shape = RandomVariableShapeFunction(CMax,CMin,NStar);% random
   end
end
%-----------------------------------------------------------


if ansr == 1
%-----------------------------------
%| FOR CONSTANT INTERPOLATION
%-----------------------------------
c = ones([1 NStar]).*shapeC;
ctemp = c;
c = repmat(ctemp,NStar,1);
B = mqRbf(r_B,c); % system matrix
lambda = B\f; % expansion coefficients
c = repmat(ctemp,MStar,1);
H = mqRbf(r_H,c); % evaluation matrix
```

```
fApprox = H*lambda; % function approximated
ConditionConstant = cond(B) % kappa
pointWiseErrorsConstant = abs(fApprox - fExact);
MaxErrorConstant = norm(fApprox - fExact)
pointWiseErrorsMeanConstant = mean(pointWiseErrorsConstant)
end


if ansr == 2
%-------------------------------------------------------------------
%For the varying shape parameter
%-------------------------------------------------------------------
c = shape;
ctemp = c;
c = repmat(ctemp,NStar,1);
B = mqRbf(r_B,c); % system matrix
lambda = B\f; % expansion coeddicients
c = repmat(ctemp,MStar,1);
H = mqRbf(r_H,c); % system matrix
fApprox = H*lambda; % function approximated
ConditionVariable = cond(B) % kappa
pointWiseErrorsVarible = abs(fApprox - fExact);
MaxErrorVariable = norm(fApprox - fExact)
pointWiseErrorsMeanVariable = mean(pointWiseErrorsVarible)
end

warning on
end
```

Listing 9: 2-D error analysis of interpolations

```
%Description: Interpolates the data from Franke Function. The center
%locations and evaluation points load from txt files. This file is used
%for my thesis of variable shape parameters.
```

```
%Input:      1.) CMax is used for shape parameter generators
%            2.) CMin is used for shape parameter generators
%            3.) Call is a number that calls the correct shape parameter
%            generator
%            4.) ShapeC designates the value of the constant shape
%            parameter
%Output:     1.) max  and min |error| of interpolation
%            2.) average |error| of interpolation
%            3.) kappa of the system matrix


function interpolationExample2dFunction(CMin, CMax, Call, ShapeC)
format long, format compact
warning off

%load the centers and evaluation points from the text files below
    xc = dlmread('frankeProblemCenters.txt',' '); %load centers
 [N,n] = size(xc); %need N(number of centers) for later use
     x = dlmread('frankeProblemEvaluationPoints.txt',' ');
         %load evaluation points
 [M,n] = size(x); %need M(number of eval. points) for later use

         f = frankeFunction(xc(:,1),xc(:,2)); %pass x and y values
    fExact = frankeFunction(x(:,1),x(:,2));
             %Exact function used in error analysis

%error checking----------------------------------------------------------
  if(ShapeC<=0)
      error('The constant shape parameter must be greater than 1!');
      end;
  if(CMax<1 || CMin<1 )
      error('CMax and CMin must be greater or equal to 1.');end;

  if(Call==1)
     b = expShapeFunction(CMax,CMin,N);end;
```

```
        %creates exponentially varying shape parameter
 if(Call==2)
    b = LinearVariableShapeFunction(CMax,CMin,N);end;
        %creates variable random shape parameter
 if(Call==3)
    b = RandomVariableShapeFunction(CMax,CMin,N);end;
        %creates variable random shape parameter
 if(Call>3 || Call<1)
    error('Call must be the number 1, 2, or 3');end;
%end of error checking-------------------------------------------------

%For the different shape parameter interpolation-----------------------

   ctemp = b;

   cstarVariable = repmat(ctemp,N,1);
   B_V = systemMatrix2d(xc,cstarVariable); %system matrix for variable
                                           %shape
   cstar2Variable = repmat(ctemp,M,1);
   H_V = evaluationMatrix2d(xc,x,cstar2Variable); %evaluation matrix
                                                  %for variable shape

   kappa_Variable = cond(B_V) %The condition number for the variable
                              %shape system matrix

   lambda_V = B_V\f; %expansion coefficients

   fApprox_V = H_V*lambda_V; %Approximation of f using the
                             %variable shape

   maxError_V = norm(fApprox_V - fExact,inf) %infinity norm
                                             %to find max
   minError_V = min(abs(fApprox_V - fExact))

   pointWiseErrors_VAverage = mean(abs(fApprox_V - fExact))
```

```
    %max_V = max(abs(fApprox_V - fExact)) Same as maxError_V

%End of interpolation ------------------------------------------------

%For the constant shape interpolation ------------------------------
    shapeConstant = ones([1 N]).*ShapeC;
    ctemp1 = shapeConstant;

    cstarConstant = repmat(ctemp1,N,1);
    B_C = systemMatrix2d(xc,cstarConstant); %system matrix for constant
                                            %shape
    cstar2Constant = repmat(ctemp1,M,1);
    H_C = evaluationMatrix2d(xc,x,cstar2Constant); %evaluation matrix
                                                   %for constant shape
    kappa_Constant = cond(B_C) %The condition number for the constant
                               %shape system matrix.
    lambda_C = B_C\f; %expansion coefficients

    fApprox_C = H_C*lambda_C; %Approximation of f using the constant
                              %shape
    maxError_C = norm(fApprox_C - fExact,inf)
                 %infinity norm to find max error for constant shape
    minError_C = min(abs(fApprox_C - fExact))

    pointWiseErrors_CAverage = mean(abs(fApprox_C - fExact))

 %End of constant shape interpolation ------------------------------
warning on
end
```

References

[1] *Application of RBF and SOFM Neural Networks on Vibration Fault Diagnosis for Aero-engines*, chapter Advances in Neural Networks, pages 414 – 419. Lecture Notes in Computer Science. Springer Berlin /Heidelberg, 2006. 1

[2] P. Baiz and M. Aliabadi. Local buckling of thin-walled structures by the boundary element method. *Engineering Analysis with Boundary Elements*, 33:302 313, 2009. 1

[3] F. Bernal. *Meshless Methods for Elliptic and Free-Boundary Problems*. PhD thesis, University of Carlos III De Madrid, 2008. 4.5, 6.1

[4] M. Bozzini, L. Lenarduzzi, M. Rossini, and R. Schaback. Interpolation by basis functions of different scales and shapes. *Calcolo*, 41(2):77 – 87, 2004. 4, 4.2

[5] M. D. Buhmann. Limits of radial basis function interpolants. *Communications on Pure and Applied Analysis*, 6(4):569 – 585, 2007. 2.3

[6] D. Higgs C. L. Bresten, S. Gottlieb and J.-H. Jung. Recovery of high order accuracy in radial basis function approximation for discontinuous problems. Preprint submitted to Elsevier Science, September 2008. 3.1

[7] T. A. Driscoll and B. Fornberg. Interpolation in the limitof increasingly flat radial basis functions. *Comput. Math. Appl.*, 43:413 – 422, 2002. 2.3

[8] J. Duchon. *Constructive Theory of Functions of Several Variables*, pages 85 – 100. Lecture Notes in Mathematics. Springer Berlin/Heidelberg, 1977. 1

[9] T. A. Foley. Near optimal parameter selection for multiquadratic interpolation. *Journal of Applied Science and Computation*, 1:54 – 69, 1994. 4.1

[10] B. Fornberg, T. Driscoll, G. Wright, and R. Charles. Obeservations on the behavior of the radial basis function approximations near boundaries. *Computers and Mathematics with Applications*, 43:473 – 490, 2002. 4.5, 6.1

[11] B. Fornberg and N. Flyer. Accurate numerical resolution of transients in intial-boundary value problems for the heat equation. *Journal of Computational Physics*, 184:526–539, 2003. 4.5, 6.1

[12] B. Fornberg and N. Flyer. The gibbs phenomenon for radial basis functions. *Sampling Theory in Signal and Imaging Processing*, 2008. 3.1

[13] B. Fornberg and J. Zuev. The runge phenonmenon and spatially variable shape parameters in rbf interpolations. *Computers and Mathematics with Applications*, 54:379 – 398, 2007. 2.3, 3.1

[14] R. Franke. *A Critical Comparison of Some Methods for Interpolation of Scattered Data*. PhD thesis, Naval Postgraduate School Monterey, California, 1979. 1, 4.1, 5.2

[15] R. Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38(157):181 – 200, 1982. 1

[16] S. Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70(350):320 – 328, 1975. 4.1

[17] D. Gottlieb and C.-W. Shu. On the gibbs phenomenon and its resolution. *SIAM Review*, 39:644 – 668, 1997. 3.1

[18] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research*, 76(8):1905 – 1915, 1971. 1, 1, 1, 4.1

[19] R. L. Hardy. Theory and applications of the multiquadric-biharmonic method: 20 years of discovery. *Computers and Mathematics with Applications*, 19(8):163 – 208, 1990. 1, 1, 1, 1, 1

[20] R. L. Hardy and W. M. Göpfert. Least squares prediction of gravity anomalies' geoidal undulations, and deflections of the vertical with multiquadric harmonic functions. *Geophysical Research Letters*, 2(10):423 – 426, 1975. 1

[21] C. S. Huang, C. F. Lee, and A. D. Cheng. Error estimate, optimal shape factor, and high precision computation of multiquadric col-

location method. *Engineering Analysis with Boundary Elements*, 31:614 – 623, 2007. 4.1

[22] E. Kansa J. Wertz and L. Ling. The role of the multiquadratic shape parameter in solving elliptic partial differential equations. *Computers and Mathematics with Applications*, 51(8):1335 – 1348, 2006. 4, 4.2

[23] J.-H. Jung. A note on the gibbs phenomenon with multiquadric radial basis functions. *Applied numerical mathematics; transactions of IMACS*, 57(2):213 – 219, 2007. 3.1

[24] E. Kansa. Multiquadratics - a scattered data approximation to computational fluid dynamics i; surface approximations and partial derivative estimates. 19(8). 4, 4.2, 4.6.2

[25] E. Kansas and R. Carlson. Improved accuracy of multiquadric interpolation using variable shape parameters. *Computers and Mathematics with Applications*, pages 98–121, 1992. 4, 4.2

[26] L. B. Lurati. Padé-gegenbauer suppression of runge phenomenon in the diagonal limit of gegenbauer approximations. *Journal of Computational Physics*, 222:1 – 8, 2007. 3.1

[27] M Kassas S. Rehman M. Mohandes, A. Balghonaim and T. Halawani. Use of radial basis functions for estimating monthly mean daily solar radiation. *Solar Energy*, 68(2):161 – 168, 2000. 1

[28] R. Mahanty and P. Gupta. Application of rbf neural network to fault classifiction and location in transmission lines. *Generation, Transmission and Distribution, IEE Proceedings*, 151(2):201 – 212, 2004. 1

[29] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2(1):11–12, 1986. 1, 1.1, 1.1, 2, 2.2, 2.4

[30] F. Louai N. Benbouza and N. Sait-Said. Application of meshless petrov galerkin (mlpg) method in electromagnetic suing radial basis functions. *Power Electronics, Machine and Drives, 4th IET Conference*, 2:650 – 655, 2008. 1

[31] M. Zaloznikb R. Vertnika and B. Sarler. Solution of transient direct-chill aluminium billet casting problem with simultaneous material and interphase moving boundaries by a meshless method. *Engineering Analysis with Boundary Elements*, 30:847 855, 2006. 1

[32] S. Rippa. Interpolation and smoothing of scattered data by radial basis functions. Master's thesis, Tel Aviv University, Israel, 1984. 1

[33] S. Rippa. An algorithm for selecting a good parameter c in radial basis function interpolation. *Advances in Computational Mathematics*, 11:193 – 210, 1999. 4.1

[34] S. A. Sarra. A numerical study of the accuracy and stability of symmetric and asymmetric rbf collocation methods for hyperbolic pdes. *Numerical Methods for Partial Differential Equations*, 24(2):670 – 686, 2008. 9

[35] S. A. Sarra. Radial basis function interpolation. pages 1 – 32, 2009. 4, 2.1, 2.1, 2.3, 5, 2.3, 2.3

[36] S. A Sarra and E. J. Kansa. *Multiquadric Radial Basis Function Approximation Methods for the Numerical Solution of Partial Differential Equations*. 1.1, 6.1

[37] I. J. Schoenberg. Metric spaces and completely monotone functions. *Annals of Mathematics*, (39):811 – 841, 1938. 1.1, 1

[38] C. J. Trahan and R. E. Wyatt. Radial bais function interpolation in the quantum trajectory method: Optimization of the multiquadratic shape parameter. *Journal of Computational Physics*, 185:27 – 49, 2003. 4.1

[39] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. SIAM, 1 edition, January 2002. 4.2

[40] Z. Qui Y. Zhou, D. Zheng and G. Dong. The application of RBF networks based on artifical immune algorithm in the performance prediction of steel bars. 6(26). 1