Radial Basis Function Interpolation: Numerical and Analytical Developments

by

Grady B. Wright
B.S., Westminster College, 1997
M.S., University of Colorado, 2000

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Applied Mathematics
2003

This thesis entitled:
Radial Basis Function Interpolation: Numerical and Analytical Developments
written by Grady B. Wright
has been approved for the Department of Applied Mathematics

Bengt Fornberg

James H. Curry

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Wright, Grady B. (Ph.D., Applied Mathematics)

Radial Basis Function Interpolation: Numerical and Analytical Developments

Thesis directed by Prof. Bengt Fornberg

The Radial Basis Function (RBF) method is one of the primary tools for interpolating multidimensional scattered data. The methods' ability to handle arbitrarily scattered data, to easily generalize to several space dimensions, and to provide spectral accuracy have made it particularly popular in several different types of applications. Some of the more recent of these applications include cartography, neural networks, medical imaging, and the numerical solution of partial differential equations (PDEs). In this thesis we study three issues with the RBF method that have received very little attention in the literature.

First, we focus on the behavior of RBF interpolants near boundaries. Like most interpolation methods, a common feature of the RBF method is how relatively inaccurate the interpolants are near boundaries. Such boundary induced errors can severely limit the utility of the RBF method for numerically solving certain PDEs. With that as motivation, we investigate the behavior of RBF interpolants near boundaries and propose the first practical techniques for ameliorating the errors there.

We next focus on some numerical developments for the RBF method based on infinitely smooth RBFs. Most infinitely smooth RBFs feature a free "shape" parameter ε such that, as the magnitude of ε decreases, the RBFs become increasingly flat. While small values of ε typically result in more accurate interpolants, the direct method of computing the interpolants suffers from severe numerical ill-conditioning as $\varepsilon \to 0$. Until recently, this ill-conditioning has severely limited the range of ε that could be considered in the RBF method. We present a novel numerical approach that largely overcomes the numerical ill-conditioning and allows for the stable computation of RBF interpolants for all values of ε, including the limiting $\varepsilon = 0$ case. This new method provides the first tool for the numerical exploration of RBF interpolants as $\varepsilon \to 0$.

The third focus of the thesis is on the behavior of RBF interpolants as $\varepsilon \to 0$. In most cases the interpolants converge to a finite degree multivariate polynomial interpolant as $\varepsilon \to 0$. However, in rare situations the interpolants may diverge. We investigate this phenomenon in great detail both numerically and analytically, and link it directly to the failure of a condition known as "polynomial unisolvency". We also find that the Gaussian RBF is inherently different from the other standard infinitely smooth RBFs in that it appears to result in an interpolant that never diverges as $\varepsilon \to 0$.

We conclude with a brief overview of two future research opportunities related to the topics of the thesis. The first involves using RBF interpolants to generate scattered-node finite difference formulas. The second involves using RBF interpolants to generate linear multistep methods for solving ordinary differential equations.

Acknowledgements

It is a great pleasure to acknowledge certain individuals and organizations that helped make this research possible.

First, I wish to thank my advisor, Professor Bengt Fornberg, for his magnificent direction, support, and understanding. His dedication, enthusiasm, insight, and passion for practicality have always inspired me. I will be forever grateful for his guidance. I also wish to thank all the members of the RBF research group at the University of Colorado. In particular, I want to express my great appreciation for the indispensable help that Dr. Elisabeth Larsson and Professor Toby Driscoll provided over the course of my research. This journey would have been much more difficult without their assistance. I would also like to thank Natasha Flyer for her wonderful sense of humor and excellent advice. Professor James Curry deserves my gratitude for his indispensable teaching and career advice. Finally, I wish to thank the rest of my committee members for their support and help: Professor Keith Julien, Dr. Paul Swarztrauber, and Professor Ellen Zweibel.

Second, I wish to acknowledge the different sources of financial support that I received over the course of my studies. Support for this research was provided by a National Science Foundation VIGRE Graduate Traineeship (NSF grant DMS-9810751). I am especially grateful to Professor James Meiss for all his help with this traineeship. Before receiving funding from VIGRE, I received support from Ionics Instruments Inc. through a work-study program. I wish to express my gratitude to the management at Ionics for making this funding possible, in particular, I wish to thank John Gress.

Lastly, I wish to express my utmost appreciation to my beautiful wife Erika. Without her unwavering support, encouragement, and patience this work would not have been possible. I am truly amazed by my good fortune of having such an exceptional spouse. I also wish to thank my parents and my brother for their continuous support.

Contents

Chapter

1 Introduction 1
- 1.1 Introduction . . . 2
- 1.2 Development of the multiquadric method . . . 3
- 1.3 The RBF method . . . 8
  - 1.3.1 Summary of theoretical results . . . 12
  - 1.3.2 Summary of computational results . . . 13
  - 1.3.3 Summary of applications . . . 14
- 1.4 Overview of thesis topics . . . 14

2 RBF Approximations Near Boundaries 16
- 2.1 Motivation . . . 16
- 2.2 Summary of paper . . . 17

3 Stable Computation of Multiquadric Interpolants 30
- 3.1 Motivation . . . 30
- 3.2 Summary of paper . . . 31
- 3.3 Additional results . . . 41
  - 3.3.1 Some observations on the strengths of the poles . . . 41
  - 3.3.2 Some observations on the distribution of poles for scattered data 44
  - 3.3.3 An observation on the IQ RBF . . . 48

4 Some Observations Regarding Interpolants in the Limit of Flat RBFs 54
- 4.1 Motivation . . . 54
- 4.2 Summary of paper . . . 54

5 Future Directions 63
- 5.1 Scattered node FD formulas generated by RBFs . . . 63
  - 5.1.1 An example using the RBF-FD method for numerically solving PDEs . . . 65
  - 5.1.2 Future ideas to consider . . . 68
- 5.2 RBF-based linear multistep methods for solving ODEs . . . 70
  - 5.2.1 Derivation of the RBF-based LMS methods . . . 71
  - 5.2.2 An example using RBF-based LMS methods for numerically solving ODEs . . . 73
  - 5.2.3 Future ideas to consider . . . 74

Bibliography 77

Appendix

A Paper 1 — Observations on the Behavior of Radial Basis Function Approximations Near Boundaries 82

B Paper 2 — Stable Computation of Multiquadric Interpolants for All Values of the Shape Parameter 102

C Paper 3 — Some Observations Regarding Interpolants in the Limit of Flat Radial Basis Functions 119

D The Contour-Padé Toolbox for MATLAB 139

Tables

Table

1.1 Common radial basis functions 9

2.1 Comparison of the accuracy of the cubic RBF approximation with the SNaK end conditions 22

3.1 Comparison of some results concerning the strength of the poles 43

4.1 Some results for polynomial unisolvent and non-unisolvent data points . 58

Figures

Figure

1.1 Interpolation of some scattered data using the absolute value. 4
1.2 Interpolation of some scattered data using the multiquadric. 5

2.1 Comparison of the 1-D cubic RBF approximation to a periodic function with different edge improvement methods. 24
2.2 Ilustration of the 2-D undetermined least squares edge improvement method 27
2.3 Comparison of the underdetermined least squares (ULS) boundary treatment method for 2-D RBF approximations 27
2.4 Comparison of the 2-D MQ RBF and triangle based cubic approximation 29

3.1 Illustration turning the Laurent series of RBF interpolant to a Taylor series and rational function 34
3.2 Flowchart for the Contour-Padé algorithm 35
3.3 Structures of different RBF interpolants in complex epsilon plane for investigating the pole strengths. 42
3.4 Histogram showing the number poles that occurred at various distances from the origin for our experiment. 46
3.5 Some statistics on the poles 47
3.6 Test for the dependence of some properies of the poles on minimum of the pairwise distances between all the points. 49
3.7 Illustration of the non-trivial poles of the IQ RBF interpolant 51
3.8 Illustration of the non-trivial poles of the GA RBF interpolant 53

4.1 Comparison of stability for polynomial unisolvent and non-unisolvent data points 59
4.2 Example of the trajectories of the poles in the MQ RBF interpolant as polynomial unisolvency fails 61

5.1 Initial condition of the heat equation model problem. 65
5.2 Structured grid for standard second order finite difference method and unstructured grid for the RBF based finite difference method. 66
5.3 Comparison of the RBF based finite differnce and standard second order finite difference methods for solving the heat equation in the unit disk. . 69
5.4 Plot of the actual solution of the example initial-value ODE 74
5.5 Comparison of the RBF-AB4 and RBF-AB4/AM4 methods for solving the example initial-value ODE. 75

Chapter 1

Introduction

This thesis documents several collaborative research projects related to radial basis functions (RBFs) that have been performed in Professor Bengt Fornberg's RBF research group at the University of Colorado, Boulder. In addition to Professor Bengt Fornberg (BF) and myself (GW), the members of this group have included NSF-VIGRE postdoctoral fellow Natash Flyer (NF), STINT postdoctoral fellow Elisabeth Larsson (EL), NSF-VIGRE postdoctoral fellow Toby Driscoll (TD), and graduate student Richard Charles (RC).

Three of the research projects that are to be documented have resulted in papers. These papers are included, in their submitted form, as appendices to this thesis. They have all been published in, or submitted to the international journal *Computers and Mathematics with Applications*. This is the primary journal for RBF research, and is currently the only journal to have put out a special issue on RBF methods for solving partial differential equations (PDEs). The current status of each of the papers is as follows:

- The paper in Appendix A has been published in the special issue on PDEs.
- The paper in Appendix B has been submitted.
- The paper in Appendix C has been accepted, and is to appear.

In the chapters that follow, the main results of these papers are summarized in a cohesive form and further results related to these papers are presented. A final chapter is included that summarizes the future directions of the research projects.

While it is difficult to identify individual contributions in collaborative projects, every attempt will be made to attribute the key contributions of each of the projects to their respective authors. In most cases, the authors will be referred to by their initials. A list containing the names and initials of each of these authors is given below.

Name	Initials
Richard Charles	RC
Toby Driscoll	TD
Bengt Fornberg	BF
Elisabeth Larsson	EL
Grady Wright	GW

In the three chapters devoted to the completed papers, a strict format is followed. It is assumed that the reader has first read the paper the chapter is summarizing before reading the chapter. At the beginning of each chapter, the motivation for the appropriate paper is presented. This is followed by a reproduction of all the section headings from the paper. Under each of these section headings is a summary of the contents of the corresponding section in the paper, including motivation for the section, who contributed to the section, and, when appropriate, additional ideas that were considered but not documented. The last chapter, devoted to future directions, is completely self-contained.

To avoid any confusion between referring to elements in the paper and to elements in the text of the thesis, a strict format is followed. When a reference to one of the elements of a paper, e.g. a section heading, figure, table, equation number, etc., is made in the thesis text, a different font will be used. For example, if the paper contains a section with the heading "1 Introduction", a figure labeled "Figure 2", a table labeled "Table 3", and an equation labeled "(4.1)", then these would be referred to in the thesis text as `1 Introduction`, `Figure 2`, `Table 3`, and `(4.1)`, respectively. For references to elements of the thesis, the standard thesis font is used.

Below is a list of the technical arcorynms and abbreviations used in this thesis.

Term	Meaning
ε	Shape parameter
$\phi(r)$	Radial function or RBF
FD	Finite difference
GA	Gaussian
IQ	Inverse quadratic
LMS	Linear multistep method
MQ	Multiquadric
ODE	Ordinary differential equation
PDE	Partial differential equation
RBF	Radial basis function
TPS	Thin plate spline

1.1 Introduction

A common problem in many scientific and engineering disciplines is the interpolation of scattered data. For one-dimensional data, many methods exist for solving this problem. Most of these (e.g. polynomial and Fourier interpolation) involve the same general idea: for a given set of n data points $\{x_j\}_{j=1}^n$ and corresponding data values $\{f_j\}_{j=1}^n$, a set of basis functions $\{\psi_j(x)\}_{j=1}^n$ is chosen such that a linear combination of these functions satisfies the interpolation conditions. To be more specific, a function $s(x)$ is sought of the form

$$s(x) = \sum_{j=1}^{n} \lambda_j \psi_j(x) \ , \tag{1.1}$$

such that $s(x_j) = f_j$ for $j = 1, \ldots, n$. The interpolation conditions lead to a linear system of equations which determines the expansion coefficients λ_j. For many choices of basis functions $\psi_j(x)$, this linear system is guaranteed to be non-singular whenever the data points $\{x_j\}_{j=1}^n$ are distinct.

For data in more than one dimension (i.e. x and x_j are not scalars but vectors $\underline{x}$, $\underline{x}_j \in R^{d>1}$), the approach described above no longer works. It can be shown that for any set of basis functions $\{\psi_j(\underline{x})\}_{j=1}^n$ $(n > 1)$ (independent of the data points), there exists sets of distinct data points $\{\underline{x}_j\}_{j=1}^n$ such that the linear system of equations for determining the expansion coefficients becomes singular (i.e. there does not exist an interpolant of the form in (1.1)). This result is often referred to as Haar's theorem [35].

This non-singularity problem can be bypassed by following a somewhat different approach for creating an interpolating function. Instead of taking linear combinations of a set of basis functions that are independent of the data points, one takes a linear combination of translates of a single basis function that is (*radially*) symmetric about its center. This approach, pioneered by R.L. Hardy [36], is referred to as the radial basis function (RBF) method.

The remaining sections of this chapter are organized as follows: In Section 2, the RBF method is introduced via Hardy's development of the multiquadric method. In Section 3, some important mathematical properties of the RBF method are summarized. The final Section 4 contains an overview of the topics of the remaining chapters.

1.2 Development of the multiquadric method

The RBF method is a generalized version of the multiquadric (MQ) method developed in 1968 by Hardy [36]. Hardy developed the MQ method to solve a problem from cartography. Namely, given a set of sparse, scattered measurements from some source points on a topographic surface (e.g. elevation measurements from Rocky Mountain National Park), construct a "satisfactory" continuous function that represents the surface. Here satisfactory means a function that provides an exact fit of the data and provides a good approximation of the features of the surface (i.e. location of hilltops, saddles, breaks in slope, and drainage junctions). Part of the motivation for constructing such a function was to create an automated and unbiased method for generating contour maps of a topographic surface. At the time, this was not the standard practice. The standard practice was for a skilled topographer to use her perception of the collected data to decide how the surface should look [37]. Granted there were some analytical methods for determining the progression of the surface from a data point to another data point, but the overall map was influenced by the topographer. This practice could easily result in two topographers producing quite different maps from the same data set.

In addition to creating an automatic and objective way of producing contour maps, there was a mathematical motivation for constructing a continous function for a topographic surface. By having such a function, analytic geometry and calculus could be used for determining such things as unobstructed lines of sight, volumes of earth, hilltops, valleys, and distance along a curve [36], [37].

The first attempts at constructing these topographic surface functions involved the use of Fourier and polynomial interpolation methods. However, both of these methods were found to be unsatisfactory. Interpolation with Fourier series was not acceptable because the sparse data sets resulted in a function that tended to oscillate too much between the data points [36]. Polynomial interpolation was inadequate because the sparse data sets resulted in a function that was unable to represent the rapid variations of real topographic surface [36]. In addition to these issues, we should recall from the previous section that the problem of determining an interpolating Fourier series or polynomial in

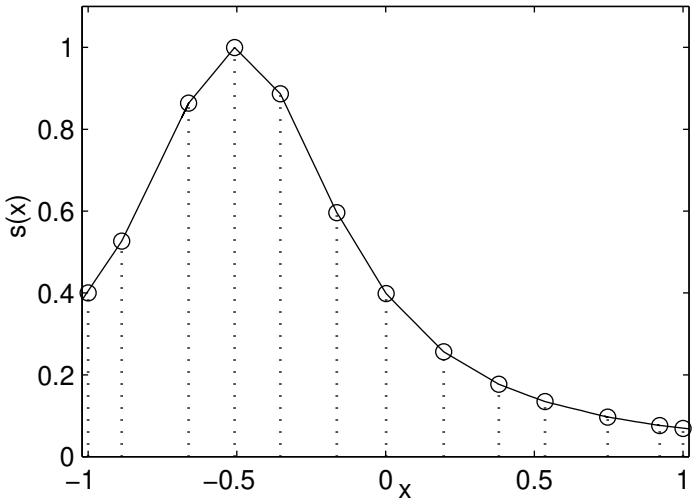


Figure 1.1: Interpolation of some scattered data using Equation 1.2. The source points are given by the circles.

two-dimensions can be singular (Haar's theorem; this result, however, does not appear to have been a major concern to the investigators).

Least-squares methods based on Fourier and polynomial series were also investigated. However, these methods were also found to be unsatisfactory because they did not provide an exact fit of the measured data [36].

The failure of the Fourier and polynomial series methods for constructing a satisfactory function that represented the topographic surface led Hardy on a search for a new type of method. By a largely trial-and-error approach, he was eventually successful.

Hardy's first step was to study a one dimensional version of the problem, namely construct a satisfactory function that represents a topographic profile (a curve) from scattered measurements of the profile. While studying this problem, he found that the profile could be represented fairly satisfactorily by a piecewise linear interpolating function [37]. For a set of n distinct (scattered) source points $\{x_j\}_{j=1}^n$ and corresponding measurements $\{f_j\}_{j=1}^n$, he proposed the following form for the interpolating function:

$$s(x) = \sum_{j=1}^{n} \lambda_j |x - x_j| \ , \tag{1.2}$$

where λ_j are determined by collocation, i.e. $s(x_j) = f_j$, $j = 1, 2, \ldots, n$. Geometrically, this corresponds to interpolating the data by a linear combination of n translates of the absolute value basis function $|x|$, where the vertex of each basis function is centered at one of the source points. Figure 1.1 shows a fit of $n = 13$ data points by this technique.

The problem with representing a topographic profile with (1.2), Hardy recognized, is that the function has a jump in the first derivative at each data point. While this may accurately model the behavior at a cliff, it does not allow one to use the techniques of calculus with the function to find such things as hilltops (maximums) and valleys (minimums). Hardy realized that this problem could be easily resolved by replacing

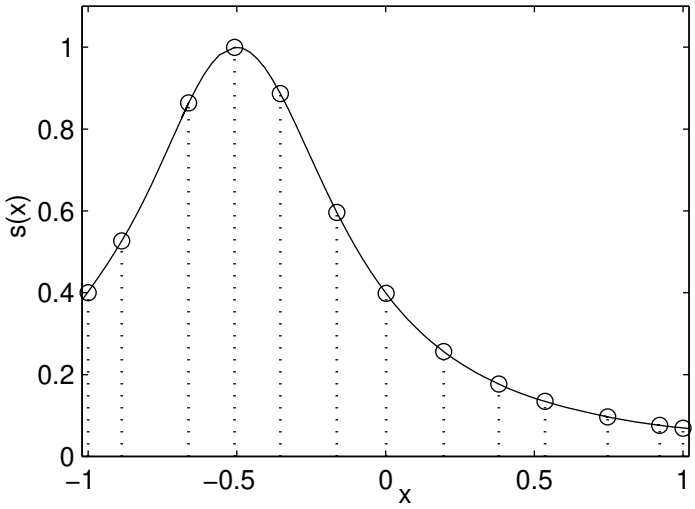


Figure 1.2: Interpolation of some scattered data using Equation 1.3 with $c = 2$. The data points are given by the circles.

the absolute value basis function in (1.2) by one that is continuously differentiable. He proposed using the basis function $\sqrt{c^2 + x^2}$ (i.e. the upper curve of a hyperbola), where c is some non-zero arbitrary constant, because of its similarity to the absolute value function [37]. The new interpolating function thus becomes

$$s(x) = \sum_{j=1}^{n} \lambda_j \sqrt{c^2 + (x - x_j)^2}\,, \tag{1.3}$$

where the λ_j are again determined by collocation. For $c \neq 0$ (1.3) clearly leads to a continuously differentiable interpolant, and for $c = 0$ (1.3) is equivalent to (1.2). The same geometric interpretation holds for this new method. However, instead of the absolute value basis function, this method corresponds to taking a linear combination of translates of the hyperbola basis function. Figure 1.2 shows a fit of the same data as Figure 1.1 using (1.3) with $c = 2$.

Hardy found that this new method not only provided an accurate representation of a topographic profile, but that the techniques of calculus could be easily applied to it. For example, the interpolant to the data in Figure 1.2 could now be used for finding the maximum and inflection points of the topographic profile.

The key property with Hardy's new approach was that it carries over to more than one-dimension. The absolute value of the difference between two one-dimensional points is simply the Euclidean distance between the points (i.e $|x - x_j| = \sqrt{(x - x_j)^2}$). The natural extension of (1.2) to two dimensions is to create an interpolating function based on translates of the Euclidean distance function in two dimensions. Specifically, given n distinct (scattered) source points $\{(x_j, y_j)\}_{j=1}^{n}$ and corresponding topographic

measurements $\{f_j\}_{j=1}^n$, Hardy proposed interpolating the data with the function

$$s(x,y) = \sum_{j=1}^{n} \lambda_j \sqrt{(x-x_j)^2 + (y-y_j)^2}\,, \tag{1.4}$$

where the λ_j are again determined by collocation, i.e. $s(x_j, y_j) = f_j$ for $j = 1, \ldots, n$. Geometrically, this method corresponds to interpolating the data by a linear combination of n translates of a cone (i.e. a radially symmetric function $\phi(r) = r$ where $r = \sqrt{x^2 + y.^2}$). The vertex of each cone is centered at one of the source points.

Like its one-dimensional equivalent, this new two-dimensional method suffers from the problem that it results in a piecewise continous function. Also like its one-dimensional equivalent, Hardy was able to find a simple solution to alleviate the problem. Instead of using a linear combination of the Euclidean distance basis function to interpolate the data, he again proposed using a linear combination circular hyperboloid basis functions (i.e. rotated hyperbola basis functions $\sqrt{c^2 + x^2}$), translated to be centered at each source point.

The exact form of this new type of interpolant is given by

$$s(x,y) = \sum_{j=1}^{n} \lambda_j \sqrt{c^2 + (x-x_j)^2 + (y-y_j)^2}\,. \tag{1.5}$$

For $c \neq 0$, this function is infinitely differentiable, thus techniques in multivariable calculus can be used for determining properties of the topographic surface the function is approximating.

Hardy found that (1.5) was an excellent method for approximating a topographic surface from sparse, scattered measurements [36]. The new method did not suffer from large oscillations like the Fourier series methods and it was able to account for rapid variations of the topographic surface unlike the polynomial series methods. Hardy named this new technique the "multiquadric method" because he considered the principal feature of the method to be a "superpositioning of quadric surfaces" [37].

While Hardy orginally developed the MQ method (1.5) for solving a two-dimensional interpolation problem, he realized that it could be easily generalized for interpolating data in any dimension. Looking at (1.5), it is clear that the only geometric feature that the basis function depends on is the Euclidean distance of the point (x, y) from its center (x_j, y_j). A three-dimensional interpolation method is then easily created by having the basis function only depend on the distance of the point (x, y, z) from its center (x_j, y_j, z_j). Continuing this idea to higher dimensions leads us to the following definition of the general MQ method for interpolating multidimensional scattered data:

Definition 1 (MQ Method) Given a set of n distinct data points (or source points) $\{\underline{x}_j\}_{j=1}^n$ in R^d (i.e. $\underline{x}_j = (x_1^{(j)}, x_2^{(j)}, \ldots, x_d^{(j)})$), and corresponding (scalar) data values $\{f_j\}_{j=1}^n$, the MQ interpolant to the data is given by

$$s(\underline{x}) = \sum_{j=1}^{n} \lambda_j \sqrt{c^2 + \left\|\underline{x} - \underline{x}_j\right\|^2}\,, \tag{1.6}$$

where $\|\cdot\|$ is the Euclidean norm. The expansion coefficients λ_j are determined from the interpolation conditions $s(\underline{x}_j) = f_j$, $j = 1, \ldots, n$, which leads to the following symmetric linear system:

$$\begin{bmatrix} & A & \end{bmatrix} \begin{bmatrix} \lambda \end{bmatrix} = \begin{bmatrix} f \end{bmatrix} , \tag{1.7}$$

where the entries of A are given by $a_{j,k} = \sqrt{c^2 + \left\|\underline{x}_j - \underline{x}_k\right\|^2}$.

Following the publication of Hardy's MQ method in [36], many researchers began using the method in other scientific disciplines. For example, the method was successfully applied to problems in hydrology (approximating the average rainfall in a region), geodesy (approximating the Earth's gravitaional field), photogrammetry (reconstructing images), geophysics (approximating the Earth's crustal movement), and geology (approximating ground water levels from well-logs); see [37] for more details.

Besides utilizing the method for new applications, many practical properties of the method were also considered. For example, a Hermite-type interpolation approach (i.e. matching data and derivative values) was proposed, the effect of the free parameter c was studied, and a physical interpretation of the method was given linking it to biharmonic potential theory [37]. In addition, new methods similar to the MQ method, but developed independently in most cases, started appearing. These new methods had the same basic form as (1.6), but instead of using translates of the basis function $\sqrt{c^2 + \|\underline{x}\|^2}$, they used translates of different, still Euclidean distant-dependent, basis functions. For example Duchon [16] used the basis function $\|\underline{x}\|^2 \log \|\underline{x}\|$ and $\|\underline{x}\|^3$, Schagen [59] used the Gaussian basis function $e^{-(\varepsilon\|\underline{x}\|)^2}$ (where ε is some free parameter), and Hardy and Göpfert [38] used the basis function basis function $(c^2 + \|\underline{x}\|^2)^{-1/2}$.

One of the most important studies of the MQ method was done in 1979 by Franke [30]. This study was primarily concerned with investigating a vast number of the available methods for interpolating two-dimensional scattered data in order to determine which methods deserved further mathematical study. It consisted of testing about 30 different interpolation methods to see how well they approximated 6 different test functions that were sampled on 3 different density levels. Franke summarized the results in [31], and found that the MQ method (1.6) was the most impressive of them all, where it provided the best approximation in 13 of the 18 tests, and provided the second best approximation on 3 of the remaining 5 test.

Although Franke provided empirical evidence that the MQ method deserved more attention, he also expressed some reservations about the method. While the similar methods of Duchon [16] and Schagen [59] had a mathematical foundation, no such foundation existed for the MQ method. For example, it was not known if the method was uniquely solvable, i.e. if the linear system (1.7) was nonsingular. Extensive numerical experiments, led Franke to conjecture that the MQ method was always nonsingular (and that the $n \times n$ A matrix in (1.7) satisfied the relationship $(-1)^n \det(A) > 0$) [32], but no proof was provided.

A mathematical foundation of the MQ method was ultimately provided in 1986 by Micchelli [50]. Besides providing a proof of Franke's conjecture, Micchelli provided sufficient conditions to guarantee the nonsingularity of the method when a number of other basis functions are used. For example Micchelli's results guaranteed nonsingularity for the basis function proposed by Hardy and Göpfert [38]. In addition, the basis

function proposed by Duchon [16], which was already shown to lead to a nonsingular method by variational principles, fit nicely into Micchelli's theory.

Nearly 15 years after Hardy introduced the idea of fitting multidimensional scattered data with a linear combination of translates of the basis function $\sqrt{c^2 + \|\underline{x}\|^2}$, Micchelli not only showed that method was unconditionally nonsingular, but that nonsingularity could also be guaranteed when a number of other basis functions are used. Consequently, the MQ method was recognized as only one specific example of a more general method. The guiding principle behind this general method is to use translates of a single basis function $\phi(r)$ that depends only on the Euclidean distance from its center in order to create a multidimensional interpolant. A function with this single dependence is of course *radially* symmetric about its center, thus these $\phi(r)$ became known as "radial functions" or "radial basis functions" (RBFs), and the general method was named the RBF method [55].

1.3 The RBF method

The RBF method is now one of the primary tools for interpolating multidimensional scattered data. Its simple form, and ability to accurately approximate an underlying function have made the method particularly popular. In this section we review the conditions on the basis functions to guarantee a nonsingular method. This is followed by summary of some important theoretical and computational results, as well as a summary of some of the applications the RBF method has been successfully applied to.

In this thesis, we define two separate versions of the RBF method. One we refer to as the "basic" RBF method and the other we refer to as the "augmented" RBF method. Unless otherwise noted, throughout this thesis we let $\underline{x}, \underline{x}_j \in R^d$ where d is some positive integer, f_j be a scalar value, and $\|\cdot\|$ denote the Euclidean norm.

The basic RBF method is defined as follows:

Definition 2 (Basic RBF Method) Given a set of n distinct data points $\{\underline{x}_j\}_{j=1}^n$ and corresponding data values $\{f_j\}_{j=1}^n$, the basic RBF interpolant is given by

$$s(\underline{x}) = \sum_{j=1}^{n} \lambda_j \phi(\|\underline{x} - \underline{x}_j\|) \ , \tag{1.8}$$

where $\phi(r)$, $r \geq 0$, is some radial function. The expansion coefficients λ_j are determined from the interpolation conditions $s(\underline{x}_j) = f_j$, $j = 1, \ldots, n$, which leads to the following symmetric linear system:

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} \\ \lambda \\ \\ \end{bmatrix} = \begin{bmatrix} \\ f \\ \\ \end{bmatrix} \ , \tag{1.9}$$

where the entries of A are given by $a_{j,k} = \phi(\|\underline{x}_j - \underline{x}_k\|)$.

As mentioned in the previous section, Micchelli [50] gave sufficient conditions for $\phi(r)$ in (1.8) to guarantee that the A matrix in (1.9) is unconditionally nonsingular, and thus that the basic RBF method is uniquely solvable. Some common examples of the

Type of basis function	$\phi(r)\ (r \geq 0)$
Infinitely smooth RBFs	
Gaussian (GA)	$e^{-(\varepsilon r)^2}$
Inverse quadratic (IQ)	$\frac{1}{1+(\varepsilon r)^2}$
Inverse multiquadric (IMQ)	$\frac{1}{\sqrt{1+(\varepsilon r)^2}}$
Multiquadric (MQ)	$\sqrt{1+(\varepsilon r)^2}$
Piecewise smooth RBFs	
Linear	r
Cubic	r^3
Thin Plate Spline (TPS)	$r^2 \log r$

Table 1.1: Some commonly used radial basis functions

$\phi(r)$ that lead to a uniquely solvable method are given in the first five entries of Table 1.1. The parameter ε in the infinitely smooth RBFs from the table is a free parameter for controlling the shape of functions. At this point, assume that it is some fixed non-zero real value. In addition, note that MQ RBF has been redefined from Hardy's original definition by the transformation $c = \frac{1}{\varepsilon}$ (and the removal of the appropriate scaling factor).

The first sufficient conditions for $\phi(r)$ to guarantee nonsingularity of the A matrix were actually given in 1938 by Schoenberg [60]. Micchelli only later showed that these conditions could be relaxed so that a larger class of functions could be considered. To understand both Schoenberg and Michelli's results it is necessary to define a *completely monotone function.*

Definition 3 (Completely Monotone Function) A function ψ is said to be completely monotone on $[0, \infty)$ if

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

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

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

Theorem 4 (Schoenberg [60]) If $\psi(r) = \phi(\sqrt{r})$ is completely monotone but not constant on $[0, \infty)$, then for any set of n distinct points $\{\underline{x}_j\}_{j=1}^n$, the $n \times n$ matrix A with entries $a_{j,k} = \phi\left(\left\|\underline{x}_j - \underline{x}_k\right\|\right)$ is positive definite (and therefore non-singular).

From this theorem we can conclude, for example, that the basic RBF method (1.8) is uniquely solvable for the first three RBFs in Table 1.1 since for $\ell = 0, 1, \ldots$, and $r > 0$,

$$\begin{aligned}
\psi(r) = \phi(\sqrt{r}) = e^{-\varepsilon^2 r} \quad &\Rightarrow \quad (-1)^\ell \psi^{(\ell)}(r) = \varepsilon^{2\ell} e^{-\varepsilon^2 r} > 0\,, \\
\psi(r) = \phi(\sqrt{r}) = \frac{1}{1+\varepsilon^2 r} \quad &\Rightarrow \quad (-1)^\ell \psi^{(\ell)}(r) = \frac{\ell! \varepsilon^{2k}}{(1+\varepsilon^2 r)^{\ell+1}} > 0\,, \\
\psi(r) = \phi(\sqrt{r}) = \frac{1}{\sqrt{1+\varepsilon^2 r}} \quad &\Rightarrow \quad (-1)^\ell \psi^{(\ell)}(r) = \frac{\Gamma\left(\ell+\frac{1}{2}\right)\varepsilon^{2\ell}}{\sqrt{\pi}\,(1+\varepsilon^2 r)^{\ell+\frac{1}{2}}} > 0\,.
\end{aligned}$$

This same conclusion cannot be made for the MQ and linear RBFs since in both cases $\psi(r) > 0$ and $\psi'(r) > 0$ for $r \geq 0$. However, by Micchelli's elegant extension of Theorem 4 this conclusion does follow.

Theorem 5 (Micchelli I [50]) Let $\psi(r) = \phi(\sqrt{r}) \in C^0[0,\infty)$, $\psi(r) > 0$ for $r > 0$, and $\psi'(r)$ completely monotone but not constant on $(0,\infty)$. Then for any set of n distinct points $\{\underline{x}_j\}_{j=1}^n$, the $n \times n$ matrix A with entries $a_{j,k} = \phi\left(\left\|\underline{x}_j - \underline{x}_k\right\|\right)$ is non-singular. Furthermore, for $n \geq 2$, the matrix has $n-1$ negative eigenvalues and one positive eigenvalue.

Clearly, the MQ and linear RBFs satsify the continuity and positivity requirements of the theorem. In addition, for both functions the first derivative of their corresponding $\psi(r)$ function is completely monotone and not constant on $(0,\infty)$ since

$$\psi(r) = \phi(\sqrt{r}) = \sqrt{1+\varepsilon^2 r} \quad \Rightarrow \quad (-1)^{\ell-1}\psi^{(\ell)}(r) = \frac{\Gamma\left(\ell - \frac{1}{2}\right)\varepsilon^{2\ell}}{2\sqrt{\pi}\left(1+\varepsilon^2 r\right)^{\ell-\frac{1}{2}}} > 0\,,$$

$$\psi(r) = \phi(\sqrt{r}) = r^{\frac{1}{2}} \quad \Rightarrow \quad (-1)^{\ell-1}\psi^{(\ell)}(r) = \frac{\Gamma\left(\ell - \frac{1}{2}\right)}{2\sqrt{\pi}\, r^{\ell-\frac{1}{2}}} > 0\,,$$

for $\ell = 1, 2, \ldots$, and $r > 0$. Thus, the basic RBF method (1.8) is uniquely solvable for the MQ and linear radial functions. Moreover, we see that result regarding the eigenvalues of the A matrix provides a proof of Franke's conjecture about the determinant of this matrix that was mentioned in the previous section.

Unfortunately, Theorem 5 does not apply to $\phi(r)$ such as the prominent TPS RBF listed in Table 1.1. This RBF nearly satisifies all the conditions of the theorem exepct that its corresponding $\psi(r)$ function is only completely monotone when differentiated twice instead of once, i.e.

$$\psi(r) = \phi(\sqrt{r}) = r\log r^{1/2} \quad \Rightarrow \quad (-1)^{\ell-2}\psi^{(\ell)}(r) = \frac{(\ell-2)!}{2\, r^{\ell-1}} > 0\,,$$

for $\ell = 2, 3, \ldots$, and $r > 0$. However, the A matrix corresponding to this RBF can easily be made singular. One of the simplest ways this is accomplished is to choose a set of $n > 1$ data points $\{\underline{x}_j\}_{j=1}^n$ in R^{n-1} such that each of the data points is at the vertex of a unit simplex. In this case, every entry of the A matrix is zero, and thus A is obviously singular!

Fortunately, Micchelli was able to extend Theorem 5 to include these types of RBFs by adding some extra restrictions. These restrictions require some additions to the basic RBF method given in Definition 2 and lead to the method that we refer to as the augmented RBF method.

Before we give the definition for the augmented RBF method, we need to make one more definition.

Definition 6 ($\Pi_m(R^d)$) Let $\Pi_m(R^d)$ be the space of all d-variate polynomials that have degree less than or equal to m. Furthermore, let M denote the dimension of $\Pi_m(R^d)$, then $M = \binom{m+d}{m}$.

Definition 7 (Augmented RBF Method) Given a set of n distinct data points $\{\underline{x}_j\}_{j=1}^n$ and corresponding data values $\{f_j\}_{j=1}^n$, the augmented RBF interpolant is given by

$$s(\underline{x}) = \sum_{j=1}^{n} \lambda_j \, \phi(\|\underline{x} - \underline{x}_j\|) \; + \sum_{k=1}^{M} \gamma_k p_k(\underline{x}) \qquad \underline{x} \in R^d \tag{1.10}$$

where $\{p_k(\underline{x})\}_{k=1}^M$ is a basis for $\Pi_m(R^d)$ and $\phi(r)$, $r \geq 0$, is some radial function. To account for the conditions from the additional polynomial terms, the following constraints are imposed:

$$\sum_{j=1}^{n} \lambda_j p_k(\underline{x}_j) = 0, \quad k = 1, 2, ..., M. \tag{1.11}$$

The expansion coefficients λ_j and γ_k are then determined from the interpolation conditions and the constraints (1.11), which leads to the following symmetric linear system:

$$\left[\begin{array}{c|c} A & P \\ \hline P^T & 0 \end{array}\right] \left[\begin{array}{c} \lambda \\ \hline \gamma \end{array}\right] = \left[\begin{array}{c} f \\ \hline 0 \end{array}\right] . \tag{1.12}$$

A is the matrix in (1.9) and P is the $n \times M$ matrix with entries $p_k(\underline{x}_j)$ for $j = 1, \ldots, n$ and $k = 1, \ldots, M$.

The value of m to use in the above definition becomes apparent from Micchelli's extension of Theorem 5 [50].

Theorem 8 (Micchelli II [50]) Let $\psi(r) = \phi(\sqrt{r}) \in C^0[0, \infty)$ and $\psi^{m+1}(r)$ be completely monotone but not constant on $(0, \infty)$ for $m \geq 0$. Then for any set of n distinct points $\{\underline{x}_j\}_{j=1}^n$ that satsify the condition $\text{rank}(P) = M$, where P is the $n \times M$ matrix in (1.12), the full $(n + M) \times (n + M)$ matrix in (1.12) is non-singular. Furthermore, if $\bar{m}$ is the smallest m such that $\psi^{m+1}(r)$ is completely monotone, then for any non-zero vector $\alpha \in R^n$ that satisfies the condition $P^T\alpha = 0$, the following relation holds: $(-1)^{\bar{m}+1}\alpha^T A\alpha > 0$, where A is the $n \times n$ sub-matrix in (1.12).

This theorem clearly provides sufficient conditions to guarantee the unique solvability of the augmented RBF method. We can, for example, use this theorem to conclude that the augmented RBF method is uniquely solvable for the cubic and TPS RBFs when $m = 1$ and the conditions on the data points $\{\underline{x}_j\}_{j=1}^n$ are satisfied. With $m = 1$ the RBF interpolant is augmented with a constant and linear polynomial, e.g. in two dimensions, the RBF interpolant is given by:

$$s(x, y) = \sum_{j=1}^{n} \lambda_j \, \phi\left(\sqrt{(x - x_j)^2 + (y - y_j)^2}\right) \; + \gamma_0 + \gamma_1 x + \gamma_2 y.$$

Revisiting the example presented above that created singularity in the basic RBF method with the TPS RBF (arranging $n > 1$ data points at the vertices of a unit simplex in R^{n-1}) we find that singularity is avoided for the augmented RBF method. This is because in this case the RBF portion of (1.10) will be identically equal to zero and the augmented constant and linear polynomial will interpolate the data.

In addition to the above remarks about Theorem 8, we also note that

- While the theorem shows that the augmented RBF method is much less restrictive than the basic RBF method on the functions $\phi(r)$ that can be used, it also shows that the augmented method is far more restrictive on the data points $\{\underline{x}_j\}_{j=1}^n$ that can be used. The only restriction on the data points for the basic method is that the points are distinct, while the restriction for the augmented method is that the data points lead to a P matrix that satisfies $\text{rank}(P) = M$. This is equivalent to saying that for a given basis $\{p_k(\underline{x})\}_{k=1}^M$ for $\Pi_m(R^d)$ the data points $\{\underline{x}_j\}_{j=1}^n$ must satisfy the condition

$$\sum_{k=1}^{M} \gamma_k p_k(\underline{x}_j) = 0 \quad \Rightarrow \quad \gamma \equiv 0 \,,$$

  for $j = 1, \ldots, n$.

- By the definition of a completely monotone function (Definition 3), we know that if $\psi^{m+1}(r)$ is completely monotone, then so is $\psi^{m+\kappa}(r)$, for $\kappa = 2, 3 \ldots$. Thus, we can conclude from the theorem that the augmented RBF method is uniquely solvable for all values of $m \geq \bar{m}$, provided that the conditions on the data points $\{\underline{x}_j\}_{j=1}^n$ are satisfied.

- Increasing the value of m in the augmented RBF method past the minimum value $\bar{m}$ (i.e. adding higher degree polynomial terms than the minimum degree that is required to guarantee unique solvability), may seem counterproductive since the conditions on the data points become increasingly more restrictive. However, the technique has some practical purposes. For example, it can be used to ensure the RBF interpolant (1.10) reproduces polynomials of some degree and it can be used to improve the accuracy of the RBF interpolants (this is explained in more detail in Chapter 2).

To summarize, Theorem 4 and 5 provide sufficient conditions to guarantee the basic RBF method is uniquely solvable and Theorem 8 provides sufficient conditions for the augmented RBF method. From this point on, both methods are genearally referred to as the RBF method, with the specific form implied by the type of RBF that it is to be used.

1.3.1 Summary of theoretical results

The types of $\phi(r)$ available for use in the RBF method can be split up into two main categories, infinitely smooth and piecewise smooth (as indicated in Table 1.1). As mentioned above, the infinitely smooth functions feature a shape parameter ε, such that as $\varepsilon \to 0$ the basis functions become increasingly flat. All three of the above theorems are valid for all non-zero values of ε.

When considering the accuracy of the RBF interpolants and the stability of the corresponding linear system, two very different situations arise. These two situations are determined by whether the $\phi(r)$ used in the method is piecewise smooth or infinitely smooth.

For the piecewise smooth case, as the number of data points increases in a fixed domain, the RBF interpolants converge algebraically to the underlying (sufficiently

smooth) function being interpolated. The rate of convergence is directly related to the smoothness of $\phi(r)$ [57, 68], and the rate often increases as the number of space dimensions increases [8, 53]. The stability of the linear system (1.9) (or (1.12)) for the piecewise smooth case is also related to the amount of smoothness of $\phi(r)$. As the number of data points increases in a fixed domain, the condition number of the matrix in the linear system grows algebraically with a rate related to the irregularity of $\phi(r)$ at the origin [51, 57].

Unlike the piecewise smooth case, the accuracy and the stability for the infinitely smooth $\phi(r)$ depend on the number of data points *and* the value of the shape parameter ε. For a fixed ε, as the number of data points increases, the RBF interpolant converges to the underlying (sufficiently smooth) function being interpolated at a spectral rate, i.e $O(e^{-\frac{const.}{h}})$ where h is a measure of the "typical" distance between data points [48, 57, 69]. In certain special cases, such as the Gaussian (GA) RBF (see Table 1.1), the RBF interpolant has been shown to exhibit "super-spectral" convergence, i.e. $O(e^{-\frac{const.}{h^2}})$ [27, 48, 57]. In either case, the value of *const.* in the estimates is effected by the value of ε. For a fixed number of data points, Madych [47] has shown that the accuracy of the RBF interpolant can often be significantly improved by decreasing the value of ε. However, decreasing ε or increasing the number of data points has a severe effect on the stability of the linear system (1.9) (or (1.12)). For a fixed ε, the condition number of the matrix in the linear system grows exponentially as the number of data points is increased [51, 57]. For a fixed number of data points, similar growth occurs as $\varepsilon \to 0$ [51, 57].

1.3.2 Summary of computational results

A very important feature of the RBF method is that its complexity does not increase as dimension of the interpolation increases (apart, of course, from the trivial change of computing distances in higher dimensions). Their simple form makes implementing the methods extremely easy (compared to, for example, a bicubic spline method). However, three main computational challenges exist

(a) The matrix for determining the interpolation coefficients (1.9) or (1.12) is dense, which makes the computational cost of the methods high.

(b) The matrix is ill-conditioned when the number of data points is large.

(c) For the infinitely smooth RBFs and a fixed number of data points, the matrix is also ill-conditioned when ε is small.

Several studies have been devoted to resolving (a). Most of them concentrate on the development of iterative techniques (see for example [5, 7, 17, 20]). The most prominent of these techniques is that of Beatson *et. al.* [5], which suggests the computational cost can be reduced to $O(n \log n)$, where n is the number of data points. Another procedure for resolving (a) is to create a sparse interpolation coefficient matrix, which can be accomplished by using the compactly supported $\phi(r)$ of Wendland [64] or Wu [67].

Most of the studies devoted to resolving (a), also provide some preconditioning techniques for resolving (b). Another technique that has been successful for resolving (b) is domain decomposition (see for example [6, 44]).

Thus far, only one technique has been developed for resolving (c). This technique was developed by BF and GW, and is explained in the paper summarized in Chapter 3.

1.3.3 Summary of applications

The RBF method has continued to be used in the various application areas mentioned in Section 1.1. New application areas include pattern recognition [22, 62], medical imaging [12], image warping [70], and especially neural networks [33].

Due to the simple form of the RBF method, its ability to accurately interpolate scattered data, and its straightforward generalization to higher dimensions, a large effort has also been directed towards using RBFs for numerically solving partial differential equations (PDEs) (especially in complex domains). This effort is due to the pioneering work of E. Kansa, who, in the early 1990's, showed that RBFs could be applied very effectively for the numerical solution of some PDEs arising in fluid mechanics [42, 43]. Since Kansa's initial investigation, numerous RBF based methods have been developed for numerically solving PDEs. Several of these methods involve approximating the spatial derivates by derivatives of a collocated RBF interpolant (see for example [18, 19, 40, 41, 42, 43, 46, 66]). Another popular technique involves using RBF interpolants in the boundary element method (BEM), where particular solutions of an inhomogeneous PDE are approximated by RBF interpolants and then combined with the homogeneous fundamental solutions [34].

1.4 Overview of thesis topics

The main focus of this thesis is on some numerical and analytical aspects of the RBF method. However, some applications will also be presented.

In Chapter 2, we focus on an aspect of the RBF interpolants that has received very little attention in the literature, namely, the behavior of the interpolants near boundaries. While the theoretical results summarized in Section 1.3.1 certainly indicate the RBF interpolants possess some attractive accuracy properties, all these results require that the underlying function satisfies some "unrealistic" boundary properties, or the results are only valid when the RBF interpolants are evaluated well within the interior of the interpolation domain. In real applications, the accuracy of the RBF interpolants (like for most other interpolants) can deteriorate quite dramatically at or near the boundaries of the domain. This issue has been noted in studies dating all the way back to Hardy's fundamental paper introducing the MQ method [36]. However, none of these studies have been devoted to investigating techniques for resolving this problem. The purpose of our paper "Observations on the Behavior of Radial Basis Function Approximations Near Boundaries" (see Appedix A) is to provide such a study. This paper is the topic of Chpater 2.

As mentioned in Section 1.3.1, the shape parameter ε dramatically affects the accuarcy and stability of the RBF method based on the infinitely smooth RBFs. If the data points and the function to be interpolated are fixed, there is typically one value of ε that produces the most accurate interpolant (we refer to this as the "optimal" value), and this value is often very small. Several methods have been developed for finding this optimal value, however, due to stability issues, they are severely limited in the range of ε that can be considered. In our paper "Stable Computation of Multiquadric Interpolants for All Values of the Shape Parameter" (see Appedix B) we present the first numerical method that overcomes the stability issues and allows the RBF interpolants to be computed for the full range of ε, including $\varepsilon = 0$. This paper is the topic of

Chapter 3.

The last statement in the previous paragraph is not a misprint. It turns out that regardless of the fact that the linear system for solving for the RBF expansion coefficients approaches singularity very rapidly as $\varepsilon \to 0$, the RBF interpolant remains well-behaved and in fact (usually) converges to a multivariate polynomial. The focus of our paper "Some Observations Regarding Interpolants in the Limit of Flat Radial Basis Functions" (see Appendix C) is on the behavior of the RBF interpolants as $\varepsilon \to 0$. This paper is the topic of Chapter 4.

In Chapter 5, we focus on two future directions of the above research. The first is motivated by the recent development of a new method for numerically solving PDEs [63]. The idea behind this new method is to use RBF interpolants instead of polynomial interpolants for generating local finite difference formulas. By using RBF interpolants, the nodes for the finite difference formulas are allowed to be placed in general locations (in any space dimension) without worrying about singularity in the interpolant. This allows the nodes to be placed in any configuration, so as to provide the best discretization of the underlying domain. With the novel algorithm presented in Chapter 4 for computing RBF interpolants for all values of the shape parameter, we present some preliminary results on how these FD formulas are effected by the value of ε.

The second direction involves the application of the RBF method for the numerical solution of ODEs. We consider generating linear multistep methods from RBF interpolants instead of polynomial interpolants (the standard method). The motiviation behind this idea is two-fold. First, high-order linear multistep methods are based on high-order polynomial interpolation, which is known to lead to the Runge-Phenomenon. By the replacing the polynomial interpolants with RBF interpolants, we may be able to take advantage of some of the techniques developed in Chapter 2 that resolve the bounary issues. The second motivation is that RBF interpolants based on the infinitely smooth basis functions feature a free parameter ε that may be able to be adjusted to provide an optimal solution to the ODE. In addition, in the $\varepsilon = 0$ limit RBF based linear multistep methods will reproduce a standard polynomial based methods. The novel idea of using RBFs to generate linear multistep methods does not appear to have received any previous consideration in the RBF literature.

Chapter 2

RBF Approximations Near Boundaries

2.1 Motivation

The initial goal of our RBF research group was to use the RBF method for numerically solving PDEs in irregularly shaped domains. The early work of Kansa [42, 43] demonstrated that this technique was very successful for solving partially and fully dissipative PDEs. We were, however, interested in solving nondissipative wave-type PDEs, such as Maxwell's equation. In this application, we envisioned using a method-of-lines formulation, where an RBF interpolant would be used for approximating the spatial derivatives of the PDE and a standard ODE solver would be used for advancing the solution in time. Our straightforward implementation of this method proved unsuccessful not only for solving Maxwell's equation in two dimensions, but also for the simple one-dimensional advection equation $u_t + u_x = 0$ with the boundary condition $u(0,t) = 0$. The problem that we found with the method was its inability to maintain stability as the system was advanced in time. This issue prompted us to search the RBF literature for ideas about how the instability could be resolved. During this search it became apparent that there were already many studies devoted to theoretical properties of the RBF method (e.g. its accuracy on an infinite integer lattice), but very few devoted to practical aspects of the RBF method. For example, our search did not turn up any study or even any comments on the RBF equivalent to the Gibbs' phenomenon or the Runge phenomenon, both of which have received numerous studies in the literature in connection with other interpolation methods (e.g. Fourier and polynomial methods). Concerned about this absence, we decided to embark on our own study of some basic properties of the RBF method.

One of the first properties that we focused on was the behavior of the RBF interpolants near boundaries. Like most interpolation methods, a common feature of the RBF method is how relatively inaccurate the interpolants are at boundaries. Various studies have made brief comments about this feature (including Hardy's initial study introducing the MQ method [36]), however none of these studies attempted to understand it or to propose any realistic techniques for improving the situation. When approximating nondissipative wave-type PDEs in a method-of-lines approach, any boundary-induced errors in the underlying method for approximating the spatial derivatives will eventually contaminate the solution everywhere across the domain. Thus, we set out to understand the properties of RBF interpolants near boundaries and to develop some practical methods for increasing their accuracy there. The results of this study comprise the paper that is the topic of this chapter (see Appendix A for the complete paper).

Below we summarize each of the sections of this paper, including who contributed to the section, and, when appropriate, additional ideas that were considered but not documented in the paper.

2.2 Summary of paper

1 Introduction

This section describes the motivation for the paper from the standpoint of using the RBF method for numerically solving nondissipative wave-type PDEs. We advocate using the RBF method in a hybrid approach, where RBFs are used only in the geometrically complex portions of the domain and high-order finite difference methods are used in the geometrically simple portions. We illustrate this approach in `Figure 1.1`. Bengt Fornberg (BF) and Toby Driscoll (TD) proposed using this approach because it would keep the compuational cost associated with the RBF method low. The example in `Figure 1.1` was conceived by the author (GW).

We next discuss how relatively inaccurate RBF interpolants are near boundaries (as illustrated in `Figure 1.2`), and how this is a major issue for using the RBF method for solving nondissipative wave-type PDEs. Consequently, we make the case that it is important to understand how the RBF interpolants behave near boundaries and whether there are any ways to improve the accuracy there.

The rest of this section is devoted to summarizing the remaining sections of the paper.

2 Definition of RBF approximations

In this section we give the definition for the basic RBF method. We consider the basic method for all the types of RBFs even though there are possibilities this can lead to a singular problem (see Section 1.3 for more details). The purpose in doing this is so we can use the same method to compare the approximation properties of all the different RBFs. In addition, this also allows us later on to view the augmented RBF method (see Definition 7) as a possible way of improving the accuracy of RBF interpolants near boundaries.

3 Cubic RBFs and their connection to splines

Early on in our RBF studies, we realized that 1-D cubic RBF interpolants are very similar to cubic spline interpolants since both are based on interpolating data with piecewise cubic polynomials. Recognizing that there are vast amounts of practical knowledge available about cubic splines, we made a large effort to try and relate the two interpolants, in order to apply this practical knowledge to cubic RBFs. This effort proved very fruitful and ultimately led to the development of some methods for improving the accuracy of RBF interpolants (not just those based on the cubic RBF) near boundaries.

This section reports on our efforts in relating cubic RBFs and cubic splines.

3.1 Infinite interval

To simplify the problem, we first consider how the two interpolants are related on an infinite interval. For this case, BF noted that there is a very simple relationship between cubic B-splines and the cubic RBF. This relationship is given in `Theorem 3.1` and `Corollary 3.2`. These two results allowed us to find an expression for the cubic RBF interpolant to cardinal data (equal to 1 at one node and 0 at all the others), and allowed us to analytically study the Gibbs' phenomenon for cubic RBF interpolants. `Figure 3.1` summarizes these results. The results in the figure and the analysis of the Gibbs' overshoot were done by BF.

3.2 Boundaries

In this section we consider the harder problem of relating the two interpolants on a finite interval. The first issue we tackle is how to create a cubic spline interpolant that is equivalent to a cubic RBF interpolant. A cubic spline interpolant is only uniquely defined when one imposes two extra conditions on the interpolant. In contrast, a basic cubic RBF interpolant has no freedoms and is uniquely determined solely by the data. This property naturally led us to investigate whether the two free conditions for the cubic spline interpolant could be chosen so that it is equivalent to a cubic RBF interpolant. TD was able to find these two conditions, which are given in `(3.2)`. This was a key discovery because it clearly demonstrated how the basic cubic RBF interpolant inherently couples the two ends of the interval together in a very unnatural way. This nonlocal coupling is the cause of the boundary troubles prevalent in cubic RBF interpolants.

Typically, one does not impose such irregular conditions as `(3.2)` for a cubic spline interpolant, but instead imposes conditions that improve the accuracy of the interpolant near the boundaries. The most common conditions that are imposed are the "natural" and Not-a-Knot conditions. Since these conditions are directed at resolving the same boundary problem for cubic spline interpolants that cubic RBF interpolants exhibit, we realized that it would be invaluable to find the cubic RBF equivalent of these conditions.

Under the heading `Natural cubic spline`, we show how to arrive at the natural end condition for the cubic RBF. This work was done by GW and BF. The rather straightforward derivation shows that this condition leads to an interpolant that is equivalent to the one given by the augmented RBF method (see Definition 7). As predicted by the theory for natural cubic splines, the natural cubic RBF exhibits $O(h^2)$ convergence at the boundaries and $O(h^4)$ convergence in the interior (where h is the maximum distance between neighboring points).

To implement the cubic RBF equivalent to the Not-a-Knot end condition we need to take a somewhat different approach then what led to the natural end condition. To understand how the Not-a-Knot end condition is implemented, one has to recall the geometric interpretation of the RBF method. The method involves creating an interpolating function to a set of data by taking a linear combination of translates of a radially symmetric functions $\phi(r)$. The most common procedure is to center a $\phi(r)$ over each of the points where the interpolation conditions are to be imposed. However, this is not a requirement, we could just as easily center the $\phi(r)$ anywhere we choose (although, if the set of centers is different from the set of data points, common proofs for unique solvability no longer apply).

Under the section heading `Not-a-Knot cubic spline`, we describe how to implement the Not-a-Knot end condition for the cubic RBF. The argument for why this

reproduces a Not-a-Knot cubic spline is as follows (assuming again (3.1)):

> Within $[-1,1]$, $s(x)$ is a piecewise cubic with jumps in the third derivative at all the data points except at the first and last interior points. Thus, $s(x)$ satisfies all the requirements of a Not-a-Knot cubic spline. Since this is well known to be unique, it must be equal to $s(x)$.

The derivation of the cubic RBF equivalent to the Not-a-Knot end condition was done by TD. Unlike the natural end condition, the Not-a-Knot end condition does not require that the RBF interpolant be augmented by any extra terms, only that the RBF centers at the first and last interior data points be moved outside the interval. Another key property about the Not-a-Knot end condition is that in the interior of the interval, the interpolant is independent of where the two RBF centers are placed (as long as they're placed outside of the interval). However, outside of the interval, the RBF interpolant is affected by the placement of the centers. The Not-a-Knot cubic RBF interpolant typically exhibits $O(h^3)$ convergence at the boundaries and $O(h^4)$ convergence in the interior.

4 Other types of RBFs

The relationship between the cubic spline and the cubic RBF allowed us to do some pretty neat analysis. However, since many other types of RBFs exist, we were interested in understanding their properties as well. This section summarizes some basic properties of a few of these different types, based on some experiments we did and from the literature.

4.1 Quintic

We were interested in this RBF because of its similarity to the quintic spline. We give the equivalent natural end conditions for the quintic RBF (these were derived by GW), as well as the equivalent not-a-knot conditions (these were derived by TD). In Figure 4.1 we compare the stability of the quintic and cubic RBFs.

Although not included in the paper, we also derived the conditions to impose on the quintic spline in order to create a quintic RBF. A quintic spline has 4 free conditions that need to be imposed in order to make it unique. If we assume for simplicity (3.1), then imposing the following 4 conditions on the quintic spline will reproduce a quintic RBF:

$$\begin{aligned}
s^{(3)}(1) &= 3\left[\frac{9}{2}s''(1)+\frac{3}{2}s''(-1)-9s'(1)+6s'(-1)+\frac{5}{2}\left(s(1)+s(-1)\right)\right], \\
s^{(4)}(1) &= 3\left[3s''(1)+2s''(-1)-8s'(1)+7s'(-1)+\frac{15}{2}\left(s(1)+s(-1)\right)\right], \\
s^{(3)}(-1) &= 3\left[-\frac{9}{2}s''(-1)-\frac{3}{2}s''(1)-9s'(-1)+6s'(1)-\frac{5}{2}\left(s(-1)+s(1)\right)\right], \\
s^{(4)}(-1) &= 3\left[\ 3s''(-1)+2s''(1)+8s'(-1)-7s'(1)+\frac{15}{2}\left(s(-1)+s(1)\right)\right].
\end{aligned}$$

These conditions were derived by GW using a similar argument to the one given in Section 3.2 for the cubic spline conditions (3.2). These conditions show that the

quintic RBF interpolant has an even tighter coupling of the two ends of the interval than the cubic RBF interpolant.

4.2 Thin plate splines (TPS)

We included this RBF in our study because of its popularity in applications [3, 12, 70]. The TPS RBFs are piecewise smooth (like the cubic and quintic RBFs), with a jump in the second derivative at their center. We found that they typically result in less accurate approximations both in the interior [16] and at the boundary (as shown later in the paper). However, as Figure 4.1 shows they lead to a better conditioned linear system (see also [57]).

4.3 Wendland functions

We were interested in these RBFs because, unlike all the other types of RBFs we have included, they are compactly supported. This means that these RBFs can lead to a sparse linear system of equations for solving for the interpolation coefficients. We have found, however, that they require a compromise between good sparsity and good accuracy. Figure 4.1 illustrates the stability of these RBFs, both as the number of data points increases and as the support radius increases.

4.4 Multiquadric (MQ)

This RBF is of particular interest because it is used in many applications (as discussed in Chapter 1). It can provide spectrally accurate interpolants [57], and it features a shape parameter ε that significantly affects its stability [57] and accuracy [47]. Figure 4.1 illustrates the stability issues involved with the MQ RBF, both as ε decreases and the number of data points increases. The effect of the shape parameter is the main topic of the next two chapters.

4.5 Inverse quadratic (IQ)

Our initial interest in the IQ RBF came from our failed attempts at trying to find a closed-form equation for the MQ RBF interpolant to cardinal data on the infinite interval. BF found that if the MQ RBF was replaced with the IQ RBF then closed-form analysis was made simpler. The closed-form cardinal IQ RBF interpolant is given in (4.2). This equation was a key discovery because it allowed us to aquire some insight about the behavior of the interpolant as the shape parameter $\varepsilon \to 0$. We were able to find that while the RBF coefficients grow exponentially as $\varepsilon \to 0$ (see (4.4)), the IQ RBF interpolant remains bounded and in fact converges to the *sinc* function.

Even power RBFs, i.e $\phi(r) = r^{2k}$ for $k = 1, 2, \ldots$, were not considered in our study because they lead to singular problems. For example, $\phi(r) = r^2$ leads to a singular problem in 1-D when the number of data points N is greater than 3. This a consequence of the fact that N linear combinations of translates of a parabola is again a parabola. Since each translated parabola is described by three coefficients, at most three of them can be linearly independent. For a general k and dimension d, we find that at most

$$\frac{2k+d}{k+d}\binom{k+d}{d}$$

translated basis functions are linearly independent [14]. This formula was derived by BF.

5 Edge effects and possible remedies in 1-D

After developing the natural and Not-a-Knot end conditions for reducing the edge errors of cubic RBF interpolants, we realized that these conditions could be generalized to possibly reduce the edge errors further. In the first part of this section, we describe these generalizations. Following this, we describe another idea that is not related to splines, but to the classical boundary clustering technique for reducing edge effects in polynomial interpolation. We conclude the section with some numerical experiments demonstrating how each of the proposed boundary remedies affect the RBF interpolants.

5.1 Adding polynomial terms

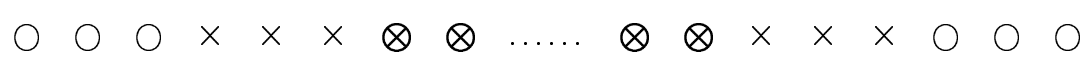
This idea was proposed by BF. It followed from the realization that agumenting the RBF with a linear and constant gives a natural spline, while augmenting it with polynomials from degree 0 up to $N-1$ (where N is the number of points) gives the standard Lagrange interpolating polynomial. For equispaced data, both the standard cubic RBF interpolant and the Lagrange interpolant behave poorly at the ends of the interval. We hoped there might exist a nice balance between the two that would result in better edge behavior. `Figure 5.1` demonstrates that the addition of up to only a quadratic appears to be beneficial. `Figure 5.2` also demonstrates this point by comparing the accuracy of derivative approximation. In order to maintain numerical stability in this type of approach, one should use Chebyshev polynomials $T_0(x), T_1(x), \ldots, T_k(x)$ instead of the standard polynomials $1, x, \ldots, x^k$, where $k < N$. The constraints, in this case become $\sum_{j=1}^{N} \lambda_j T_i(x_j) = 0$, for $i = 1, \ldots, k$. GW contributed to several of the numerical experiments that explored this idea.

While not included in the paper, we also applied this idea to the quintic RBF. We found that in this case the interpolant only benefited from the addition of up to a quartic polynomial.

5.2 Super Not-a-Knot(SNaK)

The Not-a-Knot end condition (enforcing C^3 continuity at the first two interior data points) has a regularizing effect on the cubic RBF interpolant in the vicinity of the edges. We realized that this idea could be extended further. To accomplish this, C^3 continuity is also enforced at the outermost two data points. This idea was proposed by BF and we refer to it as the SNAK condition. It leads to a cubic RBF interpolant with $O(h^4)$ accuracy throughout the entire interval and up to distance h outside (assuming the translated centers are moved at least that far). Table 2.1 compares the error in the SNaK cubic RBF approximation of the function $f(x) = 1/(1 + (x + 0.5)^2)$ sampled at equispaced points across $[-1, 1]$. The table shows that accuracy of the approximation is $O(h^4)$ both in the interior of the interval and a distance h outside the interval (although with a slightly bigger constant).

Extending the SNaK idea further by moving the centers at the second two interior data points outside the interval, i.e.



h	Error at $x = -1 - h$	Error at $x = -0.5$	Error at $x = 0.5$	Error at $x = 1 + h$
1/8	$1.7 \cdot 10^{-3}$	$2.2 \cdot 10^{-4}$	$2.7 \cdot 10^{-5}$	$1.1 \cdot 10^{-3}$
1/16	$9.6 \cdot 10^{-4}$	$1.6 \cdot 10^{-5}$	$1.3 \cdot 10^{-6}$	$9.7 \cdot 10^{-3}$
1/32	$9.3 \cdot 10^{-5}$	$9.7 \cdot 10^{-7}$	$1.0 \cdot 10^{-8}$	$2.1 \cdot 10^{-6}$
1/64	$6.5 \cdot 10^{-6}$	$6.0 \cdot 10^{-8}$	$6.9 \cdot 10^{-9}$	$1.6 \cdot 10^{-7}$
1/128	$4.3 \cdot 10^{-7}$	$3.7 \cdot 10^{-9}$	$4.5 \cdot 10^{-10}$	$1.1 \cdot 10^{-8}$
1/256	$2.7 \cdot 10^{-8}$	$2.3 \cdot 10^{-10}$	$2.9 \cdot 10^{-11}$	$7.2 \cdot 10^{-10}$

Table 2.1: Comparison of error in the SNaK cubic RBF approximation of the function $f(x) = 1/(1 + (x + 0.5)^2)$ over $[-1, 1]$ as the grid spacing is decreased. The is error computed at the interior points $x = \pm 0.5$, and at the exterior points $x = 1 + h$ and $x = -1 + h$.

results in an ill-conditioned problem. To see why this is true, consider only the left half of the interval. In this situation, the above illustration shows that a single cubic has to be fit between the four data points closest to the boundary. Since four points uniquely determines a cubic polynomial, there is no flexibility left in how this cubic joins the cubic that is defined just to its right. Extending the SNaK method even further by moving out the next center creates a singular problem, since this situation requires that a cubic polynomial interpolate five points.

5.3 Boundary clustering of nodes

BF and TD proposed testing the idea presented in this section because of its effectiveness at reducing boundary errors in 1-D polynomial interpolation. We performed various experiments with this method and found that the RBF interpolants usually benefit from some amount of grid clustering. However, too much clustering tends to create relatively large errors in the interior of the domain due to the depletion of data points there. Table 5.1 contains the results of an experiment we performed with this technique. This experiment was done by BF and GW.

5.4 Comparisons between the different edge improvement methods in 1-D

While all the edge improvement methods (apart from the Boundary clustering method) were developed for the cubic RBF, we realized that their implementation was in no way specific to the cubic RBF. In this section we study the effect of each of the methods on other types of RBFs. Figure 5.4 compares the four edge improvement on the Cubic, TPS, Wendland, and MQ RBFs. These experiments were done by GW. We make the following comments from these results:

- For the piecewise smooth RBFs, all the proposed edge improvement methods do in fact reduce the boundary error. In all cases, the best method is the SNaK method.

- The boundary clustering method, does work well at reducing the edge errors, however for the piecewise smooth RBF cases, it results in larger errors in the interior.

- The regular MQ method (i.e. no edge treatments) is more accurate at the edges than any of the other regular methods. We see that the addition of a constant and linear polynomial affects the accuracy very little, while the Not-a-Knot and SNaK methods actually make the edge error worse. For the 2-D examples we find the exact contrary of this latter result.

We also experimented with another edge improvement method for the cubic RBF, but did not include it in the paper because it had no direct application to the other types of RBFs. The derivation and experimentation with this method was done solely by GW. Like the natural and Not-a-Knot methods, this new method comes from cubic splines and is referred to as the "periodic" conditions. These conditions require that the first and second derivative at one edge equal the first and second derivative at the other edge, respectively. For example, for the interval $[-1, 1]$, the conditions are given by $s'(-1) = s'(1)$ and $s''(-1) = s''(1)$. These are the ideal conditions to impose on a cubic spline when the data to be interpolated is periodic over the interval. Just like the natural and not-a-knot conditions, the periodic conditions can also be implemented in cubic RBF terms. If we again assume that the data points are arranged as in `(3.1)`, then the periodic cubic RBF interpolant $s(x)$ is obtained by letting

$$s(x) = bx + cx^2 + \sum_{j=0}^{N} \lambda_j |x - x_j|^3 \ , \tag{2.1}$$

where

$$c = -\frac{3}{2} \sum_{j=0}^{N} \lambda_j \left[1 + x_j^2\right] \ , \tag{2.2}$$

and we require $\sum_{j=0}^{N} \lambda_j x_j = 0$. This can be easily verified by differentiating (2.1) for $x <= -1$ and for $x >= 1$. Figure 2.1 compares this condition to the regular, natural, Not-a-Knot, SNaK, and boundary clustered conditions, by comparing the corresponding approximations to the periodic function $f(x) = \exp(\sin(\pi x))$ at 21 points over $[-1, 1]$. The figure shows that the periodic conditions are clearly the best choice.

6 Edge effects and overall accuracy in 2-D

By first studying RBFs in 1-D, we were able to aquire quite a bit of understanding about their behavior at the edges of an interval, and we were able to develop some techniques for reducing their associated error there. In this section we study how our 1-D edge improvement methods generalize to 2-D and we compare the effects of these methods for some different RBFs. The two functions shown in `Figure 6.1` are used to compare the different methods. These functions were chosen because they give rise to two very different situations for the RBF approximations. The function in `Figure 6.1(a)` is very smooth and easily resolved with the 200 scattered data points shown in `Figure 6.2(a)`. In this situation, the edge errors will dominate the RBF approximations (as shown in the top row of `Figure 6.3`). The function in `Figure 6.1(b)` is

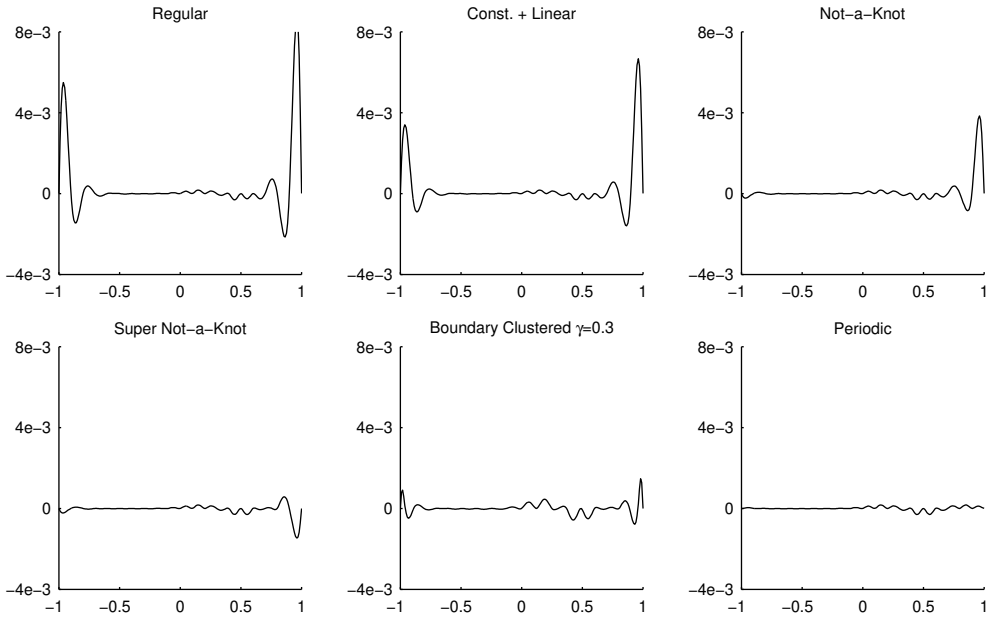


Figure 2.1: Comparison of the associated error in the different edge improvement methods for the 1-D cubic RBF approximation of the periodic function $f(x) = \exp(\sin(\pi x))$. In all cases the function was sampled at 21 data points across $[-1, 1]$.

too rough to be well resolved with the same set of data points, and the resulting RBF approximations will be dominated by interior errors (as shown in the top row of Figure 6.4). We felt it was important to see how our edge improvement methods behave in both of these situations.

6.1 Inclusion of low-order polynomial terms

In this section we describe how the inclusion of low-order polynomial terms extends to 2-D. While different types of extra terms and contraints can be used to augment the basic RBF interpolant, we show that low-order polynomial constraints occur naturally when one wants to regularize the far-field behavior of the RBF interpolant. Thus, the augmented RBF method specified in Definition 7, not only leads to a uniquely solvable problem (as shown in Theorem 8), but also leads to the regularization of the RBF interpolants far-field behavior. The derivation of the polynomial constraints from the far-field regularization argument was done by BF.

Figure 6.3 and Figure 6.4 show how the addition of a constant and linear polynomial affects the boundary errors for the different RBFs. These experiments were done by GW. Although not included, we also did several experiments where higher order polynomial terms were incorporated, but found that overall there was little benefit in going past the addition of a constant and linear polynomial. These experiments were performed by TD and GW.

6.2 Not-a-Knot and Super Not-a-Knot

This section is devoted to describing how the Not-a-Knot and SNaK methods can be implemented in 2-D. We illustrate our implementation of these methods in Figure 6.2(b) and Figure 6.2(c). In the third and fourth rows of Figure 6.3 and Figure 6.4, we show the resulting effects of these methods on the different RBF approximations. These experiments were done by GW.

There is a great deal more freedom in the 2-D implementation of the Not-a-Knot and SNaK methods than there is in 1-D. This freedom comes in two forms:

- the freedom in choosing which centers should be detached from their data points, and
- the freedom in choosing where to place the detached centers.

We did several numerical experiments to explore what happens to the RBF approximation in both of these situations. GW was significantly involved in these experiments together with TD and Richard Charles (RC). We summarize our observations from these experiments below:

- We found that it is more effective to detach the centers at the boundary than to detach the first "interior" centers. These detached centers should also be placed outside of the boundary.
- Increasing the distance the detached centers are placed from the boundary typically results in a larger reduction in boundary error when the function being approximated is smooth (this does not appear to be true for a rough functions). However, increasing the distance also causes a deterioration in the stability.

- Scattering the detached centers outside the domain typically reduces the effectiveness of the method and is more likely to cause instability issues.

We thus conclude that the Not-a-Knot and SNaK methods are most effective (in terms of accuracy and stability) when the detached centers are placed outside the boundary and near the location of their associated data points, as illustrated in Figure 6.2(b) and Figure 6.2(c).

6.3 Boundary clustering of nodes

In Figure 6.2(d) we illustrate how the boundary clustering method can be implemented in 2-D. The fifth row in Figure 6.3 and Figure 6.4, show the resulting effect of this method on the different RBF approximations. Experiments with this method were done by the TD and GW.

6.4 Comparisons between the different edge improvement methods in 2-D

In this section we summarize the results of our numerical experiments with the different 2-D edge improvement techniques. We conclude that when the underlying function being approximated is well resolved under discretization (as is the case for the function in Figure 6.1(a)), the SNaK method is generally the most effective technique for all the RBFs (except Wend22) at eliminating the errors near edges.

When the underlying function being approximated is not well resolved (as is the case for the function in Figure 6.1(b)), interior errors tend to dominate and none of the edge improvement techniques can offer any help against this.

In addition to the edge improvement methods discussed in the paper, we also considered a least squares method. This method consists of adding more centers than data points to the RBF approximation, which results in an underdetermined linear system for finding the expansion coefficients λ_j. The "missing" conditions are essentially used to minimize the sum of squares of these coefficients. We found that if the additional centers were placed outside the boundary of the domain, then a reduction of the edge error often resulted. Figure 2.2 illustrates this underdetermined least squares (ULS) technique when the approximation is done over the unit disk. In this example, we started with the 200 data points that were used in the 2-D examples from the paper, and added 25% more centers around the outside of the unit disk. Figure 2.3 shows the resulting RBF approximations of the smooth test function with this edge improvement method. The figure shows that the ULS method is effective for all choices of RBFs. Based on our experiments with this method, we found that increasing the number of additional centers that are placed outside the boundary by more than 50% of the total number of data points tends to hurt the overall accuracy of the approximations in the interior. In addition, for rough functions, like the arctan function from the paper, the ULS method is outright counterproductive since it reduces the flexibility of the approximations in the interior by imposing regularization at the boundaries. Experiments with the ULS method were done by GW, TD, and RC.

6.5 Accuracy comparison for RBFs against some other techniques

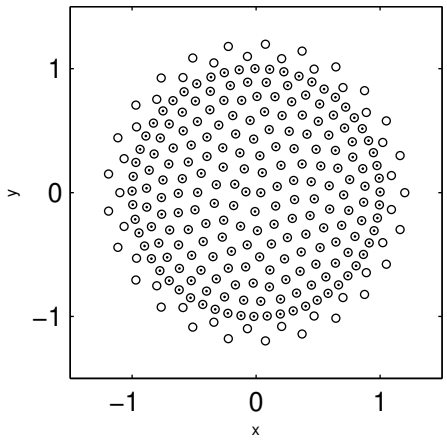


Figure 2.2: Illustration of the 2-D underdetermined least squares (ULS) boundary treatment method. The solid circles represent the locations of the data points while the open circles represent the locations of the centers.

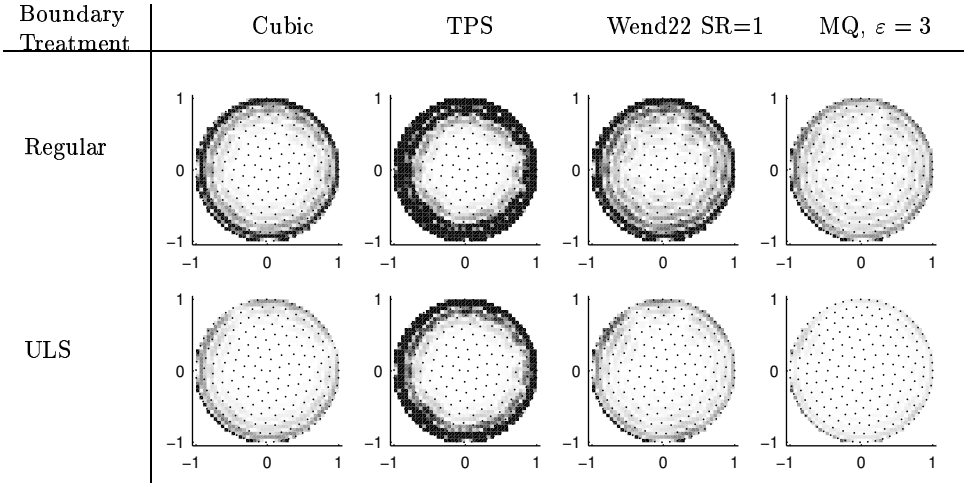


Figure 2.3: Comparison of the (absolute) error for the underdetermined least squares (ULS) boundary treatment method shown in Figure 2.2. In all cases the underlying function is $f(x,y) = \frac{1}{25+(x-1/5)^2+2y^2}$, and the black solid circles are the locations of the data points. Errors range from 0 (white) to $2.0 \cdot 10^{-5}$ (black).

This section compares MQ RBF and Fourier-Chebyshev pseudospectral approximations of the two 2-D test functions shown in `Figure 6.1`. We felt this comparision was important since the latter of these two approximation methods is well established as being particularly accurate. `Figure 6.5` contains the results of our comparison. We find that for the smooth test function the pseudospectral approximation is around two orders of magnitude more accurate than the MQ RBF approximation. But this should be expected, since pseudospectral approximations are well known to be extremely accurate for very smooth functions. It should be noted, however, that for an irregular domain, this comparison would not be possible since the pseudospectral method only works in highly regular regions. For the rough test function, we find that MQ RBF approximation is significantly more accurate. The experiments in this section were done by TD and GW.

Although not mentioned in the paper, we also felt it was important to compare the RBF method to other methods that do not require the data points to lie on a regular grid. An example, and popular choice, of these alternative methods is based on local cubic interpolation on a Deluany triangulation of the data points. MATLAB provides access to this method through its `griddata` function. Figure 2.4 compares this method to the MQ RBF method using the same two test functions as the example in `Figure 6.5`. We find that for both the smooth and the rough test functions, the MQ RBF method results in a better approximation (for the smooth case the RBF approximation is about 2 orders of magnitude more accurate). These experiments were done solely by GW.

7 Concluding observations

In this section we make some concluding remarks about the edge enhancement techniques that we studied in the paper. The general conclusion we reach is that, in the case of very fine resolution, the SNaK method is the most effective for all the RBFs we considered (apart from the Wendland function). RBF approximations with this method are more accurate at approximating the underlying function and its derivatives near edges of compuational domains.

In [46], Elisabeth Larsson and BF study the SNaK technique when applied to three of the "standard" RBF methods for solving elliptic PDEs. They find that all three of these methods are improved by the SNaK technique.

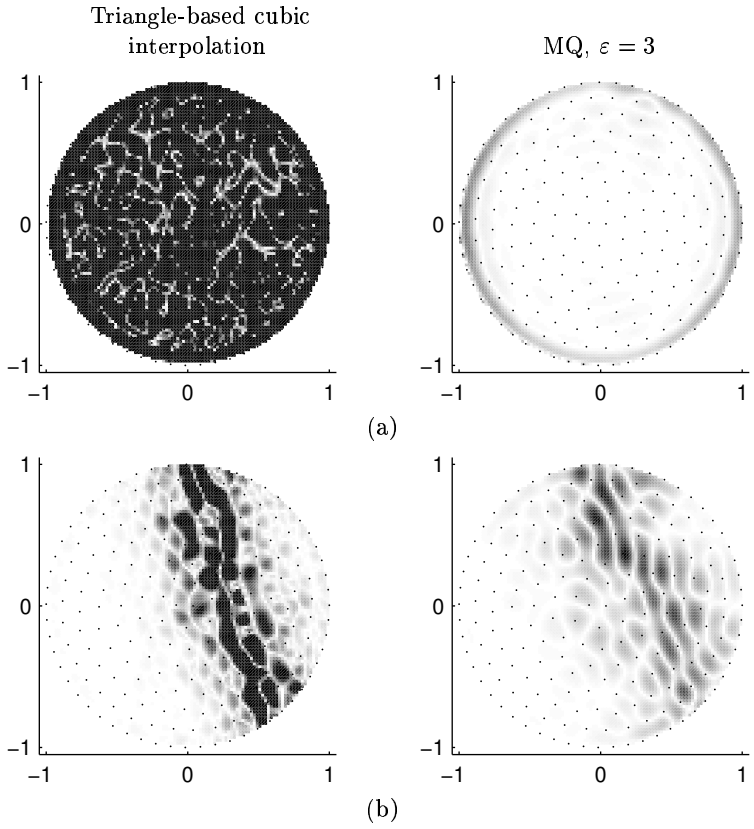


Figure 2.4: Comparison of the triangle-based cubic interpolation method used by MATLAB's `griddata` function to the MQ RBF method. In both cases the approximations are based on the 200 scattered data locations represented by the black solid circles. Part (a) shows the (absolute) errors, ranging from 0 (white) to $2.0 \cdot 10^{-7}$ (black), to the function $f(x,y) = 1/(25+(x-1/5)^2+2y^2)$. The Super Not-a-Knot boundary treatment has been applied to MQ RBF approximation in this part. Part (b) shows the errors, ranging from 0 (white) to 0.01 (black), to the function $f(x,y) = \arctan(2(x+3y-1))$. No boundary treatment has been applied to the MQ RBF approximation in this part.

Chapter 3

Stable Computation of Multiquadric Interpolants

3.1 Motivation

During our investigation of some basic properties of the RBF method, we became very interested in interpolants based on infinitely smooth RBFs (e.g. IQ, MQ, and GA). Our particular interest was in the effect of the shape parameter ε on the interpolants. Based on some numerical and analytical experiments, Bengt Fornberg (BF) conjectured, and partly resolved, that under some mild restrictions on the RBFs, the 1-D interpolant converges to the standard Lagrange interpolation polynomial as $\varepsilon \to 0$. The proof of the result was later completed by Toby Driscoll (TD) and is given in [14]. In higher-D, the limiting interpolant (when it exists) will be a multivariate polynomial [45, 57]. This is discussed more in the next paper (Chapter 4).

Recall from Chapter 1, that the linear system (1.9) for finding the expansion coefficients approaches singularity very rapidly as $\varepsilon \to 0$. However, from the above results we know that reagardless of this ill-conditioning the RBF interpolants (usually) converge to a well conditioned function (a polynomial). This remarkable result means that to compensate for the divergence of the expansion coefficients, an extreme amount of numerical cancelation must occur in the sum of the RBF interpolant. What this implies is that computing the RBF interpolant directly from the expansion coefficients is a numerically ill-conditioned step in an overall well conditioned process.

Motivated by this observation, we set out to find an algorithm that bypasses this numerically unstable step, and allows for the stable computation of RBF interpolants for all values of ε, including the limiting $\varepsilon = 0$ case. In the paper (see Appendix B) which is the topic of this chapter, we describe such an algorithm. Although our method appears to be limited to relatively small data sets, it nevertheless forms a powerful (indeed, the **only** existing) numerical tool for studying the limit of RBF interpolants as $\varepsilon \to 0$.

The ill-conditioning that is inherent in the computation of the expansion coefficients, has led to a permeating 'theory' in the RBF literature that it is impossible to attain both numerical stability and high accuracy when computing RBF interpolants [57]. Our numerical method proves this theory wrong by showing it is possible to bypass the ill-conditioning problem and to compute the RBF interpolants in a stable manner even in the limit of flat basis functions. This algorithm forms the first direct counter example to this 'theory', and therefore suggests that an entirely stable algorithm for any number of points may exist.

Below we summarize each section of the paper in Appendix B and, like before, include comments about individual contributors, and, when appropriate, additional

ideas related to the section. Following this summary, we describe in Section 3.3 some additional results related to the topic of the paper.

3.2 Summary of paper

1 Introduction

The first part of this section describes the three main difficulties with computing RBF interpolants, and states that our focus is on resolving the ill-conditioning issue related to small values of ε (for studies related to resolving the other two issues see [5, 7, 17, 20]). We follow this by discussing the motivation for the paper and reveiwing some analytical and numerical results regarding the parameter ε in RBF interpolation.

In the second part of this section, we describe the main idea of the algorithm. We note that the algorithm is in no way specific to the MQ RBF and can—without any change—be applied to many different RBFs. The RBFs we give results for are the MQ, IQ, and GA. In order to simplify the description of the algorithm, we initially only consider the basic RBF method. `Section 6` contains some comments about using the augmented RBF method with our algorithm. Equation `(2)` is very important because it illustrates how our method works. The insight that the RBF interpolant can be written in the form stated in `(2)` was made by BF.

We next comment on some important and unresolved issues related to the limiting RBF interpolants, for which our algorithm will now permit numerical explorations. Although not included in this paper, some of these unresolved issues have since been studied by our research group. The issue related to generalizing pseudospectral methods was studied somewhat in [46] for solving Poisson's equation. The issue related to finite difference methods is discussed in Chapter 5.

The remainer of the section is devoted to outlining the rest of the paper.

2 First test problem

In this section the test problem that we use to describe our algorithm is presented. The test problem consists of computing the MQ RBF interpolant to the function given in `(3)`, based on the 41 data points distributed over the unit disk shown in `Figure 1`. This test example was developed by the author (GW). In `(6)`, we give the vector-form of the equation for evaluating the interpolant at some point $\underline{x}$ and for some value of ε. We refer to this equation as the "direct method" of evaluating the RBF interpolant. `Figure 2` displays the magnitude of the error in the MQ RBF interpolant as a function of ε when computed with the direct method. The figure illustrates the inherent numerical ill-conditioning in the direct method as ε decreases. GW contributed this figure. To demonstrate this ill-conditioning, we used Rouché's theorem to numerically compute the leading order of ε in the determinant of the $A(\varepsilon)$ matrix given in `(4)`. This compuation was done by the GW with some assistance from BF. We found that $\det(A(\varepsilon)) = \alpha \cdot \varepsilon^{416} + O(\varepsilon^{418})$ (where α is a non-zero constant).

The function given in `(7)` is very important to understanding the numerical ill-conditioning issues. Based on the result that $s(\underline{x}, \varepsilon)$ (usually) converges to a polynomial interpolant as $\varepsilon \to 0$, the entries of $C(\varepsilon)$ should be $O(1)$. However, for small ε, the entries of $A(\varepsilon)^{-1}$ are typically $O(\varepsilon^{-p})$, where p is positive integer that increases rapidly

with the number of data points, and the entries of $B(\varepsilon)$ are typically $O(1)$. Thus, as stated in the motivation section of this chapter, there must be an extreme amount of cancelation occurring in forming the product $B(\varepsilon) \cdot A(\varepsilon)^{-1}$. For small values of ε the direct formation of $A(\varepsilon)^{-1}$ is an extremely ill-conditioned problem, and thus numerical instabilities will plague the computation of $C(\varepsilon)$. In `Section 4` we describe how this ill-conditioning can be bypassed by instead evaluating $C(\varepsilon)$ at points along a circle in the complex ε-plane, at which $A(\varepsilon)$ is well-conditioned.

3 Test problem viewed in the complex ε-plane

Since the algorithm described in `Section 4` is based on computing the RBF interpolant in the complex ε-plane, we use this section to examine the behavior of $s(\underline{x}, \varepsilon)$ for complex values of ε. GW contributed significantly to the observations on the behavior of RBF interpolants in the complex ε-plane.

In `Figure 3` we show how the condition number of $A(\varepsilon)$ behaves in the complex ε-plane. The figure illustrates two points. First, $A(\varepsilon)$ becomes ill-conditioned equally severe in all directions as $\varepsilon \to 0$. Second, for some complex values of ε, $A(\varepsilon)$ is singular (these values are seen as the spikes in the figure). This second point means that Micchelli's nonsingularity results given in Theorem 5 and 8 do not extend to the complex plane. `Figure 3` was contributed by GW.

From `(7)`, we see that there are two types of singularities that can occur in $s(\underline{x}, \varepsilon)$. The first type comes from the basis functions themselves. For example, the MQ has a branch point singularity and the IQ has a pole singularity when $\varepsilon = \pm\frac{i}{r}$ (the GA has no singularities, apart an essential singularity at $\varepsilon = \infty$). We refer to these types of singularities as "trivial" since their location is easy to determine from the set of data points and the evaluation point. The second type of singularity comes from values of ε where $A(\varepsilon)$ is singular (as illustrated by the sharp spikes in `Figure 3`). We refer to these types of singularities as "non-trivial", since their location is not trivial to determine.

As stated in the paper, the non-trivial singularities of $s(\underline{x}, \varepsilon)$ can only be poles. This was orginally proved by BF using Cramer's rule. We present here an alternative argument based on the same idea, since it will be useful for discussion throughout the chapter. First, we note that `(7)` can be re-written as

$$C(\varepsilon) = \frac{1}{\det(A(\varepsilon))} \begin{bmatrix} \quad B(\varepsilon) \quad \end{bmatrix} \begin{bmatrix} \quad \text{adj}(A(\varepsilon)) \quad \end{bmatrix} \tag{3.1}$$

where $\text{adj}(A(\varepsilon))$ is the adjoint of the $A(\varepsilon)$ matrix. Let $\Gamma_{j,k}(\varepsilon)$ be the cofactors of $A(\varepsilon)$, then $\Gamma_{j,k}(\varepsilon) = \Gamma_{k,j}(\varepsilon)$ for $j, k = 1, \ldots, n$ since $A(\varepsilon)$ is symmetric. Thus, expanding $\det(A(\varepsilon))$ also in cofactors, gives the following result for the jth entry of $C(\varepsilon)$

$$C_j(\varepsilon) = \frac{\sum_{k=1}^{n} \phi(\|\underline{x} - \underline{x}_k\|)\Gamma_{k,j}(\varepsilon)}{\sum_{k=1}^{n} \phi(\|\underline{x}_j - \underline{x}_k\|)\Gamma_{k,j}(\varepsilon)} \tag{3.2}$$

Now the numerator and denominator of (3.2) are analytic everywhere apart from the trivial singularites of $\phi(r)$ on the imaginary axis. Thus, at every point apart from the trivial singularities, they have a convergent Taylor expansion in some region. None of the trivial singularities can occur at $\varepsilon = 0$ since this would require $r = \infty$. Hence, there can

only be a pole at $\varepsilon = 0$ if the leading power of ε in the denominator is greater than the leading power in the numerator. We usually find the leading powers are equal, making $\varepsilon = 0$ a removable singularity (in the next paper (Chapter 4) we study situations when this is not the case). Apart from $\varepsilon = 0$, and the trivial singularities, the only singularity that can arise in $C_j(\varepsilon)$ comes when $A(\varepsilon)$ is singular. Due to the analytic character of the numerator and denominator, this type of singularity can only be a pole. Although not pursued until the next paper (Chapter 4), (3.1) leads to a very useful formula for the cardinal RBF interpolant.

`Figure 4` illustrates the structure of $s(\underline{x}, \varepsilon)$ in the complex ε-plane. We included this figure since it demonstrates several key properties of $s(\underline{x}, \varepsilon)$. First it shows the trivial singularities (marked by $\times$'s) that can occur (since we are using the MQ RBF, these are branch points). For our test problem, these can never get closer to $\varepsilon = 0$ than $\pm\frac{i}{2}$. Second, it illustrates the non-trivial poles of $s(\underline{x}, \varepsilon)$ that can sometimes arise. These poles always exhibit a four-fold symmetry, due to symmetries of $s(\underline{x}, \varepsilon)$. Finally, it shows a contour which we can use for safe evaluation of $s(\underline{x}, \varepsilon)$.

4 Numerical method

This section contains the description of our numerical method, which we refer to as the Contour-Padé algorithm. The algorithm essentially consists of numerically computing the Laurent expansion of $s(\underline{x}, \varepsilon)$ and then recasting the negative powers of this expansion into Padé rational form, so that we can compute $s(\underline{x}, \varepsilon)$ in the form stated in `(2)`. In Figure 3.1, we use the test example from this section to illustrate how this method works. The algorithm was conceived by BF. The contributions of GW have come from exploring the viability of the method as well as providing an implementation. Below we discuss several points regarding the implementation.

For convenience, a basic flowchart showing the key parts of the algorithm is included in Figure 3.2. We briefly discuss the details of each of the blocks.

Determine the radius of the contour.

The method obviously requires us to choose a radius for the circular path to evaluate $s(\underline{x}, \varepsilon)$ on. The requirements on this path are

- avoid the region of numerical ill-conditioning, and
- not include any trivial singularities.

The circular path should also be selected so that it avoids any non-trivial poles. However, it is currently not possible to determine *a priori* the location of these poles. In the discussion for the last block of the algorithm, we describe a technique for determining if we have run too close to any of these poles, and we can then adjust the radius accordingly.

To satisfy the first two requirements, we first compute the pairwise distances between all the data points and the evaluation point (note that these have to be computed anyway). From these distances we can determine the closest trivial singularity, which we denote as ε_s. If standard 64-bit floating point is being used, then we make the requirement that the condition number of $A(|\varepsilon_s|)$ be less than $5 \cdot 10^{14}$. If this criteria is not met then the method should not be used because of numerical instabilities reduce the effectiveness of the method. If this criteria is statisfied, then we use a simple bisection

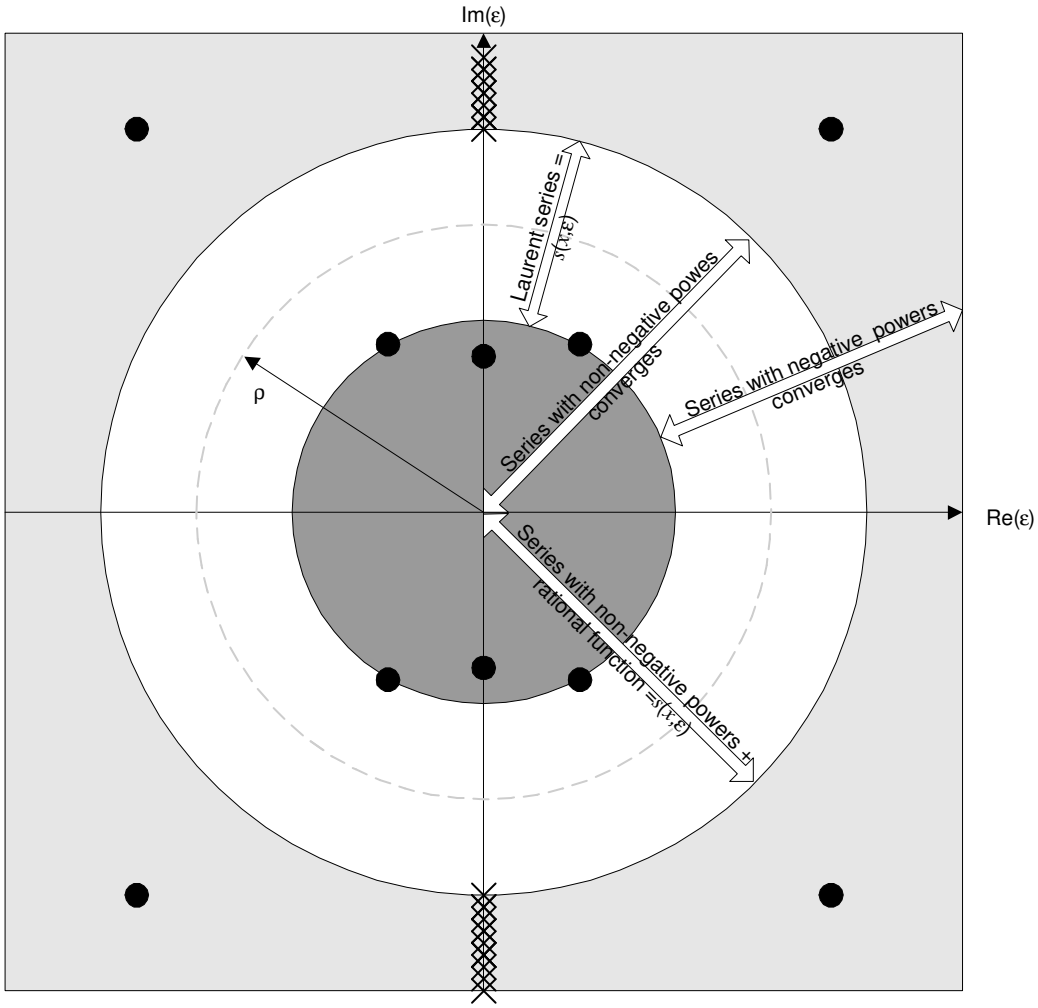


Figure 3.1: Illustration of how `(8)` can be used to compute $s(\underline{x}, \varepsilon)$ for all values of $\varepsilon \leq \rho$.

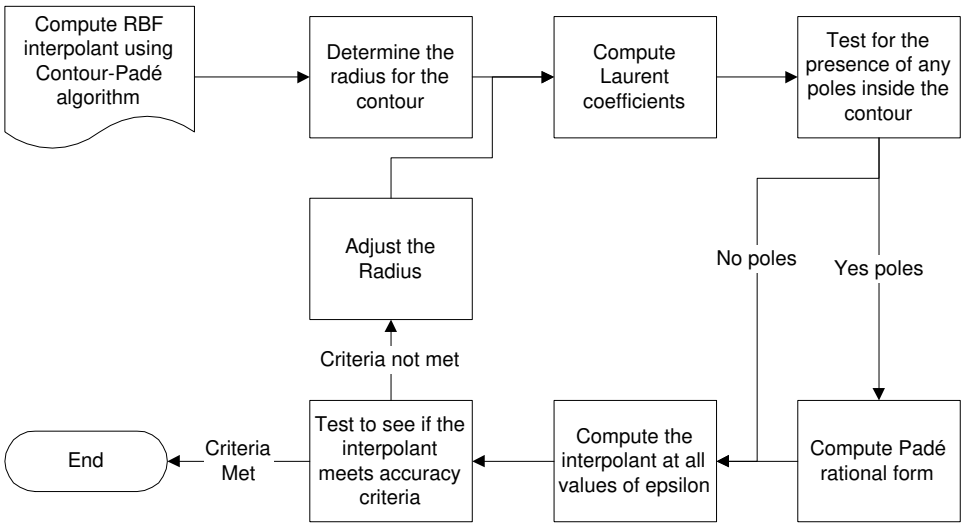


Figure 3.2: Flowchart for the Contour-Padé algorithm

method to find a radius that results in a condition number where the logarithm base 10 is less than $\frac{\log_{10}(\text{cond}(A(\varepsilon_s)))+15}{2}$ on the real axis. We are guaranteed to find a radius in this range, because of the condition number increases monotonely as $\varepsilon \to 0$. Based on numerical experiments, this method appears very practical for deterimining the radii. This technique was developed, and tested thoroughly by GW.

Compute Laurent coefficients.

In the paper we describe a general method for computing the Laurent coefficients of $s(\underline{x}, \varepsilon)$. However, when implementing the method, there are several properties of $s(\underline{x}, \varepsilon)$ that can be used to increase the stability and reduce the cost of the computation.

First, we note that for all standard infinitely smooth $\phi(r)$, the r-term in their equation is always multiplied by ε, e.g. $\phi(r) = \sqrt{1 + (\varepsilon r).^2}$. This property allows us to transform the evaluation of $s(\underline{x}, \varepsilon)$ around a circle of radius ρ to a circle of radius 1 by scaling the data points and the evaluation points by ρ prior to the computation (note that this also requires that we scale the values of ε we wish to evaluate the resulting interpolant at by $1/\rho$). By doing this we can avoid having to re-scale the Laurent coefficients by ρ after the inverse FFT, which allows us to circumvent any instabilities that may arise from dividing or multiplying the coefficients by very large numbers. This idea was developed by GW.

Another property that we can exploit, comes from the symmetry of the infinitely smooth $\phi(r)$. The standard infinitely smooth $\phi(r)$ that lead to a nonsingular RBF method, can be Taylor expanded as follows:

$$\phi(r) = a_0 + a_1(\varepsilon r)^2 + a_2(\varepsilon r)^4 + \ldots \, , a_k \in R \tag{3.3}$$

Consequently,

$$s(\underline{x}, -\varepsilon) = s(\underline{x}, \varepsilon) \, , \tag{3.4}$$

and

$$s(\underline{x}, \overline{\varepsilon}) = \overline{s(\underline{x}, \varepsilon)} \, . \tag{3.5}$$

These two properties mean $s(\underline{x}, \varepsilon)$ only needs to be computed over 1/4 instead of the whole circle. In addition, the Laurent series will only contain even powers of ε. This idea was developed by BF and GW.

The symmetry relations (3.4) and (3.5) also allow us to reduce the size of the inverse FFT that needs to be computed. If we have chosen M points to discretize around the circle, then we can exploit (3.4) and (3.5) to compute the Laurent coefficients with one inverse FFT of size $M/4 + 1$ (see [25, pp. 180–181]). This idea was developed by BF and EL.

If the exact coefficients of the Laurent expansion are $\tilde{d}_j$ for $j = 0, \pm 1, \pm 2, \ldots$, then the error in any of the approximate coefficients d_k is given by

$$d_k - \tilde{d}_k = \ldots + \tilde{d}_{k-2M} + \tilde{d}_{k-M} + \tilde{d}_{k+M} + \tilde{d}_{k+2M} + \ldots \, , \tag{3.6}$$

for $k = -\frac{M}{2} + 1, -\frac{M}{2} + 2, \ldots, \frac{M}{2} - 1, \frac{M}{2}$ (see for example [39, pp. 31–32]). Thus, to reduce the error of the aliasing terms, $s(\underline{x}, \varepsilon)$ needs to be sampled densely enough over the circle. Based on extensive numerical experiments we have found that $M = 128$ is typically sufficient. From the above discussion this means we only need to evaluate $s(\underline{x}, \varepsilon)$ at $M/4 + 1 = 33$ points over the circle and to compute one inverse FFT of the same size.

Another idea that was considered by GW for improving the accuracy of the Laurent coefficients was borrowed from [24]. The idea consists of using two circles of different radii to generate the Laurent coefficients of $s(\underline{x}, \varepsilon)$ and then use Richardson extrapolation on these values to eliminate the $\tilde{d}_{k\pm M}$ terms in (3.6). However, it was found that this idea was not viable for the RBF problem since it requires the two radii of the circles to differ fairly significantly [24]. We typically do not have the freedom in choosing radii of varying size because we need to avoid the computationally ill-conditioned region near $\varepsilon = 0$. In addition, if the second radii is chosen so that the resulting circle does not enclose the same singularities as the first, then a different Laurent expansion will be generated for $s(\underline{x}, \varepsilon)$. Thus, the two will not be able to be combined to eliminate the aliasing terms.

Test for the presence of any poles inside the contour.

In order to determine if there are any poles present inside the contour, we can compute the Laurent coefficients with M and $M/2$ points, and then compare the first few resulting negative power coefficients from each expansion. If there are poles present, then these coefficient will be approximately equal. If there are no poles, then the coefficients should not agree at all since they will simply consist of different aliasing errors from the coefficients of the Taylor part of the expansion. Both GW and BF contributed to this technique.

Another less expensive, yet less accurate way to test for the presence of poles is to check the magnitude of the first few terms of the negative power coefficients (from the expansion corresponding to the M points). If there are poles present, then the magnitude of the coefficients should be larger than the machine noise. Extensive numerical tests have shown that there does not appear to be a consistent level of significance for these magnitudes with which we can determine if a pole is present or not. This technique was contributed by GW.

Compute Padé rational form.

To compute the Padé rational form, we essentially follow the description given in the paper. However, there are again a few properties of $s(\underline{x}, \varepsilon)$ that we can exploit to our advantage. First, since the Laurent expansion only contains even powers of ε, we can reduce the cost of computing the Padé approximants by 1/2. In addition, from the symmetry relations (3.4) and (3.5) we know there is a four fold symmetry in the poles. Therefore, when computing the Padé approximants, we can choose the degrees of the numerator and the denominator to be equal. Both of these ideas were developed by GW and BF. For a description of Padé approximations, see [9, pp. 383–410].

In order to compute $s(\underline{x}, \varepsilon)$ for all values of ε inside the circle of radius ρ, it is necessary for the poles of the Padé approximant to match or exceed the total number of poles within the circle (cf. Figure 3.1). In order to satisfy this criteria we proceed as follows:

1) Initialize the degree to 2 for the numerator and denominator of the Padé approximant.

2) Compute the Padé approximant.

3) Convert the Padé approximant back to Laurent form.

4) Compare the coefficients from the negative powers of Padé approximants' Laurent expansion to those of the negative powers from the original Laurent expansion of $s(\underline{x}, \varepsilon)$.

5) If a sufficient number of coefficients beyond what was used in computing the Padé approximant are in agreement, then we have found the correct number of poles. If they are not then increase the degree of the numerator and denominator by 2 and repeat step 2.

This method was developed by GW and BF. In some numerical experiments that are presented in Section 3.3.2, we show that for scattered data points, the number of enclosed poles rarely exceeds 10.

The denominator of the Padé rational form $r(\varepsilon)$ will describe the pole locations of $s(\underline{x}, \varepsilon)$. From (3.2), we know that the non-trivial pole locations of $s(\underline{x}, \varepsilon)$ are entirely determined by the data points x_j. Thus, once $r(\varepsilon)$ has been determined for a given evaluation point $\underline{x}$, we can reuse its denominator for evaluating the interpolant at other evaluation points. This significantly reduces the cost of the method when the interpolant is to be evaluated at several different locations. This idea was developed by GW and BF.

Implementation detail:

It could conceivably happen that a zero in the denominator of (3.2) gets canceled by a simultaneous zero in the numerator for one evaluation point but not another. This would result in the denominator of $r(\varepsilon)$ for one point being different from the denominator for the other point. While we have only observed this phenomenon in very rare situations, the code nevertheless needs to handle it appropriately. Our way to do this is to compute the Padé denominators from a collection of evaluation points, and use the denominator that is most common between them.

Compute the interpolant at all values of ε

If no poles are contained within the circle then we can simply use the Taylor part of the Laurent series to compute $s(\underline{x}, \varepsilon)$ for any value of ε inside the circle. If there are poles present then we combine $r(\varepsilon)$ with the Taylor part to compute $s(\underline{x}, \varepsilon)$.

Test to see if the interpolant meets the accuracy criteria

There are two tests that we can use to verify that the $s(\underline{x}, \varepsilon)$ has been computed accurately. The first test simply involves comparing the direct computation of $s(\underline{x}, \varepsilon)$ at $\varepsilon = \rho$ (i.e. the radius of the circle) with the value computed using the Contour-Padé method. Based on extensive numerical tests, the difference between these to values should be less than 10^{-10} to validate the Contour-Padé computation.

The second test comes from exploiting the following property of the RBF method: if the data values being interpolated are given by $g_j = \phi(\|\underline{x}_j - \underline{x}_k\|)$ for $j = 1, \ldots, n$, $1 \leq k \leq n$, and some $\varepsilon = \varepsilon_t$, then for all $\underline{x}$, $s(\underline{x}, \varepsilon_t) = s_\phi(\underline{x}, \varepsilon_t) = \phi(\|\underline{x} - \underline{x}_k\|)$. Since the pole locations are entirely dependent on the data values, we can use the same radius ρ and the Padé denominator that was used to compute the original $s(\underline{x}, \varepsilon)$, to compute $s_\phi(\underline{x}, \varepsilon_t)$. If we choose $\varepsilon_t << \rho$, we can validate the original computation by testing how close $s_\phi(\underline{x}, \varepsilon_t)$ is to $\phi(\|\underline{x} - \underline{x}_k\|)$. Based extensive numerical test, the difference between these to values should be less than 10^{-8} to validate the Contour-Padé computation.

If neither of these validation criteria are met, then it is likely that the circular path used in the computation ran too close to some non-trivial poles. If this is the case, then we go onto the next block.

Both of these verification strategies were developed by GW.

Adjust the radius

Determining how to adjust the radius is a rather delicate process. Based on extensive numerical tests, some of which are discussed in Section 3.3.1 and 3.3.2, we have found that for scattered data sets, the poles do not appear to cluster in one area. In addition, the strengths of the poles do not appear to be very large (or very small) in magnitude. Thus, one very simple strategy for adjusting the radius, is to simply decrease it or increase it by a very small fraction from the original radius. With this strategy, care must be taken so that the circular path does not fall in the ill-conditioned region or near any of the trivial singularities.

GW has developed a MATLAB package for the Contour-Padé algorithm, called the Contour-Padé Toolbox. We intend to make this package freely available through

- the general RBF website (`http://rbf-pde.uah.edu`)
- the MATLAB Central website (`http://www.mathworks.com/matlabcentral/`)
- and through the websites of GW and BF.

The instructional manual for Contour-Padé package is included in Appendix D. The entire package (including the included manual) were developed solely by GW.

5 Numerical method applied to the test problem

In this section we apply the Contour-Padé algorithm to the test problem from `Section 2`. All of the work in this section was done by GW.

To determine how accurate the method is at computing $s(\underline{x}, \varepsilon)$ for small ε, we compare it against values computed using high-precision arithmetic (60 digits). We find that the Contour-Padé method computes the true value of $s(\underline{x}, \varepsilon)$ to at least 12 digits of accuracy. These results, along with a comparison of accuracy of the direct method of computing $s(\underline{x}, \varepsilon)$ are summarized in `Table 1`.

In `Figure 5`, we illustrate how the Contour-Padé algorithm allows the RBF interpolant to be computed in a stable manner for the full range of ε. This figure shows that the value of ε that results in the most accurate interpolant is well inside the numerically ill-conditioned region of the direct method. Without the Contour-Padé algorithm, standard methods for locating the optimal value of ε (e.g. [11, 23, 56]) would never be able to find the true optimal value for this example because of the ill-conditioning problems.

We conclude this section with a comparison of the computational effort required to compute $s(\underline{x}, \varepsilon)$ with the direct method and with the Contour-Padé method. The direct method requires that high-precision arithmetic be used. As the timing results show, this severely increases the time required for computing the interpolant as ε decreases, and makes the direct method completely impractical for exploring the small ε behavior. In contrast, the Contour-Padé algorithm can be carried out entirely using standard 64-bit floating point, and the time required for computing the interpolant does not increase as $\varepsilon \to 0$.

6 Some additional examples and comments

In the first part of this section we study another test problem. For this problem we use the Contour-Padé method for computing the MQ, IQ, and GA RBF approximations of (3) when sampled over the 62 data points shown in Figure 6. We compute the RMS error in the resulting RBF approximations in order to get a better idea of how the error behaves over the whole region. The results of the experiments are shown in Figure 8. We find that for the three types of RBFs, the Contour-Padé method computes the approximations in a stable manner as $\varepsilon \to 0$. In addition, the method allows us to compute the approximations at the optimal ε, where we find the RMS error is at least 2 orders of magnitude smaller than what could be achieved with the direct method. In Figure 7 we illustrate the structure of each $s(\underline{x}, \varepsilon)$ in the complex ε-plane. All of the work in this part of the section was done by GW.

The next example we explore was first discussed in [14], and consists of computing $s(\underline{x}, \varepsilon)$ with input data given by the 5×5 equispaced Cartesian grid over $[0, 1] \times [0, 1]$. This is an important example to study because it shows that the Contour-Padé method can also be used to compute $s(\underline{x}, \varepsilon)$ for cases when divergence as $\varepsilon \to 0$ occurs. It turns out that for this example, the IQ and MQ interpolants have a double pole at $\varepsilon = 0$ (the GA interpolant contains no such pole and in fact converges for this example). We show that the Contour-Padé algorithm automatically handles this situation, and display in Figure 9 the RMS error as $\varepsilon \to 0$ for the MQ RBF interpolant. Using the Contour-Padé algorithm we can get a better understanding of how $s(\underline{x}, \varepsilon)$ behaves in this example by computing each of the $d_k(\underline{x})$ functions in (12). The first 6 of these functions are displayed in Figure 10. One of the interesting features this figure shows is that the $d_0(\underline{x})$ interpolates the data and provides a very accurate approximation to the underlying function. This poses the interesting question of using $d_0(\underline{x})$ (with possibly higher order correction terms $d_2(\underline{x})$, $d_4(\underline{x}), \ldots$) in place of $s(\underline{x}, \varepsilon)$ when divergence occurs. In the paper that is summarized in the next chapter, we study convergence and divergence of the RBF interpolants as $\varepsilon \to 0$ in greater detail and link it to a condition known as "polynomial unisolvency". The work in this part of the section was done by the GW and BF.

We next comment on the augmented RBF method. The Contour-Padé method can be used for computing the augmented RBF interpolants by simply replacing $A(\varepsilon)$ and $B(\varepsilon)$ in (6) with the augmented counter parts (see (1.12)). We have found that the basic RBF interpolant and the augmented RBF interpolant are not significantly different for small values of ε (except when the GA is used; this is discussed in greater detail in the next chapter). The observations regarding the use of the Contour-Padé algorithm with the augmented RBF method were contributed by GW.

This section is concluded with another idea we attempted for computing the RBF interpolant for small values of ε. The idea is based on a Richardson/Romberg extrapolation method that uses larger values of ε to successively eliminate the low order positive powers in the expansion of the interpolant. Unfortunately the method is not practical for reasons outlined in the paper. This idea was explored by BF.

7 Concluding remarks

In this paper we have shown that the numerical ill-conditioning that has severely

limited computing RBF approximations for small values of ε can be circumvented by a novel choice of numerical algorithm. It is the first, and currently the only, numerical tool for studying the limit of the RBF interpolants as $\varepsilon \to 0$.

In [46], EL and BF also apply the Contour-Padé method for computing RBF based solutions of elliptic PDEs for the full range of ε. They find that the accuracy in some Poisson equation test cases far exceeds what can be acheived by any other general purpose method, especially when using ε-values that have previously been "off limits" due to ill-conditioning.

3.3 Additional results

This section contains some additional results that did not appear in the paper, but that are related to the topic of the paper. Unless otherwise noted, all the work contained in these sections was done solely by GW.

3.3.1 Some observations on the strengths of the poles

In order for the Contour-Padé algorithm to accurately compute $s(\underline{x}, \varepsilon)$ for small values of ε, it needs to be able to detect the non-trivial poles (if any) that are enclosed by the circle. If the strength of these poles is really small then their detection may be difficult. Additionally, if the strength of some of the poles is very large compared to any of the other poles, then detection of the smaller strength poles may also be difficult. In this section we numerically explore the strengths of the poles to see if these issues are common in the RBF method. Results are given for the MQ, IQ, and GA.

As discussed in `Section 3`, the non-trivial pole locations of $s(\underline{x}, \varepsilon)$ are determined solely by the data points. The strength of the poles, on the other hand, will vary based on the data points, the evaluation point $\underline{x}$, and the input data f_j. However, the input data has no real bearing on the issues of detecting the poles since we could use the Contour-Padé algorithm to compute the cardinal RBF interpolants and use these for constructing the interpolant to the input data. Thus, in our experiments we remove any dependencies on the input data by using the cardinal RBF interpolants to study the pole detection issues.

For the numerical experiments, we consider 4 different scattered data distributions of size $N = 28$, 45, 66, and 73 over the unit disk. The structure of $s(\underline{x}, \varepsilon)$ for these distributions and for the three types of RBFs is shown in Figure 3.3(a)-(d). From these figures, we see that each data distribution leads to four poles. Because of the four-fold symmetry of $s(\underline{x}, \varepsilon)$, it suffices to study the strength of the poles in the first quadrant (including the imaginary axis). We refer to the poles of each $s(\underline{x}, \varepsilon)$ that are closest to the origin (in the first quadrant) as Pole 1 and the remaining pole as Pole 2 (for the $N = 73$ distribution the poles on the imaginary axis are closest to the origin). For each distribution, we select a random evaluation point $\underline{x}$ in the unit disk and compute the set of all cardinal RBF interpolants $\{s_j(\underline{x}, \varepsilon)\}_{j=1}^{N}$, i.e. $s_j(\underline{x}_k, \varepsilon) = 1$ if $j = k$ and $s_j(\underline{x}_k, \varepsilon) = 0$ otherwise.

To study the first detection issue, we compute the minimum (in magnitude) strength of Pole 1 and Pole 2 from each set of cardinal interpolants. These results are given in the first part of Table 3.1. We see from the table that the minimum strength for all the distributions and RBFs is never less than 10^{-6}, and is typically on the order

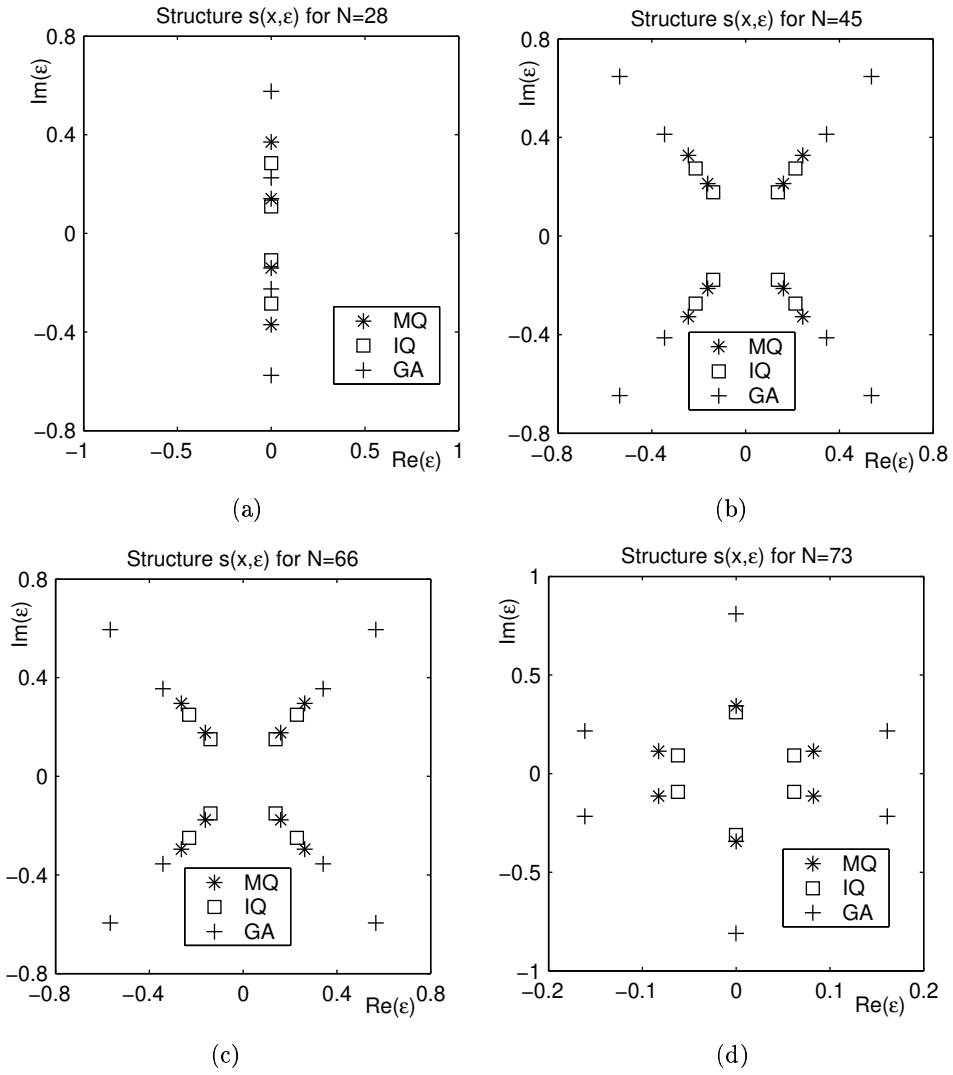


Figure 3.3: Structure of $s(\underline{x}, \varepsilon)$ in complex epsilon plane for data distributions of size (a) $N = 28$, (b) $N = 45$, (c) $N = 66$, and (d) $N = 73$.

RBF	Dist.	Minimum strength Pole 1	Minimum strength Pole 2	Strengths with the largest variation in size Pole 1	Strengths with the largest variation in size Pole 2
MQ	$N = 28$	$1.82 \cdot 10^{-5}$	$5.28 \cdot 10^{-4}$	$3.5 \cdot 10^{-4}$	$4.7 \cdot 10^{-1}$
	$N = 45$	$4.71 \cdot 10^{-4}$	$1.60 \cdot 10^{-3}$	$4.7 \cdot 10^{-4}$	$6.8 \cdot 10^{-3}$
	$N = 66$	$5.27 \cdot 10^{-5}$	$3.72 \cdot 10^{-4}$	$4.6 \cdot 10^{-4}$	$2.3 \cdot 10^{-2}$
	$N = 73$	$7.55 \cdot 10^{-5}$	$7.05 \cdot 10^{-4}$	$3.2 \cdot 10^{-2}$	$7.6 \cdot 10^{-5}$
IQ	$N = 28$	$8.39 \cdot 10^{-6}$	$1.57 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$	$3.1 \cdot 10^{-1}$
	$N = 45$	$4.28 \cdot 10^{-4}$	$1.06 \cdot 10^{-3}$	$4.3 \cdot 10^{-4}$	$6.1 \cdot 10^{-3}$
	$N = 66$	$4.79 \cdot 10^{-5}$	$1.71 \cdot 10^{-4}$	$3.6 \cdot 10^{-4}$	$2.0 \cdot 10^{-2}$
	$N = 73$	$1.13 \cdot 10^{-4}$	$8.60 \cdot 10^{-4}$	$3.7 \cdot 10^{-2}$	$1.1 \cdot 10^{-4}$
GA	$N = 28$	$3.79 \cdot 10^{-6}$	$8.49 \cdot 10^{-5}$	$9.0 \cdot 10^{-5}$	$2.4 \cdot 10^{-1}$
	$N = 45$	$3.38 \cdot 10^{-4}$	$6.92 \cdot 10^{-4}$	$5.1 \cdot 10^{-3}$	$1.1 \cdot 10^{-1}$
	$N = 66$	$4.98 \cdot 10^{-5}$	$1.39 \cdot 10^{-4}$	$6.6 \cdot 10^{-4}$	$3.5 \cdot 10^{-2}$
	$N = 73$	$1.20 \cdot 10^{-3}$	$4.71 \cdot 10^{-4}$	$4.7 \cdot 10^{-4}$	$6.8 \cdot 10^{-3}$

Table 3.1: Comparison of some results concerning the strength of the poles shown in Figure 3.3. The first part of this table shows the minimum strength of Pole 1 and Pole 2 computed from the set of all cardinal RBF interpolants to the given data point distribution. The second part of the table shows the strengths of Pole 1 and Pole 2 from all the cardinal interpolants that have the largest variation in size.

of 10^{-4}. The results also show that the minimum strength does not decrease as the number of data points increases. Both of these results should lessen the concern about the possibility of poles with very small strengths occuring in the RBF interpolants.

To study the second detection issue, we compute the strength of Pole 1 and Pole 2 from each of the cardinal interpolants and then compute which of these poles have strengths that vary the most in size. These results are shown in the second part of Table 3.1. We see from the table that the maximum difference is in size is typically between 10^{-1} and 10^{-2}. In addition, the size difference is not increasing with the number of data points. These results should lessen concerns about the possibility of the strength of some of the poles being very large compared to any of the other poles in the RBF interpolants.

3.3.2 Some observations on the distribution of poles for scattered data

Another potential concern with the Contour-Padé method pertains to the number of poles that occur within the circular path for evaluating $s(\underline{x}, \varepsilon)$. . If the number is really large then accounting for all the poles with the Padé procedure can be numerically difficult. Additionally, if there are large numbers of poles, it may make choosing the circular path very difficult. In this section we make several observations regarding the location and the number of enclosed poles for scattered data sets of varying size. Based on the numerical experiments that are presented, we find that the concerns over large numbers of poles appear to be unfounded. Since the pole locations of the IQ and MQ RBFs are very similar, we restrict our attention to the IQ and GA RBFs.

The observations that we make are based on several numerical experiments. For the numerical experiments that follow, we use data collected from 4 sets, each of which contain 100 different scattered data point distributions across the unit circle. The individual sets correspond to data point distributions of size $N = 21, 28, 36$, and 45. For the distributions in each of the 4 sets, the Contour-Padé algorithm was used to compute the location of the poles. In all cases, the radius of the circular path used in the computations was fixed at $\varepsilon = 0.46$ for IQ and $\varepsilon = 1.3$ for GA.

3.3.2.1 Distances of the poles from the origin

For this experiment we computed the distance all the poles are from the origin. The goal was to see if the number of poles near the origin increases as the number of data points increases and to see if there are any obvious patterns that arise in the distribution of the poles from the origin. Figure 3.4 shows a histogram of the number of poles that occurred at various distances from the origin for both IQ and GA. The bin size for each of the histograms is 0.05.

The figure shows that there is no trend upwards in the number of poles that occurred near the origin for either IQ or GA. We do however see a trend upwards in the number of poles occurring for IQ at distances above 0.35. The table below summarizes the minimum of all the distances from each of set of 100 point distributions.

	Minimum Pole Distance from the Origin			
	$N = 21$	$N = 28$	$N = 36$	$N = 45$
IQ	5.7×10^{-3}	4.7×10^{-3}	9.1×10^{-3}	7.7×10^{-3}
GA	9.2×10^{-3}	1.2×10^{-3}	5.5×10^{-3}	1.2×10^{-2}

We see from the table that the minimum distance did not decrease as the number of points was increased. In addition the values seem to be staying relatively constant

3.3.2.2 Frequency that a given number of poles occurs

For this experiment, we study the frequency that a given number of poles occurrs in all of our distributions. This can also be viewed as the number of times that a given degree in the denominator of the Padé rational form (2) occurrs. The goal for this experiment was to see if a certain number of poles enclosed within the ε-circle occurs more than another. Figure 3.5(a) displays a bar graph of the results for both IQ (in gray) and GA (in white).

The figure clearly shows that for both IQ and GA the most common number of poles that occurred is 4. In addition, the number of occurrences of 4 poles increased for GA and decreased for IQ as the number of data points was increased. We also see that the GA interpolant lacks the large numbers of enclosed poles that the IQ interpolant exhibits. The GA interpolant only had 6 occurrence of 12 or more poles appearing whereas this occurred 24 times for the IQ interpolant. There is also a definite increase in the occurrences of larger numbers of poles for IQ as the number of data points was increased. Finally, the figure shows that the maximum number of poles to occur for the IQ interpolant is 18, while the maximum for the GA interpolant is 16.

The table below summarizes the total number poles that occurred for each set of 100 data distributions.

	Total number of poles			
	$N = 21$	$N = 28$	$N = 36$	$N = 45$
IQ	542	664	790	924
GA	468	478	524	534

We see that more poles occurred for IQ than for GA and that the number increased more rapidly for IQ as the number of data points was increased. This is a pretty neat observation. One might expect that since the radius of the circle we used for GA calculations in the Contour-Padé algorithm was so much larger than the one used for the IQ calculations, that this creates more space in the complex ε-plane with which to allow for poles. However, even though the radius was bigger for GA we don't find that we have more poles enclosed in the circle than we found for IQ. Although we only have 4 data points to make the observation, the total number of poles for IQ increased at a near linear rate with a slope of approximately 16.

3.3.2.3 Fraction of purely imaginary poles

Now that we have experimented with how frequent a given number of poles occurs, it seems natural to ask whether it is more common to find poles on the imaginary axis than off. This is the topic of this section and the results are displayed in Figure 3.5 (b).

The figure shows that when the distributions resulted in 4 enclosed poles, 100 percent of these poles appeared on the imaginary axis for both IQ and GA and regardless of the number of data points (we also see that this is the case for 2 enclosed poles, but we knew this had to be true by symmetry). The figure also shows that the percentage of purely imaginary poles increased for the 6 enclosed poles case as the number of data points was increased.

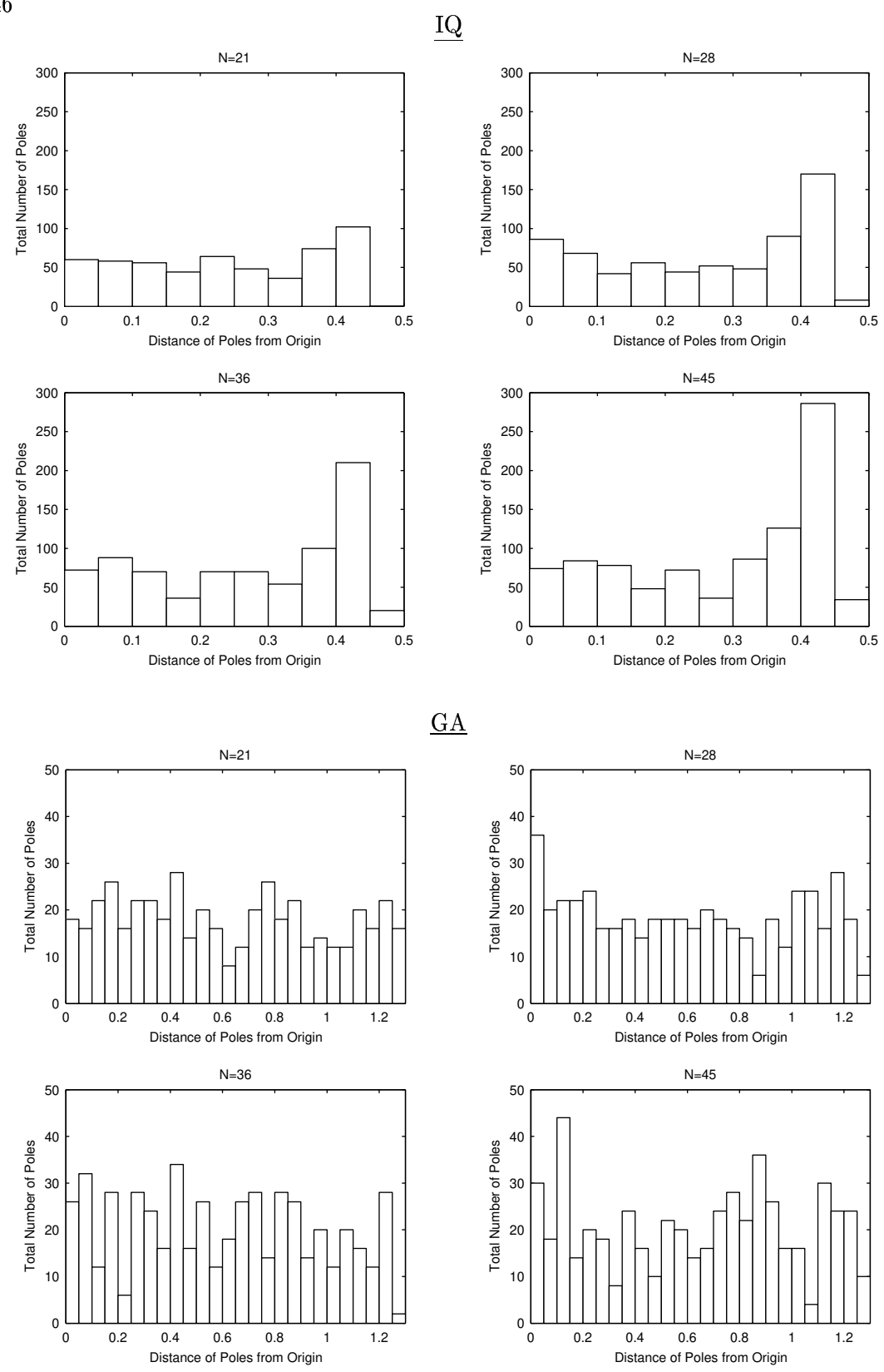
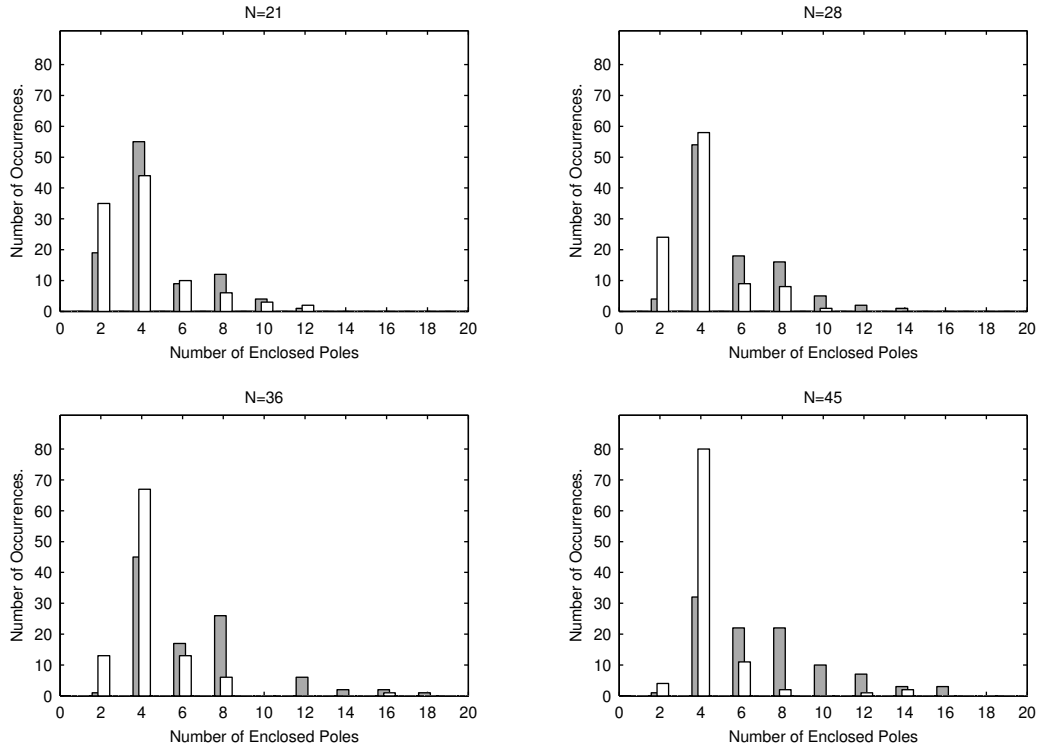
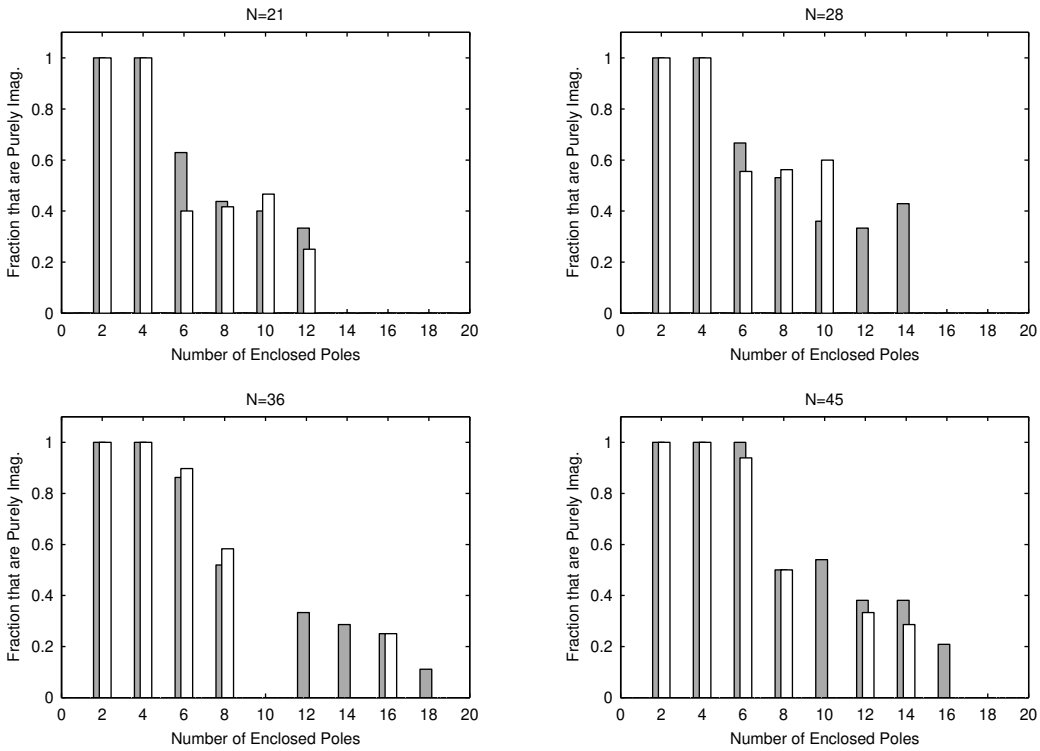


Figure 3.4: Histogram showing the number poles that occurred at various distances from the origin, as accumulated over 100 different data distributions.



(a) Frequency that a given number of poles occurs (Gray=IQ, White=GA)



(b) Fraction of the purely imaginary poles (Gray=IQ, White=GA)

Figure 3.5: (a) and (b), results for 100 different data distributions.

3.3.2.4 A simple search for dependencies

For this last section, we experimented to see if we could find any simple dependencies for the minimun distances of the poles from the origin and for the number of enclosed poles that occurred for a given distribution of data points. One property that we can compute for any distribution of data points is the minimum of all the pairwise distances between the points. Thus, it seems reasonable to test whether this property has any effect on either the minimum distance of the poles from the origin or the number of enclosed poles. Figure 3.6(a) and (b) display the results of this test for the $N = 36$ data point distributions. We can see from the figures that there does not appear to be any obvious dependencies on these variables.

3.3.3 An observation on the IQ RBF

When using the Contour-Padé algorithm, a radius needs to be chosen for the circular path that the interpolant is to be evaluated around. Recall from `Section 4`, that the requirements on the resulting circular path are that it avoids the ill-conditioned region of $A(\varepsilon)$ and that it does not contain any trivial singularities of $s(\underline{x}, \varepsilon)$. Since the trivial singularities of $s(\underline{x}, \varepsilon)$ come from the singularities of the RBFs (e.g. branch points for MQ), the second requirement can be avoided by using an entire RBF (e.g. GA). However, since a non-constant entire function grows without bound in the complex plane, entire RBFs lead to regions of ill-conditioning—apart from the region surrounding the origin—that must be avoided. For example, the GA grows very rapidly as the magntiude of the imaginary part of ε grows, and thus our circular path must avoid the region of ill-conditioning that arises for large imaginary values of ε. The fact that the RBFs either lead to trivial singularities of $s(\underline{x}, \varepsilon)$ or lead to regions of ill-conditioning in the complex plane, limit the size of the problem that can he handled by the Contour-Padé algorithm. Therefore, it could be potentially beneficial if both of these issues could be avoided. In this section we show how this is possible for the IQ.

For data points $\underline{x}_1, \ldots, \underline{x}_n$, and corresponding data values $f_1, \ldots, f_n$, the IQ interpolant can be written in the style of `(6)`, as $s(\underline{x}, \varepsilon) = B(\varepsilon) \cdot A(\varepsilon)^{-1} f$, where f is the vector of data values,

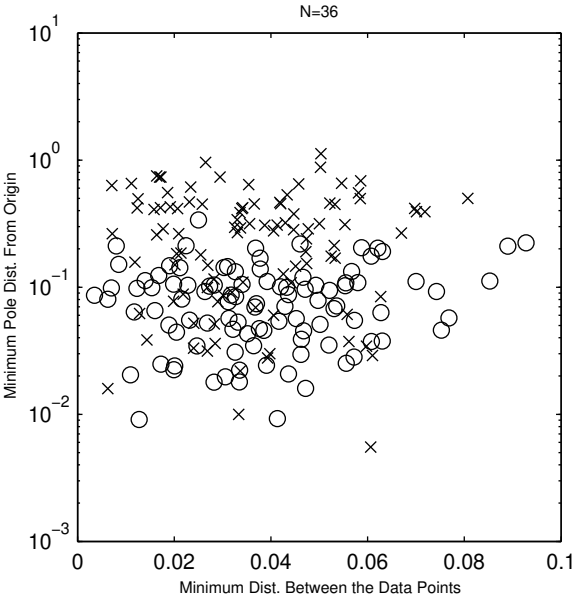
$$B(\varepsilon) = \begin{bmatrix} \frac{1}{1+\varepsilon^2\|\underline{x}-\underline{x}_1\|^2} & \frac{1}{1+\varepsilon^2\|\underline{x}-\underline{x}_2\|^2} & \cdots & \frac{1}{1+\varepsilon^2\|\underline{x}-\underline{x}_n\|^2} \end{bmatrix}, \tag{3.7}$$

and,

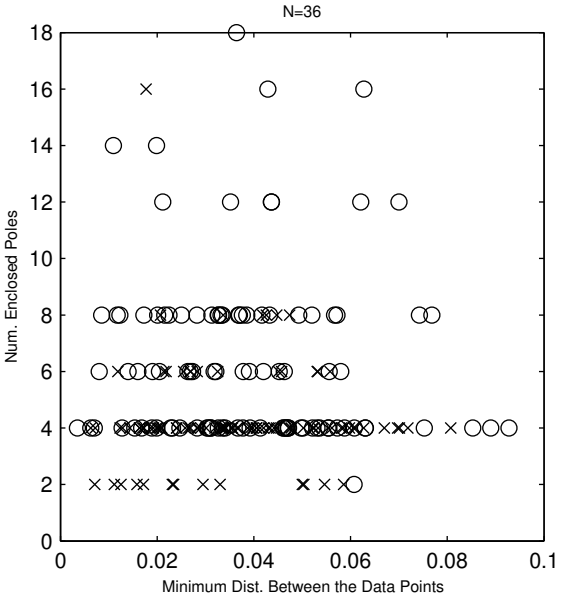
$$A(\varepsilon) = \begin{bmatrix} 1 & \frac{1}{1+\varepsilon^2\|\underline{x}_1-\underline{x}_2\|^2} & \cdots & \frac{1}{1+\varepsilon^2\|\underline{x}_1-\underline{x}_n\|^2} \\ \frac{1}{1+\varepsilon^2\|\underline{x}_2-\underline{x}_1\|^2} & 1 & \cdots & \frac{1}{1+\varepsilon^2\|\underline{x}_2-\underline{x}_n\|^2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{1+\varepsilon^2\|\underline{x}_n-\underline{x}_1\|^2} & \frac{1}{1+\varepsilon^2\|\underline{x}_n-\underline{x}_2\|^2} & \cdots & 1 \end{bmatrix}. \tag{3.8}$$

The function that is important to the Contour-Padé algorithm is $C(\varepsilon) = B(\varepsilon) \cdot A(\varepsilon)^{-1}$. Based on (3.7) and (3.8), one might be inclined to think that the trivial pole singularities of $s(\underline{x}, \varepsilon)$ come from the two sources

$$\text{(i)} \;\; \varepsilon = \pm \frac{i}{\left\|\underline{x}_j - \underline{x}_k\right\|} \quad \text{(ii)} \;\; \varepsilon = \pm \frac{i}{\left\|\underline{x} - \underline{x}_j\right\|}$$



(a)



(b)

Figure 3.6: Test for the dependence of some properies of the poles on minimum of the pairwise distances between all the points. In both (a) and (b) ◯ =IQ and × =GA

for $j,k = 1,\ldots,n$ and $j \neq k$. However, from (3.2) we know that the jth entry for $C(\varepsilon)$ is given by

$$C_j(\varepsilon) = \frac{\sum_{k=1}^{n} \frac{\Gamma_{k,j}(\varepsilon)}{1+\varepsilon^2\|\underline{x}-\underline{x}_k\|^2}}{\sum_{k=1}^{n} \frac{\Gamma_{k,j}(\varepsilon)}{1+\varepsilon^2\|\underline{x}_j-\underline{x}_k\|^2}} , \tag{3.9}$$

where $\Gamma_{k,j}(\varepsilon)$ is the cofactor corresponding to the k,j entry of $A(\varepsilon)$. Since the cofactors contain the pole singularities of type (i) and the same cofactors appear in the numerator as the denominator, the type (i) singularities should presumably cancel. Thus, $s(\underline{x},\varepsilon)$ will only contain trivial singularities of type (ii). It should be noted that it may be possible for cancelations of the trivial singularities of type (i) to occur in the denominator of (3.9) but not in the numerator, however, we have never observed this phenomenon.

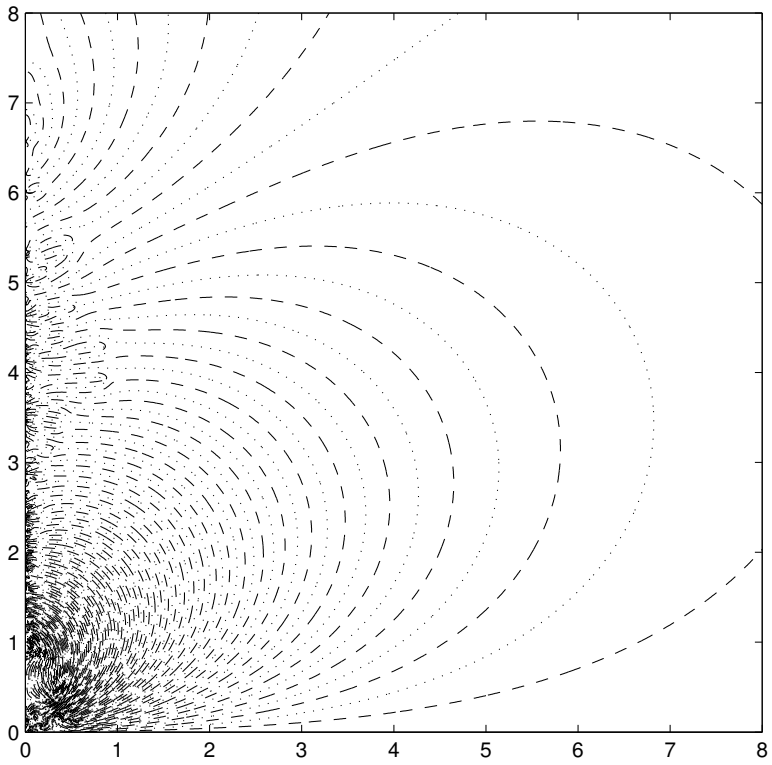
Based on the above observation regarding the trivial poles, a simple scheme can be devised for analytically removing them from $s(\underline{x},\varepsilon)$ during the Contour-Padé algorithm. This is done by using the function,

$$\tilde{s}(\underline{x},\varepsilon) = p(\underline{x},\varepsilon)s(\underline{x},\varepsilon) = \left[\prod_{j=1}^{n}\left(1+\varepsilon^2\left\|\underline{x}-\underline{x}_j\right\|\right)\right] s(\underline{x},\varepsilon) , \tag{3.10}$$

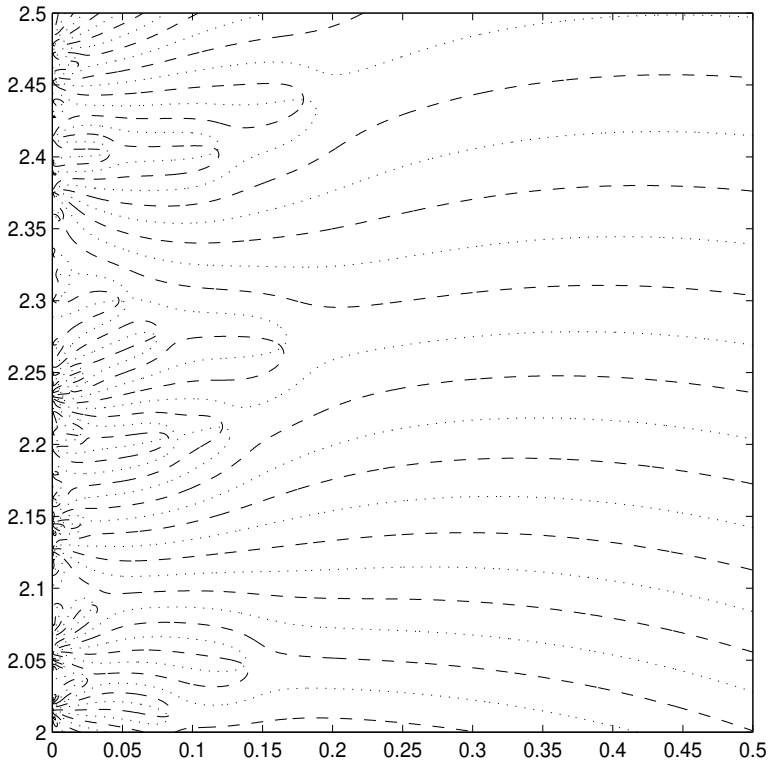
in place of $s(\underline{x},\varepsilon)$ in the Contour-Padé algorithm. This function (presumably) contains no trivial poles and thus is not restricted in the size of the radius that can be used. Once (3.10) has been computed by the Contour-Padé algorithm, simple division by $p(\underline{x},\varepsilon)$ gives us the original $s(\underline{x},\varepsilon)$. In practice, $p(\underline{x},\varepsilon)$ would really only need to contain roots for the poles that are enclosed within the circular path.

Although (3.10) allows us to use larger circular paths in the Contour-Padé method, these paths will be useless if there are large numbers of nontrivial poles of $s(\underline{x},\varepsilon)$ that get enclosed. In order to see if this can be expected, we consider a test example of 25 scattered data points over the unit square. From (3.9) we know that the non-trivial poles of $s(\underline{x},\varepsilon)$ are due to zeros of $\det(A(\varepsilon))$. Thus, to get an idea of the number of these poles we compute $\det(A(\varepsilon))$ over a very fine mesh covering $[0,8]\times[0,8]$ in the complex ε-plane; then plot the zero contours of the real and imaginary parts. These result are shown in Figure 3.7 (a) and on a more refined scale in Figure 3.7 (b). Both of these figures were contributed by BF. The trivial poles are located where the zero contours of the real and imaginary part of $\det(A(\varepsilon))$ cross. We can see from the figures that there are scores of non-trivial poles occuring near the imaginary axis. Because of the four-fold symmetry in $s(\underline{x},\varepsilon)$, the true number of poles will be even larger than what is shown. This is far too many poles for the Contour-Padé algorithm to account for. Based on this experiment, it appears that the idea of analytically removing the poles via (3.10) is impractical. However, the idea demonstrates that the circular path need not be limited by the trivial singularities of $s(\underline{x},\varepsilon)$.

To compare these results with the GA RBF, we used the same 25 data point example as above to compute the zero contours of the real and imaginary part of $\det(A(\varepsilon))$ for GA over a fine mesh covering the complex ε-plane. These results are shown in Figure 3.8 (a) and (b). Both of these figures were contributed by BF. The figure in part (a) shows that for GA, $\det(A(\varepsilon))$ contains large numbers of non-trivial zeros starting at a distance of around 3.5 from the origin and in the vicinity of the 45° line. This is



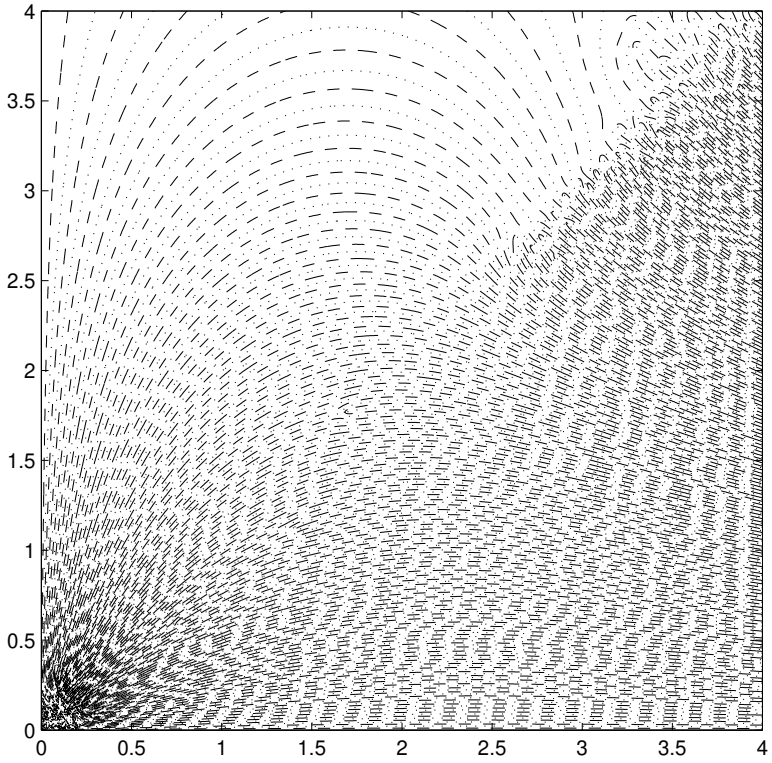
(a)



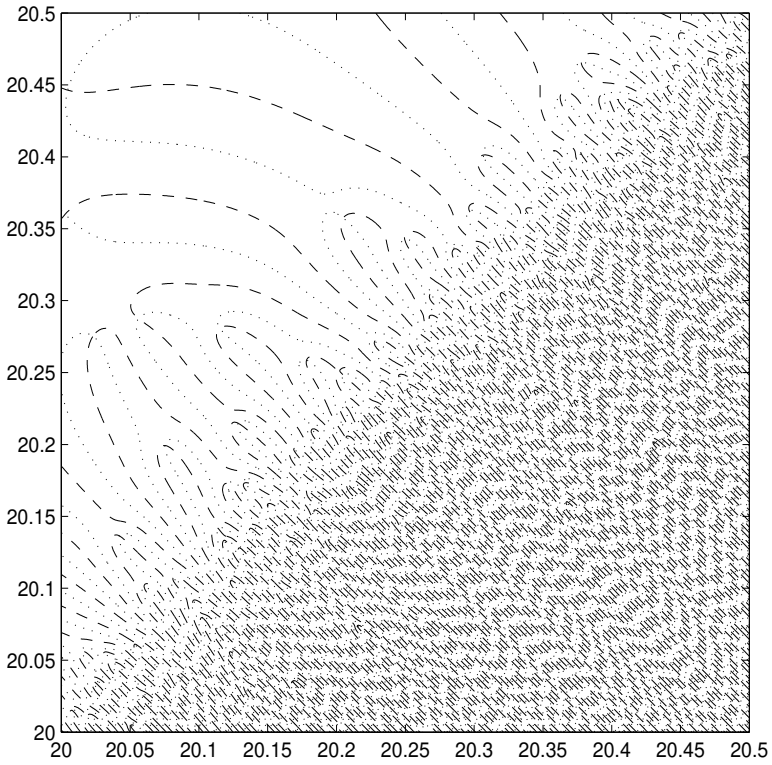
(b)

Figure 3.7: Contours of zero real part (dashed) and zero imaginary part (dotted) of $\det(A(\varepsilon))$ for the IQ RBF over the domain (a) $0 \leq |\text{Re}\ \varepsilon| \leq 8$, $0 \leq |\text{Im}\ \varepsilon| \leq 8$ and (b) $0 \leq |\text{Re}\ \varepsilon| \leq 2.5$, $2 \leq |\text{Im}\ \varepsilon| \leq 2.5$

very different from the IQ results, where the non-trivial zeros started appearing closer to the origin and in the vicinity of the imaginary axis. The figure in part (b) shows that number of non-trivial zeros increases significantly as the distance from the origin increases along this line. These results appear to suggest that using large circular paths for the GA RBF would also be impractical because of the number of non-trivial poles that would arise. Thus, even if the ill-conditioned region that results for the GA RBF for large imaginary values of ε could be avoided, we would still be limited in the size of the circular path that could be used in the Contour-Padé method.



(a)



(b)

Figure 3.8: Contours of zero real part (dashed) and zero imaginary part (dotted) of $\det(A(\varepsilon))$ for the GA RBF over the domain (a) $0 \leq |\text{Re}\ \varepsilon| \leq 4$, $0 \leq |\text{Im}\ \varepsilon| \leq 4$ and (b) $20 \leq |\text{Re}\ \varepsilon| \leq 20.5$, $20 \leq |\text{Im}\ \varepsilon| \leq 20.5$

Chapter 4

Some Observations Regarding Interpolants in the Limit of Flat RBFs

4.1 Motivation

Although 1-D interpolants based on smooth RBFs (usually) converge to the Lagrange interpolating polynomial as the RBFs become increasingly flat (i.e $\varepsilon \to 0$), the situation in higher-D is significantly different. As first observed by Toby Driscoll (TD) and Bengt Fornberg (BF) in [14], there are three different situations that can arise depending on the set of data points and the RBF used:

(a) the interpolants converge to the same finite degree polynomial limit

(b) the interpolants converge to a different finite degree polynomial limit

(c) the interpolants diverge.

Intrigued by these very different results, we began a study to investigate how each one can arise. Based on extensive analytical and numerical work (much of which was only made possible by the Contour-Padé algorithm), we were able to make a number of interesting observations about these three situations (especially (c)). In the paper (see Appendix C) which is the topic of this chapter, we report on some observations from this study.

Below we summarize each section of the paper in Appendix C and, like before, include comments about individual contributors, and, when appropriate, additional ideas related to the section.

4.2 Summary of paper

1 Introduction

In the first part of this section we briefly introduce the RBF method and discuss (or give references for) some of its properties. We focus our attention on the basic RBF method to simplify the discussion in the paper. However, we include some comments on the augmented RBF method in Section 6

The second part of this section discusses a number of intriguing (practical) features which have recently been found for RBF interpolants in the $\varepsilon \to 0$ limit. The majority of these features have been discovered by our RBF research group. In this paper, we discuss some more observations we have made regarding the limiting ($\varepsilon \to 0$)

RBF interpolants. One of our primary goals is to shed more light on the exceptional situations that lead to divergence as $\varepsilon \to 0$.

2 Closed-form expression for the RBF interpolant

The point of this section is to introduce the closed-form expression for the cardinal RBF interpolant given in Theorem 2.1. This simple expression is a key tool for understanding RBF interpolants as $\varepsilon \to 0$, and is used throughout the paper. One example of its value is given in Theorem 2.2, where it is used to prove that if an RBF interpolant converges in the $\varepsilon \to 0$ limit, then it will be a polynomial. Both Theorem 2.1 and Theorem 2.2 were discovered by BF.

It should be added that Theorem 2.1 can also be proven by means of (3.2) from the previous chapter.

3 A collection of examples with closed-form solutions for the $\varepsilon \to 0$ limit

In this section, we study a number of simple examples where closed-form analysis of the RBF interpolants is possible. These examples are key to understanding what happens in more general situations discussed in Section 4.

3.1 Three points along a straight line: Evaluation along the line

The example in this section was included to illustrate how the Taylor expansion coefficients of $\phi(r)$ influence the resulting RBF interpolant. When the inequalities on these coefficients are satisfied, we show how the 1-D RBF interpolant becomes equivalent to the standard Lagrange polynomial interpolant in the $\varepsilon \to 0$ limit (as the theorem in [14] predicts). The work in this section was done by BF and the author (GW).

3.2 Three points along a straight line: Evaluation off the line

The example in this section was included to demonstrate how the result from the previous example changes when the interpolant is evaluated at a point (x, y). As (3.1) shows, this situation leads to an additional y^2 term in the interpolant with a coefficient that depends on the choice of $\phi(r)$. The real importance of this example is in showing what values this coefficient takes for different $\phi(r)$. The result that the factor becomes 0 for the GA is essential for the results in Section 3.4. The work in this section was done by BF and GW.

3.3 Three points not on a line

The example in this section was included to show how the leading ε-term in the numerator and denominator depends on the location of the data points. When the three points are not collinear, we find that the leading power is ε^4, i.e 2 powers less than the previous examples. Additionally, this example shows that the limiting interpolant can behave discontinuously with respect to the data locations. The work in this section was

done by BF and GW.

3.4 Five or more points along a straight line

The example in this section proved to be crucial towards our understanding of the different behavior exhibited by the RBF interpolants as $\varepsilon \to 0$.

As noted previously, we had observed that certain data configurations give rise to a pole at $\varepsilon = 0$ (i.e. divergence in the interpolant as $\varepsilon \to 0$). However, due to the number of data points in these cases, we were unable to do any closed-form analysis. Thus, we set out on a search for a simpler example where such analysis would be possible. The search ended when GW discovered that 5 or more collinear data points would result in a pole at $\varepsilon = 0$ (for certain RBFs) when the interpolant was evaluated off the line. To our knowledge this is the simplest example to exhibit the $\varepsilon = 0$ pole phenomenon.

Due to the simplicity of this example, we were able to make several interesing observations about the limiting interpolants that had previously not been known. The most important of these being that the GA is inherently different than the other types of standard RBFs as $\varepsilon \to 0$. GW found that for the collinear example, the GA interpolant never appeared to diverge $\varepsilon \to 0$. This ultimately led GW to conjecture that the GA interpolants would never diverge for any number of collinear data points, regardless of where it was evaluated. This conjecture was ultimately proved by BF, and is given in `Theorem 3.1`.

Following the proof of `Theorem 3.1`, we make some comments about the possibility of using oscillatory $\phi(r)$ in the RBF method. This seems to have received very little consideration in the RBF literature, and appears to be a whole new research area to explore.

3.5 Some generalizations to higher dimensions

This section was included to show how some of our one-dimensional results from the previous section can be extended to higher dimensions. `Theorem 3.2` shows how the 1-D result for GA interpolants generalizes to higher-D rectangular lattices. This theorem was proved by BF following strong empirical evidence provided by GW.

In `Table 3.1`, we demonstrate how the divergence result from the previous section generalizes to the case where the data is scattered on a d-dimensional hyperplane. The table shows the lowest number of points that leads to divergence when the interpolant is evaluated off the hyperplane. Like the result from previous section, the GA interpolant does not exhibit divergence in any of these cases. GW contributed the results for this part of the section, including the suggested formula that predicts the lowest number of points to lead to divergence. The Contour-Padé algorithm played a key role in these observations.

3.6 Points placed along a parabola

This section was included to show how the results of our collinear data point example extend to more general curves, e.g. parabolas. We find that divergence of the interpolants can also occur when 8 or more points are placed on a parabola and the evaluation point off the parabola. As described in the paper, we observe a second order pole at $\varepsilon = 0$ in both the MQ and IQ interpolants for this example, but no pole for the

GA interpolant. The parabola example was contributed by Elisabeth Larsson (EL) and BF.

This example, and the collinear example from Section 3.4 are intended to illustrate how the failure of polynomial unisolvency (defined in Section 4) can lead to divergence.

Based on the examples and theorems discussed in Section 3, and a plethora of other numerical results from the Contour-Padé algorithm, we conjecture that the GA interpolants will never diverge as $\varepsilon \to 0$. This conjecture is due to GW and BF, and appears to be the first time it has appeared in the literature.

4 Polynomial unisolvency, and some results for scattered points

In this section, we introduce the idea of polynomial unisolvency (Theorem 4.1) and describe how it applies to the RBF interpolation problem. This connection proved to be the key in understanding the three different situations that had been observed to arise with RBF interpolants as $\varepsilon \to 0$ (cf. Section 4.1).

Theorem 4.4 describes when the different RBF interpolants will converge to the same finite degree polynomial limit as $\varepsilon \to 0$. Note that there are in fact two conditions on the data points for the theorem. The first is that their number n agree with the dimension of P_K. This means in 2-D that the number needs to be $n = 1, 3, 6, 10, 15, ...$ for $K = 0, 1, 2, 3, 4, \ldots$, respectively. For a general dimension d and degree K, the number is given by $n = \binom{K+d}{K}$. The second condition is that the points are unisolvent with respect to a basis for P_K. If neither of these conditions are satisfied then the different RBF interpolants may converge to different (finite degree) polynomials, or they may diverge as $\varepsilon \to 0$. We have only observed divergence when the second condition is violated, as illustrated in the collinear and parabola examples from Section 3.4 and 3.6. However, divergence is not always a result of this failure, as Table 3.2 and Example 4.5 illustrate.

GW was the first to make the connection between polynomial unisolvency and RBF interpolation as $\varepsilon \to 0$, and to show that divergence was associated with failure of this condition. Theorem 4.4 is due to EL and BF [45]. Example 4.5 was contributed by EL.

Although not directly mentioned in the paper, polynomial unisolvency also has an interesting effect on the stability of the linear system (1.9) for finding the expansion coefficients λ_j. To illustrate this effect, consider again the parabola example from Section 3.6. Table 4.1 compares the leading powers of ε that occur in $\det(A)$, and the leading inverse powers in λ_j for this example, against what occurs for scattered (i.e. polynomial unisolvent) points. The table shows that the λ_j for the parabola case will grow much more rapidly as ε decreases than for the scattered case. Numerical evidence strongly suggests that this phenomenon carries over to less contrived non-unisolvent examples as well. The effect of polynomial unisolvency on the stability of the RBF method was made by GW, BF, and EL. It does not appear to have ever been considered in the literature.

To illustrate just how sensitive the linear system (1.9) can be to the polynomial unisolvency condition, we compare the condition number of the A matrix (based on IQ), for $n = 18$ points on a parabola against $n = 18$ points that satisfy the polynomial unisolvency conditions. The points on the parabola are given by $x_k = \frac{k-1}{17}$, $y_k =$

	Type Distribution	n - number of data points								
		5	6	8	10	12	14	16	18	20
Leading power of ε in $\det(A)$	Scattered	12	16	28	40	56	72	90	110	130
	Parabola	12	18	32	50	72	98	128	162	200
Leading inverse power of ε in λ_j	Scattered	4	4	6	6	8	8	10	10	10
	Parabola	4	6	8	10	12	14	16	18	20

Table 4.1: Powers of ε arising in the case of 2-D scattered data (i.e. polynomial unisolvent) and data on a parabola (i.e. fails polynomial unisolvency). A is the RBF interpolantion matrix for finding the expansion coefficients λ_j.

$x_k^2,\ k = 1, 2, \ldots, 18$.. The unisolvent points are given by the same x coordinates but, with y given by $y_k = \delta_k x_k^2,\ k = 1, 2, \ldots, 18$, where the δ_k's are some random values satisfying $0 < \delta_k < 0.01$. Figure 4.1 displays the results for this example. The figure clearly shows that the condition number of the A matrix is growing much more rapidly for the parabola points as $\varepsilon \to 0$. This is quite amazing since the unisolvent data points are just a slight perturbation of the parabola points. This example was contributed by GW.

5 Complex ε-plane considerations and the numerical Contour-Padé algorithm

Our primary focus in this section is on the behavior of the poles in the complex ε-plane for non-unisolvent and unisolvent data points. One of the goals is to understand the situation where a pole arises at $\varepsilon = 0$. This section contains both analytical and numerical results (much of which were made possible by the Contour-Padé algorithm).

5.1 Numerical algorithm

The purpose of this section is to give a brief overview of the Contour-Padé algorithm, and demonstrate its usefulness in studying limiting RBF interpolants. As an example of the latter point, we use the algorithm to compute the cardinal IQ RBF interpolant for the data points shown in Figure 5.1. This distribution of data points results in the interpolant having a fourth order pole at $\varepsilon = 0$. We show how the algorithm can be used to compute each term in the series of (5.2), and display these terms in Figure 5.2. The figure illustrates a very interesting feature regarding the term $s_0(\underline{x})$ in (5.2). Specifically, it shows that this term actually provides a very good approximation to the underlying function (this was also discussed briefly in the last paper (Chapter 3)). This observation suggests the intriguing idea of using the $s_0(\underline{x})$ term, with possibly higher order corrections $s_2(\underline{x})$, $s_4(\underline{x})$, etc., when the data points lead to a pole at $\varepsilon = 0$. Presently, the Contour-Padé algorithm is the only tool available for this option. This section was contributed by GW and BF.

5.2 Pole locations in the complex ε-plane

In the subsections that follow we present some analytical and numerical results

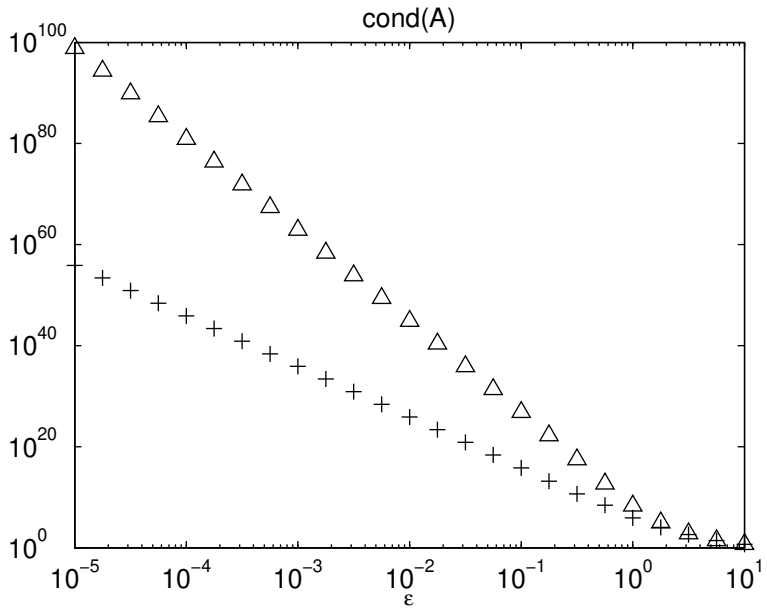


Figure 4.1: Comparison of the growth in condition number of the RBF interpolation matrix A (1.9) as a function of ε for $n = 18$ polynomial unisolvent points and $n = 18$ points on a parabola.

on the locations of the poles in the complex ε-plane. We show how these locations are important to understanding how a pole at $\varepsilon = 0$ can appear.

5.2.1 Points approach cases where polynomial unisolvency fails

The purpose of this section is to revisit a couple of the examples from Section 3 to see how the poles in the complex ε-plane behave as the data points approach cases where polynomial unisolvency fails. The work in this section was contributed primarily by BF, with some minor contributions by GW.

We chose the two examples of this section because they are simple enough for closed-form analysis, and demonstrate two very different situations that can arise when the points fail the polynomial unisolvency conditions. The first example demonstrates the situation where there are poles that approach $\varepsilon = 0$ as the points approach non-unisolvency, but become removable singularities when the points actually become non-unisolvent. The second example demonstrates the situation where there are again poles that approach $\varepsilon = 0$, but they become non-removable singularities when the points fail the unisolvency conditions. For both examples, the pole locations are accurately approximated for y_0 small by (5.5) and (5.7), respectively. Both of these approximations were contributed by BF.

While (5.7) gives an accuarate approximation to the MQ pole locations when y_0 is small, it does not show the intriguing behavior as y_0 varies from small to large. This result is displayed in Figure 4.2. The figure shows the pole locations as y_0 varies from 5×10^{-5} to 10. We see that as y_0 gets small the poles move in to the origin (as (5.7) shows). The interesting changes in the pole locations happen as y_0 varies from 0.01 to 5. We see that the poles make a nice loop and start returning back to the origin as y_0 gets big. The poles on the imaginary axis start small for y_0 small and then extend up (or down) the imaginary axis to their maximum (or minimum) when $y_0 \approx 2.92$; they then move back toward the origin as y_0 increases. The rate at which the poles approach the origin slows dramatically as y_0 gets large. As $y_0 \to \infty$ the interpolant ultimately diverges, but not due to a pole arising at $\varepsilon = 0$. This analysis was contributed by GW.

5.2.2 Points are located so that polynomial unisolvency fails

The intention of this section is to show how the poles behave in a larger region of the complex plane for points where polynomial unisolvency fails. The examples we use in this section contain far too many points to allow for closed-form analysis of the pole locations. Thus, we make extensive use of the Contour-Padé algorithm since it can be used to determine the pole locations.

For the experiments, we use several different data points that are arranged on a parabola. Our results are displayed in Figure 5.4, Figure 5.5, and Figure 5.6, for the MQ, IQ, and GA RBFs, respectively. We can see that the results for the GA interpolant are inherently different. Besides having no pole that ever appears at $\varepsilon = 0$, we find that there are far fewer poles that occur inside the circle. The exact numbers are summarized in Table 5.2.

All the results in this section were contributed by EL and BF.

5.2.3 Scattered points

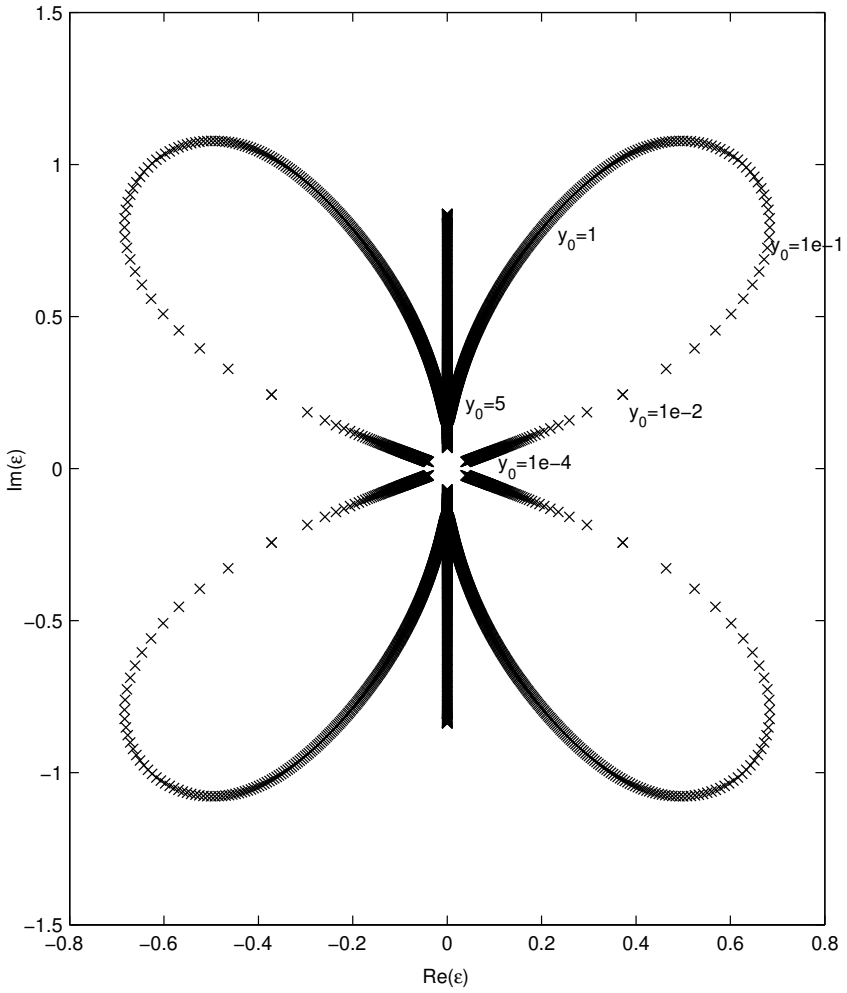


Figure 4.2: The trajectories of the 6 poles in the MQ RBF interpolant as a function of y_0, for the 5 *almost-collinear data points* example given in `Section 5.2.1`. The smaller the value of y_0, the closer the 5 data points become to being colinear.

The purpose of this section is to describe some of the observations we have made regarding the poles of the RBF interpolants for cases when the data points are scattered. The Contour-Padé algorithm was again paramount to these observations. Presently, no other tool exists for this kind of study.

The example that we consider and the resulting pole locations are given in `Figure 5.7 a-d`. Many of our observations regarding the pole locations came from the experiments discussed in Section 3.3.2 of the last chapter.

All the results in this section were contributed by GW.

6 RBF approximations with a constant term included

Throughout the paper we have only considered the basic RBF method. For completeness, this section was included to also briefly comment about how our results carry over to the augmented RBF method (when the augmentation is a constant term).

The closed form expression for the cardinal RBF interpolant of `(6.1)` is given in `Theorem 6.1`. We see this expression is nicely related to the expression for the basic RBF method given in `(2.3)`. This theorem was discovered by BF.

We have found that the same examples that led to convergence/divergence with the basic RBF method also typically lead to convergence/divergence with the augmented RBF method. One very interesting exception to this observation is the GA RBF. We have found that the augmented GA interpolants diverge as $\varepsilon \to 0$, for all the cases that lead to divergence with the standard RBFs (i.e MQ and IQ). This result was contributed by GW and BF.

7 Conclusions

In addition, to the concluding remarks in the paper, we comment here that this is the first study to link the issue of polynomial unisolvency to the existence and uniqueness of RBF interpolants as $\varepsilon \to 0$. Additionally, we have shown the importance of looking at the interpolants in the complex ε-plane to explain their behavior for strictly real values of ε.

Chapter 5

Future Directions

In the three previous chapters we have shown that

- Boundary errors for RBF interpolants can be controlled effectively,
- Stable computations of RBF interpolants are possible as the basis functions become increasingly flat, and
- In the limit of increasingly flat basis functions, the RBF interpolants (usually) converge to polynomial interpolants.

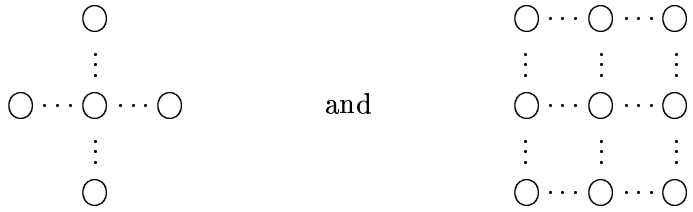
Based on these results, we see a number of possible extensions and generalizations of classical numerical methods based on polynomials, to methods based on RBFs. Two prominent examples of such methods are finite difference (FD) methods for the numerical solution of partial differential equations (PDEs), and linear multistep methods (LMS) for the numerical solution of ordinary differential equations (ODEs). In the two sections that follow, we briefly discuss the possible extensions and generalizations of these methods, and include some encouraging preliminary results.

5.1 Scattered node FD formulas generated by RBFs

The principle behind generating 1-D FD formulas is to fit a polynomial interpolant through a set of nodes and then differentiate it analytically in order to obtain a set of weights that can be used for approximating the derivatives of some function [25]. Such 1-D formulas are typically combined to create FD formulas in higher dimensions. This technique requires that the nodes fall on some kind of regular grid, which severely limits the application of FD formulas for numerically solving PDEs in complex domains. It would be highly beneficial to instead allow the nodes to be placed freely, so that a good discretization of the physical domain could be obtained. This technique has been attempted with some degree of success [1, 61]. However, it has not become widely used, partly because it leads to several ambiguities about how to generate the FD formulas. For example, what mixed terms should be included in the multivariate polynomial interpolant to the nodes, and how should a set of nodes that leads to a singular polynomial interpolation problem be handled (recall from Section 1.1 that only 1-D polynomial interpolation is always well-defined)? These ambiguities can be easily resolved if an RBF interpolant is used to generate the weights of the FD formula instead of a multivariate polynomial interpolant. For example, if the GA RBF is used then no decisions need to be made about what mixed terms to include, and, regardless of the location of the nodes

and the dimension, the interpolation problem can never become singular. Moreover, in the $\varepsilon \to 0$ limit, polynomial-type FD formulas will be obtained.

The initial idea of using RBF generated FD formulas (RBF-FD formulas) was first proposed to our group by BF. He initially experimented with RBF-FD formulas for the 2-D Laplacian $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. This work was primarily focused on exploring the behavior of the formulas for the standard 5 and 9 point stencils, i.e.



where the distance between the node points is h. In the $\varepsilon \to 0$ limit, BF found that the RBF-FD formulas (based on standard RBFs) for the 5-point stencil all converged to the classical second-order FD formula for the 2-D Laplacian. For the 9-point stencil, BF found that the formulas for the different RBFs converged to different limits as $\varepsilon \to 0$. However, all the formulas were second order accurate.

Since BF's initial investigation, GW has taken the lead on exploring the viability of using RBF-FD formulas for numerically solving PDEs on unstructured grids (i.e. grids with spatially scattered nodes). The Contour-Padé algorithm described in Appendix B (Chapter 3) has played a key role in this investigation since it is the only available tool for the stable computation of RBF interpolants for all values of ε. The primary focus of this investigation thus far has been on numerically solving the 2-D diffusion equation on the unit disk. Some preliminary results suggest that the RBF-FD method is not only accurate, but also does not suffer from any severe time stepping constraints for explicit methods (e.g. fourth-order Runge-Kutta) even if some grid points happen to be very close to each other. This last observation is extremely encouraging since, for equispaced grids with spacing h, strict time stepping conditions usually arise for small h (i.e. $\frac{\Delta t}{h^2} < const.$). In Section 5.1.1, we describe an example using the RBF-FD method for numerically solving the diffusion equation in the unit disk and numerically explore how the solution is affected by ε. Moreover, we compare the RBF-FD solution to the solution based on the standard second-order finite difference method. Following the example, we make a number of remarks in Section 5.1.2 about future ideas to explore with the RBF-FD method.

It should also be mentioned that the RBF-FD method has been explored, entirely independent from our group, by Shu, Ding, and Yeo [63]. They have applied the method for numerically simulating natural convection on unstructured grids, and have found that the technique works very well. In their study, they note that the stencils based on small values of ε tend to produce particulary accurate numerical solutions to the PDEs. However, due to the ill-conditioning problems of computing the interpolants for small ε, they were severely limited in the range of ε that could be considered. With the Contour-Padé algorithm, we can now explore the RBF-FD method for the full range of ε.

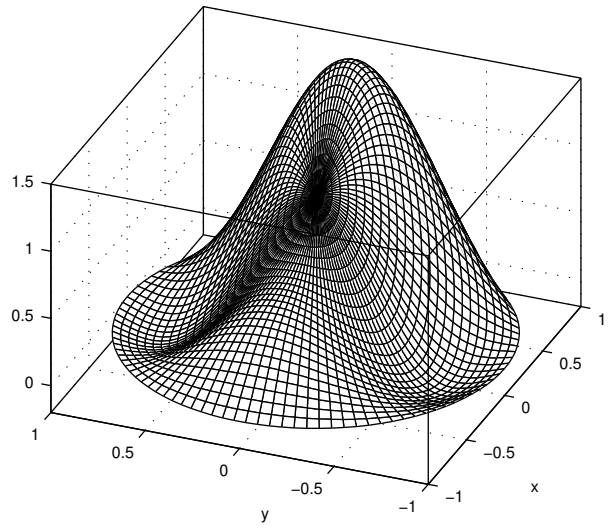


Figure 5.1: Initial condition of (5.1).

5.1.1 An example using the RBF-FD method for numerically solving PDEs

In this section, we explore the RBF-FD method for numerically solving the 2-D diffusion equation on the unit disk with an unstructured grid. The goal is to see how the solution is affected by ε and how the solution compares to the standard second order finite difference (FD2) method (which requries a polar grid). All the results in the section were contributed by GW.

The model problem used for the investigation is given as follows:

$$u_t = \kappa \nabla^2 u = \kappa \left(u_{rr} + \frac{1}{r} u_r + \frac{1}{r^2} u_{\theta\theta} \right) \qquad t > 0,\ 0 \leq r < 1,\ 0 \leq \theta \leq 2\pi \tag{5.1}$$

$$u(0, r, \theta) = J_0(r\eta_0) + J_1(r\eta_1)\cos(\theta) + J_2(r\eta_2)\cos(2\theta) \qquad 0 \leq r < 1,\ 0 \leq \theta \leq 2\pi$$

$$u(t, 1, \theta) = 0 \qquad t > 0,\ 0 \leq \theta \leq 2\pi$$

where η_0, η_1, and η_2 are the first zeros of the J-Bessel functions J_0, J_1, and J_2, respectively. The exact solution to this initial-value problem is

$$u(t, r, \theta) = \exp(-\kappa\eta_0^2 t) J_0(r\eta_0) + \exp(-\kappa\eta_1^2 t) J_1(r\eta_1)\cos(\theta) + \exp(-\kappa\eta_2^2 t) J_2(r\eta_2)\cos(2\theta) \tag{5.2}$$

For the experiment, we set the diffusion coefficient to unity, i.e. $\kappa = 1$. The initial condition of (5.1) is shown in Figure 5.1.

Figure 5.2 (a) shows the structured grid used for discretizing the unit disk for the FD2 method, and Figure 5.2 (b) shows the unstructured grid used for the RBF-FD method. Both grids contain 200 data points, and approximately the same number of

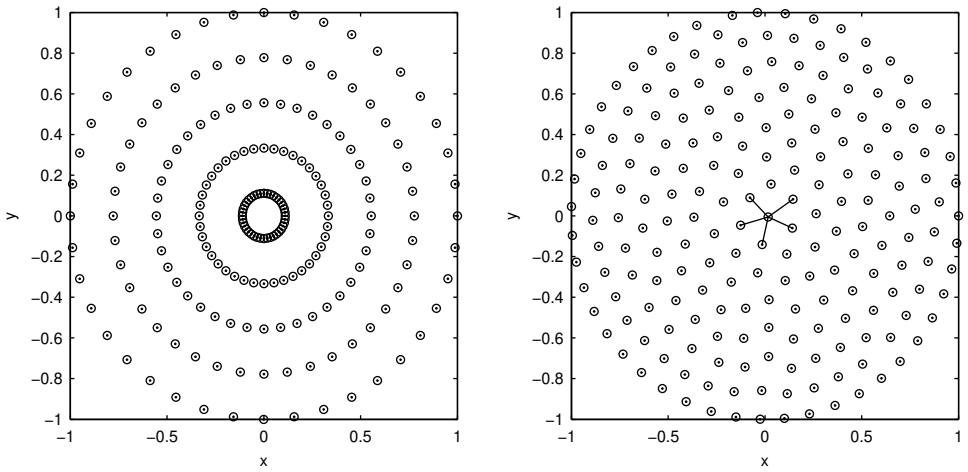


Figure 5.2: (a) Structured gird for the FD2 method; (b) Unstructured grid for the RBF-FD method. The connected line segments on the RBF-FD grid indicate a typical 6-point stencil used to generate the RBF-FD formulas.

boundary and interior points. The FD2 grid contains 40 boundary and 160 interior points, and the RBF-FD grid contains 46 boundary and 154 interior points.

We restrict our attention to RBF-FD formulas based on 6 points. Since we use the unstructured grid in 5.2 (b), the RBF-FD formula for each point in the grid will be different. To generate the formulas, we iterate through each of the interior points $\underline{x}_j = (x_j, y_j)$, $j = 1, \ldots, n_i$, where we want to approximate the Laplacian (recall that the model problem has Dirichlet boundary conditions). For each point $\underline{x}_j = \underline{x}_0^{(j)}$, we select its 5 nearest neighboring points $\underline{x}_1^{(j)}, \ldots, \underline{x}_5^{(j)}$, and compute the 6 cardinal RBF interpolants based on the points $\underline{x}_0^{(j)}, \ldots, \underline{x}_5^{(j)}$, i.e. we compute the interpolants $\psi_0^{(j)}(\underline{x}), \ldots, \psi_5^{(j)}(\underline{x})$, where

$$\psi_k^{(j)}(\underline{x}_\ell^{(j)}) = \begin{cases} 1 & \text{if } \underline{x}_\ell^{(j)} = \underline{x}_k^{(j)} \quad (0 \le \ell \le 5) \\ 0 & \text{otherwise} \end{cases} \quad , \text{for } k = 0, \ldots, 5.$$

Based on these cardinal interpolants, the RBF interpolant to the 6 points $\underline{x}_0^{(j)}, \ldots, \underline{x}_5^{(j)}$ is given by

$$s^{(j)}(\underline{x}) = \sum_{k=0}^{5} \psi_k^{(j)}(\underline{x}) f_k^{(j)} \ , \tag{5.3}$$

where $f(\underline{x}_k^{(j)}) = f_k^{(j)}$ are some given function values. The desired weights for the RBF-FD formula at the point $\underline{x}_j = \underline{x}_0^{(j)}$ are thus

$$c_k^{(j)} = \nabla^2 \psi_k^{(j)}(\underline{x})\Big|_{\underline{x}=\underline{x}_0^{(j)}} \ , k = 0, \ldots, 5. \tag{5.4}$$

We can thefore approximate the Laplacian of $f(\underline{x})$ at the point $\underline{x}_j$ with the RBF-FD formula

$$\nabla^2 f(\underline{x}_j) \approx \sum_{k=0}^{5} c_k^{(j)} f(\underline{x}_k^{(j)}) \ , k = 0, \ldots, 5. \tag{5.5}$$

Figure 5.2 (b) shows a typical set of 6 points for generating the RBF-FD formulas used in the experiments. In the context of the above discussion, the point in the center of this stencil is $\underline{x}_j = \underline{x}_0^{(j)}$. While the above discussion is limited to 6 point stencils, the generalization to any size stencil should follow directly.

Every set of 6 points in Figure 5.2 (b) for generating the RBF-FD stencils satisfies the polynomial unisolvency condition. Therefore, by `Theorem 4.4` in Appendix C, each RBF interpolant (5.3) converges to the unique second degree interpolating polynomial as $\varepsilon \to 0$ (with some mild assumptions on the basis functions). Thus, in the $\varepsilon \to 0$ limit, (5.5) should be exact for all polynomials (in x and y) of degree ≤ 2. This means that we can expect the $\varepsilon = 0$ stencils to have the accuracy $O(h)$, where h is some measure of the stencil width. In contrast, the standard FD2 formulas for grid in Figure 5.2 (a) are exact for polynomials of degree ≤ 3 (in r and θ), which means they are $O(h^2)$.

For the numerical experiment, we used the GA RBF to generate the 6 point RBF-FD formulas (5.4) for approximating the Laplacian in (5.1) at every interior point of the unstructured grid shown in Figure 5.2 (b). We then combined these formulas with the standard explicit fourth-order Runge-Kutta (RK4) method for advancing the system in time. We also used this technique for computing the FD2 solution. Figure 5.3

compares the results for both methods by comparing the maximum absolute error in the solutions at various time values t. The error for the FD2 method, RBF-FD method with $\varepsilon = 0$, and the RBF-FD method with the "optimal" ε are marked on each of the plots. We can see from the figure that the error in the RBF-FD solution for $\varepsilon = 0$ (i.e. the polynomial-type FD solution) is smaller than the FD2 solution (also polynomial based) for each time value. Moreover, the error in the RBF-FD solution initially decreases as ε moves away from 0 and reaches a minimum that is significantly less than the error for the FD2 solution (especially for $t \geq 0.5$). Another interesting feature in the solution is that the optimal value of ε appears to remain fixed for $t \geq 0.5$.

For the RBF-FD method, we found that to maintain stability as the system was advanced in time, a time step $\Delta t \leq 1.0 \cdot 10^{-3}$ was required for the RK4 method (and for all $0 \leq \varepsilon \leq 2.5$). In contrast, we found that for the FD2 method $\Delta t \leq 1.0 \cdot 10^{-4}$ was required.

The results from the above experiment are very encouraging. First, they show that the RBF-FD method can be more accurate than the FD2 method for a wide range of ε, including $\varepsilon = 0$. This is partly due to the great flexibility that the RBF-FD method has in the placement of the nodes. This flexibility allows one to use a fairly uniform distribution of points over the domain unlike the standard FD2 method, which requires a certain amount of clustering of the points (cf. Figure 5.2 (a) and (b)). Second, the results show that, apart from the initial computational cost of computing the RBF-FD formulas, the computational cost of the RBF-FD method can be less than the FD2 method (when using an explicit time-stepping scheme).

5.1.2 Future ideas to consider

The preliminary results of the previous section and the results of Shu *et. al.* [63] strongly suggest that the RBF-FD method deserves further consideration. Below we list some future ideas that should be investigated.

- The rate at which the error in the RBF-FD method decreases as the unstructured grid is refined needs to be understood. For the $\varepsilon = 0$ solution, the error should decrease at a rate related to the degree of the polynomial that the formulas are exact for. However, for $\varepsilon > 0$, it is not obvious how the error might decrease.

- The effect of increasing the number of points in the RBF-FD stencils needs to be explored. Two issues that need to be considered are how the size of the stencils affect the accuracy and stability of the RBF-FD method.

- For evolutionary PDEs, the time stepping restrictions for both explicit and implicit methods needs to be explored both as a function of the spacing of the points in the unstructured grid and as a function of ε.

- In the experiments from the previous section, the RBF-FD solution was computed using the same value of ε for all the RBF-FD formulas. It may be more beneficial, however, to vary the value of ε used in formulas.

- The real advantage of the RBF-FD method appears to be its ability to effectively handle irregular geometries, while the real advantage of the standard FD method

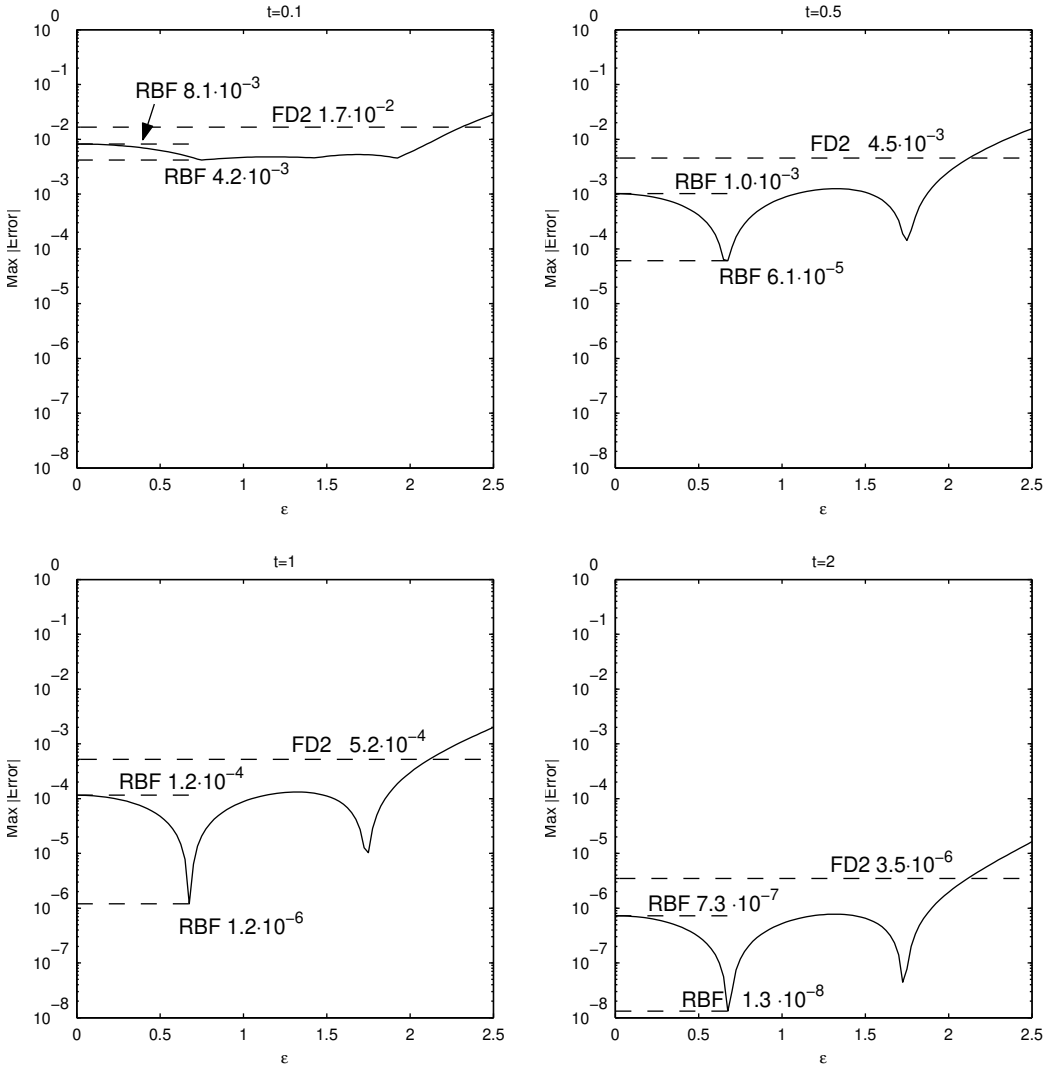


Figure 5.3: Comparison of the standard FD2 and RBF-FD methods for solving (5.1) at various time values t. The (structured) grid used for the FD2 solution is shown in Figure 5.2 (a) and (unstructured) grid for the RBF-FD solution is shown in Figure 5.2 (b). For both methods, the explicit fourth order Runge-Kutta method (RK4) was used to advance the systems in time.

is its ability to effectively handle simple geometries. This suggests that a hybrid-type approach, combining both methods where they are the most effective, could be highly beneficial.

- The viability of the method for numerically solving nondissipative PDEs (e.g. Maxwell's equations in lossless media) should also be explored.

5.2 RBF-based linear multistep methods for solving ODEs

Linear multistep methods (LMS) form an important class of ODE solvers. The basic idea behind these methods is to approximate the solution of an initial-value ODE system

$$y' = f(t, y) \ , \quad t \geq t_0 \ , \quad y(t_0) = y_0 \ , \tag{5.6}$$

at $t = t_{n+s}$ ($n \geq 0$ and $s \geq 1$) by a linear combination of past (and present) approximations of y and f, i.e.

$$y_{n+s} = \sum_{m=0}^{s-1} \alpha_m y_{n+m} + \sum_{m=0}^{s} \beta_m f_{n+m} \ , \tag{5.7}$$

where $y_{n+m} \approx y\left(t_{n+m}\right)$, $f_{n+m} = f\left(t_{n+m}, y_{n+m}\right)$, and α_m and β_m are constants. The technique for choosing the values of α_m and β_m defines the LMS method. Three popular classes of methods are Adams-Bashforth (AB), Adams-Moulton (AM), and backward differentiation formulae (BDF). A common theme between these three methods is that the constants are computed from a polynomial interpolant. This polynomial interpolant is based on past (and for AM and BDF also present) approximations of $f(t, y)$. In all of these methods, we utilize a polynomial interpolant right at the end of the interval on which it is defined. It is well known that polynomial interpolants on equispaced grids behave very badly near the ends of intervals (i.e. they feature the Runge phenomenon). Therefore, one may wish to see if the LMS methods can be improved by replacing the polynomial interpolants used in their derivation by some other types of interpolants. RBF interpolants seem like a natural choice for this replacement, since

- The interpolants can be spectrally accurate,
- Techniques have been developed for improving the interpolants behavior near the ends of intervals (as discussed in the paper from Appendix A),
- The interpolants feature a free parameter ε that can be used for improving their accuracy, and
- The standard (i.e. polynomial based) LMS methods result in the $\varepsilon = 0$ limit.

The novel idea of using RBFs to generate LMS methods was developed entirely by GW. It does not appear to have received any previous consideration in the RBF literature.

In Section 5.2.1, we briefly review the ideas behind the AB, AM, and BDF methods, and derive the RBF-based form of these methods. Following this, we describe an example in Section 5.2.2 where RBF-based AB and AM methods are used for numerically solving an initial-value ODE. The focus of the example is on how the solution is affected by ε. In the final Section 5.2.3, we make several remarks about future ideas to explore with RBF-based LMS methods.

5.2.1 Derivation of the RBF-based LMS methods

To derive the RBF-based form of the AB, AM, and BDF methods, we make use of the cardinal RBF interpolants. We denote the n cardinal RBF interpolants to the data points $\underline{x}_1, \ldots, \underline{x}_n$ by $\psi_1(\underline{x}), \ldots, \psi_n(\underline{x})$, i.e.

$$\psi_j(\underline{x}) = \sum_{i=1}^{n} \lambda_i^{(j)} \phi(\|\underline{x} - \underline{x}_i\|) \quad \text{for } j = 1, \ldots, n, \tag{5.8}$$

where $\lambda_i^{(j)}$ are determined by requiring

$$\psi_j(\underline{x}_\ell) = \begin{cases} 1 & \text{if } \underline{x}_\ell = \underline{x}_j \quad (1 \le \ell \le n) \\ 0 & \text{otherwise} \end{cases} .$$

A closed for expression for these cardinal RBF interpolants is given by `(2.3)` in Appendix C.

5.2.1.1 Adams-Bashforth with RBFs

The general solution of (5.6) at $t = t_{n+s}$ can be written as follows

$$y(t_{n+s}) = y(t_{n+s-1}) + \int_{t_{n+s-1}}^{t_{n+s}} y'(\tau)\, d\tau = y(t_{n+s-1}) + \int_{t_{n+s-1}}^{t_{n+s}} f(\tau, y(\tau))\, d\tau \,. \tag{5.9}$$

The AB method exploits this general solution by extrapolating the value of $f(\tau, y(\tau))$ in the interval $[t_{n+s-1}, t_{n+s}]$ based on the s previous approximations of $y(t)$ at $t = t_n, t_{n+1}, \ldots, t_{n+s-1}$. The standard AB method can be derived by first computing the Lagrange interpolating polynomial to the data points $\{t_{n+m}\}_{m=0}^{s-1}$ and corresponding data values $\{f_{n+m}\}_{m=0}^{s-1}$, and then integrating this the polynomial over the required interval to obtain y_{n+s} (the approximation of $y(t_{n+s})$). We are instead interested in how the method behaves when we use an RBF interpolant instead of a polynomial interpolant.

The RBF interpolant to the data points $\{t_{n+m}\}_{m=0}^{s-1}$ and corresponding data values $\{f_{n+m}\}_{m=0}^{s-1}$ is given by

$$q(t) = \sum_{m=0}^{s-1} \psi_{m+1}(t) f_{n+m} \,, \tag{5.10}$$

where each $\psi_{m+1}(t)$ is given by (5.8). Plugging (5.10) into the integrand of (5.9) we obtain the following approximation for $y(t_{n+s})$:

$$y_{n+s} = y_{n+s-1} + \int_{t_{n+s-1}}^{t_{n+s}} q(\tau)\, d\tau \,.$$

Writing this in the form of (5.7) we have

$$y_{n+s} = y_{n+s-1} + \sum_{m=0}^{s-1} \beta_m f_{n+m} \,, \tag{5.11}$$

where

$$\beta_m = \int_{t_{n+s-1}}^{t_{n+s}} \psi_{m+1}(\tau)\, d\tau \quad \text{for } m = 0, \ldots, s-1.$$

Assuming a uniform time step h, we can write the general expression for β_m so it is independent of n:

$$\beta_m = \int_0^h \psi_{m+1}(\tau)\, d\tau \quad \text{for } m = 0, \ldots, s-1. \tag{5.12}$$

Note that while β_m for the standard polynomial-based AB method is also independent of h, this does not hold for the RBF-based AB method.

We refer to (5.11) as the s-step RBF-AB method, or RBF-ABs ($s \geq 1$). We can see from this equation that y_{n+s} does not depend on the value of f_{n+s}. Hence the RBF-ABs method is an explicit method. We next derive the RBF-based AM method which does have this dependency and is therefore an implicit method.

5.2.1.2 Adams-Moulton with RBFs

The basic idea of the AM method is the same as the AB method, i.e. approximate the value of the integrand in (5.9) over the interval $[t_{n+s-1}, t_{n+s}]$. However, instead of using the s past values $y_n, y_{n+1}, \ldots, y_{n+s-1}$ to extrapolate an approximate value of the integrand, we use the $s+1$ values $y_n, y_{n+1}, \ldots, y_{n+s-1}, y_{n+s}$ to interpolate an approximate value. We now proceed with these $s+1$ values in the same manner as we did for the RBF-ABs method (paying no attention to the fact that we don't have a value for y_{n+s}). This yields the following result, assuming a uniform time step h:

$$y_{n+s} = y_{n+s-1} + \sum_{m=0}^{s} \beta_m f_{n+m}\ , \tag{5.13}$$

where

$$\beta_m = \int_0^h \psi_{m+1}(\tau)\, d\tau \quad \text{for } m = 0, \ldots, s. \tag{5.14}$$

We refer to (5.13) as the s-step RBF-AM method, or RBF-AMs ($s \geq 1$). Since this method depends implicitly on y_{n+s} (from the term f_{n+s}), we either need to solve a linear system at each time step or we need to compute an approximation of this value. The latter can be done with an RBF-ABs method (5.11). This is typically referred to as the AB/AM *predictor-corrector* method.

5.2.1.3 BDFs with RBFs

While the AB and AM methods exploit (5.9), the BDF method exploits the ODE system (5.6) itself. The basic idea behind the method is to approximate the derivative $y'(t_{n+s})$ based on the polynomial interpolant to past values of y. However, we are instead interested in using an RBF interpolant. The procedure for doing this is described below.

First, we form the RBF interpolant to the $s+1$ data points $\{t_{n+m}\}_{m=0}^{s}$ and corresponding data values $\{y_{n+m}\}_{m=0}^{s}$

$$q(t) = \sum_{m=0}^{s} \psi_{m+1}(t) y_{n+m}\ , \tag{5.15}$$

where each $\psi_{m+1}(t)$ is given by (5.8). We next compute the derivative of (5.15) and evaluate it at $t = t_{n+s}$. Using (5.6), we have that

$$\left.\frac{dq}{dt}\right|_{t=t_{n+s}} = y'_{n+s} \approx y'(t_{n+s}) = f(t_{n+s}, y(t_{n+s})),$$

$$\Longrightarrow y'_{n+s} = f_{n+s}$$

Hence, the RBF-based BDF method can be defined as follows:

$$f_{n+s} = \sum_{m=0}^{s} \alpha_m y_{n+m}$$

or, in the form of (5.7),

$$y_{n+s} = \beta_s f_{n+s} - \frac{1}{\alpha_s} \sum_{m=0}^{s-1} \alpha_m y_{n+m} \,, \tag{5.16}$$

where $\beta_s = \frac{1}{\alpha_s}$ and

$$\alpha_m = \left.\frac{d}{dt}\psi_{m+1}(t)\right|_{t=t_{n+s}} \quad \text{for } m = 0, \ldots, s. \tag{5.17}$$

If we again assume a uniform time step h, then (5.17) is independent of n.

We refer to (5.16) as the s-step RBF-BDF method, or RBF-BDFs ($s \geq 1$). Note that this method also depends implicitly on y_{n+s}.

5.2.2 An example using RBF-based LMS methods for numerically solving ODEs

In this section, we present some preliminary results for numerically solving initial-value ODEs with RBF-based LMS methods. The goal is to demonstrate the viability of the methods and to see how the methods are affected by ε. To keep the discussion brief, we restrict our attention to the RBF-AB4 method and the RBF-AB4/AM4 predictor-corrector method since similar results were obtained for larger and smaller s-step RBF methods.

The example we consider is the *Curtiss-Hirschfelder* equation

$$y' = -10(y - \cos(t)) \,, \quad y(0) = 1 \,, \quad t \geq 0 \,. \tag{5.18}$$

The analytic solution of this initial-value ODE is

$$y = \frac{100}{101}\cos(t) + \frac{10}{101}\sin(t) + \frac{1}{101}\exp(-10t) \,, \quad t \geq 0 \,. \tag{5.19}$$

We see that the solution approaches a periodic curve at an exponential rate. Figure 5.4 shows a plot of the solution for $0 \leq t \leq 7$.

For the numerical experiment, we used the GA RBF to generate both sets of coefficients for the RBF-AB4 scheme (5.11), and the RBF-AM4 scheme (5.13) (which we then combined with RBF-AB4 to form the predictor-corrector scheme). Since the

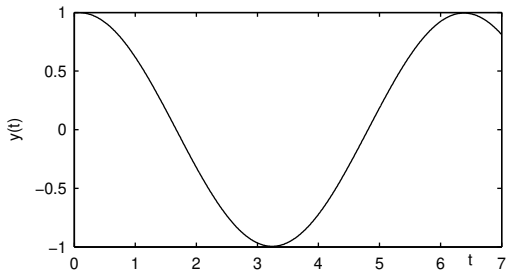


Figure 5.4: Plot of the actual solution (5.19) of (5.18).

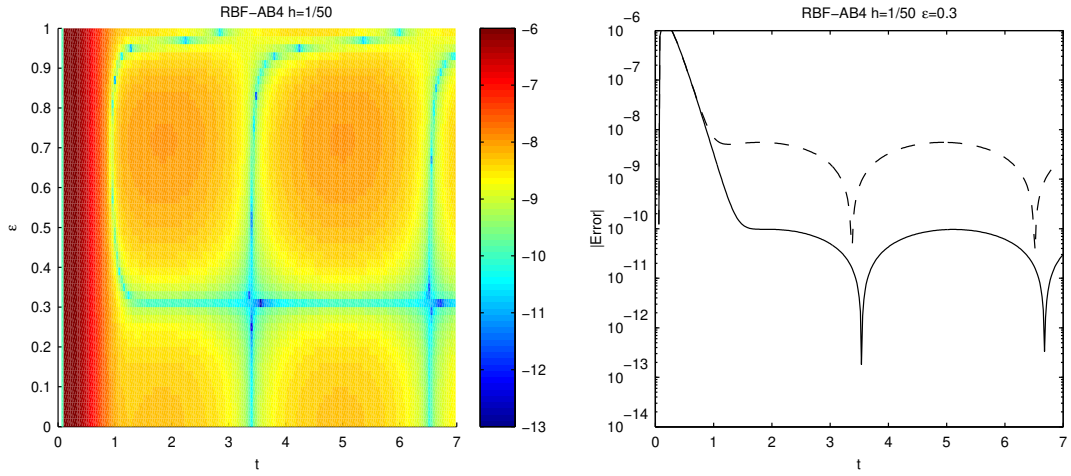
model problem (5.18) is mildly stiff, we were forced to use a time step of $h = 1/50$ to obtain a stable solution for all t and for all ε. This time step is also what would be needed for the standard AB4 and AB4/AM4 methods. The results of the experiment are shown in Figure 5.5 (a) and (b). The plots in the left column of the figure show how the error in the methods vary for $t \geq 0$ and $0 \leq \varepsilon \leq 1$. We can see from these plots that for $t < 1$, the solution behaves similarly for all ε. However, as t increases beyond this value, the solutions begin to differ with ε. Both plots shows that there are certain non-zero values of ε that consistently provide a better approximation to the solution of (5.18) (indicated by the blue shaded line moving from left to right). Since the solution of the ODE is (almost) periodic, we should expect (and indeed see) that the numerical solution becomes very accurate for certain values of t. The $\varepsilon = 0$ solution corresponds to the standard AB4 and AB4/AM4 methods. While it may not be clear from the plots, the error in both of the methods initially decreases as ε moves away from 0 and reaches a minimum (indicated by the blue shading). This indicates that there appears to be a rather wide range of ε where the error in the RBF-based methods is smaller than the error for the standard methods.

The plots in the right column of Figure 5.5 (a) and (b) show the magnitude of the error in the numerical solution corresponding to the optimal ε (solid line) and in the numerical solution corresponding to $\varepsilon = 0$ (dashed line). Here the optimal ε is defined as the value that provides the most accurate solution for all t. This optimal value is indicated in the title of each plot. It is interesting that this value is the same for both methods. We can see from the plots that the error in both solutions is initially approximately the same. However, for $t \geq 1$, the error in the optimal ε solution is in general about two order of magnitude smaller. Based on some additional experiments with different ODEs with periodic solutions, it appears that the optimal ε is definitely related to the periodicity of the solution.

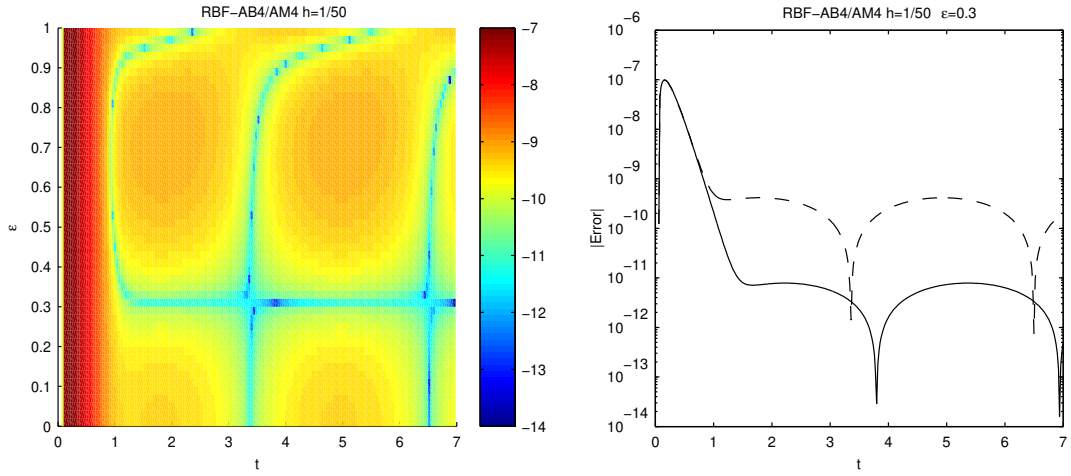
5.2.3 Future ideas to consider

The preliminary results from the numerical examples of the previous section indicate that RBF-based LMS methods deserve to be explored further. Below we list some future ideas that should be investigated.

- The classical theory of LMS methods (e.g. consistency, accuracy, and stability)



(a)



(b)

Figure 5.5: Error in the (a) RBF-AB4 and (b) RBF-AB4/AM4 solutions of (5.18) for $0 \leq t \leq 7$. The figures in the left column display the magnitude of the error (in $\log_{10}$ scale) for $0 \leq \varepsilon \leq 1$. The color bar just to the right of these figures correlates the error to the different colors (note the different scales for (a) and (b)). The figures in the right column display the magnitude of the error in the optimal ε solution (solid line) with the error in the standard (polynomial-based) AB4 and AB4/AM4 solution (dashed line). The step size k and the optimal ε are displayed in the title of each plot. For both (a) and (b) the GA RBF was used.

should be investigated to see how it can be extended to include these new RBF-based LMS methods.

- For initial-value ODEs with periodic solutions, the RBF-based LMS methods appear to be significantly more accurate for a small range of ε. Techniques for either analytically or numerically finding this optimal ε should be investigated.

- The viability of RBF-based LMS methods for numerically solving several different types of ODEs (not just ODEs with periodic solutions) should be investigated.

- The stability domains of the different RBF-based LMS methods should be investigated to see how they are affected by ε. It could be that for certain non-zero values of ε the stability domains exhibit some desirable properties.

- The boundary improvement techniques discussed in Appendix A (e.g. the SNaK technique), should be implemented in the various RBF-based LMS methods to see how the error in the method is affected.

- Simple formulas for generating the coefficients in standard LMS methods (5.7) typically exist. It would be highly beneficial if similar formulas could also be found for generating the coefficients of RBF-based LMS methods. Presently, the coefficients are generated numerically using the Contour-Padé algorithm discussed in Appendix B.

- In Section 5.1 we introduced the concept of using RBF-FD formulas for approximating the spatial derivatives of a PDE. It seems natural to investigate how these formulas can be combined with RBF-based LMS methods for numerically solving time-dependent PDEs in a method-of-lines approach. Since both techniques are based on RBF interpolants, it could be that the solutions are much more accurate than those based on the standard (polynomial-based) techniques.

Bibliography

[1] T. Adolph and W. Schönauer, The generation of high quaility difference and error formulae for arbitrary order on 3-D unstructured grids, ZAMM special issue for GAMM-Tagung, (2000).

[2] K. Balakrishnan, R. Jureshkumar, and P. A. Ramachandran, An operator splitting-radial basis function method for the solution of transient nonlinear Poisson problems, Comput. Math. Appl., 43 (2002), pp. 289–304.

[3] I. Barrodale, D. Skea, M. Berkley, R. Kuwahara, and R. Poeckert, Warping digital images using thin-plate splines, Pattern Recognition, 26 (1993), pp. 375–376.

[4] B. J. C. Baxter, On the asymptotic behaviour of the span of translates of the multiquadric $\phi(r) = (r^2 + c^2)^{1/2}$ as $c \to \infty$, Comput. Math. Appl., 24 (1994), pp. 1–6.

[5] R. K. Beatson, J. B. Cherrie, and C. T. Mouat, Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration, Adv. Comput. Math., 11 (1999), pp. 253–270.

[6] R. K. Beatson, W. A. Light, and S. Billings, Fast solution of the radial basis function interpolation equations: Domain decomposition methods, SIAM J. Sci. Comput., 22 (2000), pp. 1717–1740.

[7] R. K. Beatson and M. J. D. Powell, An iterative method for thin plate spline interpolation that employs approximations to Lagrange functions, in Numerical Analysis 1993, D. F. Griffiths and G. A. Watson, eds., Longman Scientific and Technical, Harlow, 1994.

[8] A. Bejancu, Local accuracy for radial basis function interpolatio on finite uniform grids, DAMTP Report NA19, University of Cambridge, 1997.

[9] C. M. Bender and S. A. Orszag, Advanced Mathematical Methods for Scientists and Engineers, McGraw-Hill, New York, 1978.

[10] M. D. Buhmann and N. Dyn, Spectral convergence of multiquadric interpolation, in Proceedings of the Edinburgh Mathematical Society, vol. 36, Edinburgh, 1993, pp. 319–333.

[11] R. E. Carlson and T. A. Foley, The parameter R^2 in multiquadric interpolation, Comput. Math. Appl., 21 (1991), pp. 29–42.

[12] J. C. Carr, W. R. Fright, and R. K. Beatson, Surface interpolation with radial basis functions for medial imaging, IEEE Trans. Medial Imgaing, 16 (1997), pp. 96–107.

[13] E. W. Cheney and W. A. Light, A Course in Approximation Theory, Brooks/Cole, New York, 2000.

[14] T. A. Driscoll and B. Fornberg, Interpolation in the limit of increasingly flat radial basis functions, Comput. Math. Appl., 43 (2002), pp. 413–422.

[15] M. R. Dubal, Domain decomposition and local refinement for multiquadric approximations. I. Second order equations in one-dimension, J. Appl. Sci. Comput., 1 (1994), pp. 146–171.

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

[17] N. Dyn, D. Levin, and S. Rippa, Numerical procedures for global surface fitting of scattered data by radial functions, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 639–659.

[18] G. E. Fasshauer, Solving partial differential equations by collocation with radial basis functions, in Surface fitting and multiresolution method, Vol.2 of the proceedings of the 3rd International Conference on Curves and Surfaces, A. LeMehaute, C. Rabut, and L. L. Schumaker, eds., Chamonix–Mont-Blac, 1997, Vanderbilt University Press, Nashville Tennessee.

[19] ——, Solving partial differential equations with radial basis functions: multilevel methods and smoothing, Adv. Comput. Math., (1999), pp. 139–159.

[20] A. C. Faul and M. J. D. Powell, Krylov subspace methods for radial basis function interpolation, in International Conference on Numerical Analysis, Dundee, August 1999.

[21] A. I. Fedoseyeve, M. J. Friedman, and E. J. Kansa, Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary, Comput. Math. Appl., 43 (2001), pp. 439–455.

[22] J. Flusser, An adaptive method for image registration, Patter Recognition, 25 (1992), pp. 45–54.

[23] T. A. Foley, Near optimal parameter selection for multiquadric interpolation, J. Appl. Sci. Comput., 1 (1994), pp. 54–69.

[24] B. Fornberg, Numerical differentiation of analytic functions, ACM Tans. Math. Sofw., 7 (1981), pp. 512–526.

[25] ——, A Practical Guide to Pseudospectral Methods, Cambridge University Press, Cambridge, 1996.

[26] B. Fornberg, T. A. Driscoll, G. Wright, and R. Charles, Observations on the behavior of radial basis functions near boundaries, Comput. Math. Appl., 43 (2002), pp. 473–490.

[27] B. Fornberg and N. Flyer, Accuracy of radial basis function derivative approximations in 1-d, Adv. Comput. Math., (2003). Submitted.

[28] B. Fornberg and G. Wright, Stable computation of multiquadric interpolants for all values of the shape parameter, Comput. Math. Appl., (2003). Submitted.

[29] B. Fornberg, G. Wright, and E. Larsson, Some observations regarding interpolants in the limit of flat radial basis functions, Comput. Math. Appl., (2003). to appear.

[30] R. Franke, A critical comparison of some methods for interpolation of scattered data, TR NPS-53-79-003, Naval Postgraduate School, 1979.

[31] ——, Scattered data interpolation: tests of some methods, Math. Comput., 38 (1982), pp. 181–200.

[32] ——, Lecture notes on global basis function methods for scattered, in International Symposium on Surface Approximation, Govgano, Italy, 1983, University of Milano.

[33] F. Girosi, Some extensions of radial basis functions and their applications in artificial intelligence, Comput. Math. Appl., 24 (1992), pp. 61–80.

[34] M. A. Golberg and C. S. Chen, Discrete Projection Methods for Integral Equations, Computational Mechanics Publications, Southampton, 1997.

[35] A. Haar, Die minkowskische geometrie und die ann an stetige funktionen, Math. Ann., 18 (1918), pp. 294–311.

[36] R. L. Hardy, Multiquadric equations of topograpy and other irregular surfaces, J. Geophy. Res., 76 (1971), pp. 1905–1915.

[37] ——, Theory and applications of the multiquadric-biharmonic method: 20 years of discovery, Comput. Math. Appl., 19 (1990), pp. 163–208.

[38] R. L. Hardy and W. M. Göpfert, Least squares prdiction of gravity anomalies, geoidal undulations, and deflections of the vertical with multiquadric harmonic functions, Geophys. Res. Lett., 10 (1975), pp. 423–426.

[39] P. Henrici, Applied and Computational Complex Analysis, vol. 3, John Wiley and Sons, New York, 1986.

[40] Y. C. Hon, K. F. Cheung, X. Z. Mao, and E. J. Kansa, Multiquadric solution for shallow water equations, ASCE J. Hydr. Engrg., 125 (1999), pp. 524–533.

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

[42] E. J. KANSA, Multiquadrics – a scattered data approximation scheme with applications to computational fluid-dynamics – I: Surface approximations and parital derivative estimates, Comput. Math. Appl., 19 (1990), pp. 127–145.

[43] ——, Multiquadrics – a scattered data approximation scheme with applications to computational fluid-dynamics – II: Solutions to parabolic, hyperbolic and elliptic partial differential equations, Comput. Math. Appl., 19 (1990), pp. 147–161.

[44] E. J. KANSA AND Y. C. HON, Circumventing the ill-conditioning problem with multiquadric radial basis functions: Applications to elliptic partial differential equations., Comput. Math. Appl., (2001). Submitted.

[45] E. LARSSON AND B. FORNBERG, Theoretical aspects of multivariate interpolation with increasingly flat radial basis functions. To be submitted.

[46] ——, A numerical study of some radial basis function based solution methods for elliptic PDEs, Comput. Math. Appl., (2001). To appear.

[47] W. R. MADYCH, Miscellaneous error bounds for multiquadric and related interpolants, Comput. Math. Appl., 24 (1992), pp. 121–138.

[48] W. R. MADYCH AND S. A. NELSON, Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation, J. Approx. Theory, 70 (1992), pp. 94–114.

[49] R. J. Y. MCLEOD AND M. L. BAART, Geometry and Interpolation of Curves and Surfaces, Cambridge University Press, Cambridge, 1998.

[50] C. A. MICCHELLI, Interpolation of scattered data: distance matrices and conditionally positive definite functions, Constr. Approx., 2 (1986), pp. 11–22.

[51] F. J. NARCOWICH AND J. D. WARD, Norm estimates for the inverses of a general class of scattered-data radial-function interpolation matrices, J. Approx. Theory, 69 (1992), pp. 84–109.

[52] M. J. D. POWELL, Univariate multiquadric interpolation: some recent results, in Curves and Surfaces, P. J. Laurent, A. LeMehaute, and L. L. Schumaker, eds., San Diego, 1991, Academic Press.

[53] M. J. D. POWELL, The theory of radial basis function approximation in 1990, in Advances in Numerical Analysis, Vol. II: Wavelets, Subdivision Algorithms and Radial Functions, W. Light, ed., Oxford University Press, Oxford, UK, 1992, pp. 105–210.

[54] S. D. RIEMENSCHNEIDER AND N. SIVAKUMAR, Gaussian radial-basis functions: Cardinal interpolation of ℓ^p and power-growth data, Adv. Comput. Math., 11 (1999), pp. 229–251.

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

[56] ——, An algorithm for selecting a good value for the parameter c in radial basis function interpolation, Adv. Comput. Math., 11 (1999), pp. 193–210.

[57] R. Schaback, Error estimates and condition numbers for radial basis function interpolants, Adv. Comput. Math., 3 (1995), pp. 251–264.

[58] ——, Radial basis functions viewed from cubic splines, in Proceedings of Manheim Conference, Birkhauser, 1997.

[59] I. P. Schagen, Interpolation in two dimensions–a new technique, J. Inst. Math. Appl., 23 (1979), pp. 53–59.

[60] I. J. Schoenberg, Metric spaces and completely montone functions, Ann. Math., 39 (1938), pp. 811–841.

[61] W. Schönauer and T. Adolph, How we solve PDEs, J. Comput. Appl. Math., 131 (2001), pp. 473–492.

[62] M. Shin and C. Park, A radial basis function approach to pattern recognition and its applications, ETRI Journal, 22 (2000), pp. 1–10.

[63] C. Shu, H. Ding, and K. S. Yeo, Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier-Stokes equations, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 941–954.

[64] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Adv. Comput. Math., 4 (1995), pp. 389–396.

[65] ——, Gaussian interpolation revisted, in Trends in Approximation Theory, K. K. amd T. Lyche and M. Neamtu, eds., Vanderbilt University Press, Nashville, 2001, pp. 427–436.

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

[67] ——, Multivariate compactly supported positive definite radial functions, Adv. Comput. Math, 4 (1995), pp. 283–292.

[68] Z. Wu and R. Schaback, Local error estimates for radial basis function interpolation of scattered data, I.M.A. J. Numer. Anal., 13 (1993), pp. 13–27.

[69] J. Yoon, Spectral approximation orders of radial basis function interpolation on the Sobolev space, SIAM J. Math. Anal., 23 (2001), pp. 946–958.

[70] C. A. Zala and I. Barrodale, Warping aerial photographs to orthomaps using thin plate splines, Adv. Comput. Math, 11 (1999), pp. 211–227.

APPENDIX A: OBSERVATIONS ON THE BEHAVIOR OF RADIAL BASIS FUNCTION APPROXIMATIONS NEAR BOUNDARIES

BENGT FORNBERG*, TOBIN A. DRISCOLL†, GRADY WRIGHT‡, AND RICHARD CHARLES§

Abstract. RBF approximations would appear to be very attractive for approximating spatial derivatives in numerical simulations of PDEs. RBFs allow arbitrarily scattered data, generalize easily to several space dimensions, and can be spectrally accurate. However, accuracy degradations near boundaries in many cases severely limit the utility of this approach. With that as motivation, this study aims at gaining a better understanding of the properties of RBF approximations near the ends of an interval in 1-D and towards edges in 2-D.

Key words. Radial basis functions, RBF, PDEs, cubic splines.

1. Introduction. Radial basis function (RBF) approximations have proven to be effective and flexible in the numerical solution of certain PDEs with partly or fully dissipative character (e.g. [10], [11]). It would be very desirable to exploit their geometric flexibility and high accuracy also in strictly non-dissipative situations. For example, we are interested in applying RBFs in a method-of-lines (MOL) type approach for approximating Maxwell's equations in lossless media featuring an irregular interface. Figure 1.1 illustrates schematically how the RBF centers (represented by circles) could be distributed to capture the behavior at an interface. Overlapping the centers with a Cartesian grid on both sides of the interface would allow us to combine interpolation between the RBF solution around the interface with a high-order finite-difference solution on the regular grid. To keep computational costs low, this approach would use RBFs only where maximal geometric flexibility is needed. Our first obvious implementations of this method using standard time integrators led to time instabilities. So, it was decided to launch a more basic study of the key features of RBF approximations, with a focus on how they behave at boundaries.

A common feature in all RBF approximations is how relatively inaccurate they are at boundaries. For example, Figure 1.2 shows some RBF approximations to constant equispaced data in 1-D. When approximating (non-dissipative) wave-type PDEs, large boundary-induced errors of this type will contaminate the solution everywhere across the domain. Thus, it is necessary to understand how RBFs behave near boundaries and whether there is a way to improve accuracy there (as noted in [15] and [13]). This paper reports some results of this effort.

The plan for the remaining sections of this paper is as follows: Section 2 gives a brief introduction to RBFs and presents the RBFs considered in this paper. The following section introduces some very basic properties of cubic RBFs and cubic splines, and summarizes how they are related in 1-D. In Section 4, we make some comments on cubic RBF approximations vs. those based on other types of basis functions. In Section 5 we focus on the accuracies of different RBF variations at the edge of an interval, and some possible ways to increase the

[0]The first, third and fourth author may be reached at: University of Colorado, Department of Applied Mathematics, CB-526, Boulder, CO 80309.

*fornberg@colorado.edu. This work was supported by NSF grants DMS-9810751 and DMS-0073048.

†University of Delaware, Department of Mathematical Sciences, Ewing Hall, Newark, DE 19716 driscoll@math.udel.edu. This work was supported by an NSF Vigre Postdoctoral Fellowship under grant DMS-9810751.

‡Grady.Wright@colorado.edu. This work was supported by an NSF Vigre Graduate Traineeship under grant DMS-9810751.

§rmcharles@west.rayteon.com

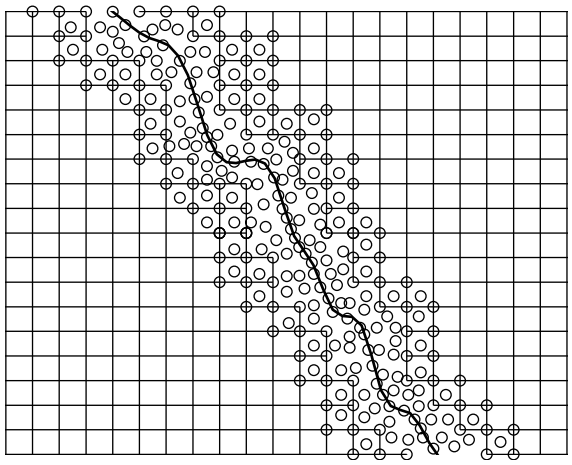


FIG. 1.1. *Example application for applying RBFs to approximate Maxwell's equations in lossless media at an irregular interface.*

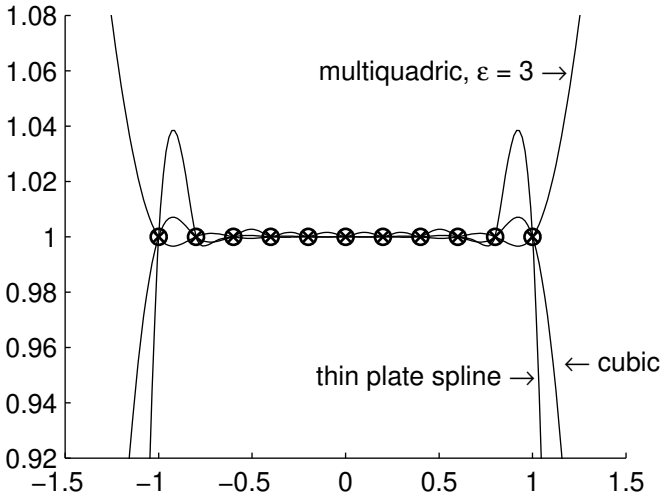


FIG. 1.2. *Some RBF approximations of constant data over [-1,1].*

accuracy. Some numerical results for 1-D RBF interpolation are also presented. In Section 6, we focus on how our observations for RBFs in 1-D carry over to 2-D, along with some numerical results. The concluding section summarizes our observations.

2. Definition of RBF approximations. The following defines a "basic" RBF approximation in any number of dimensions:

DEFINITION 2.1. *Given a function* $\phi(r)$, $r \geq 0$, *distinct centers* $x_0, x_1, \ldots, x_N$, *and data* $f_i = f(x_i)$, $i = 0, 1, \ldots, N$, *the basic interpolating RBF approximation is*

$$s(x) = \sum_{i=0}^{N} \lambda_i \phi(\|x - x_i\|_2) \tag{2.1}$$

where the λ_i are chosen so that $s(x_i) = f_i$. The variable x and points x_i can here be either scalars or vectors.

The types of basis functions we will consider in this study are

Piecewise Smooth:	
$\phi(r) = r^3$	cubic RBF
$\phi(r) = r^5$	quintic RBF
$\phi(r) = r^2 \log r$	thin plate spline (TPS) RBF
$\phi(r) = (1-r)_+^m p(r)$	Wendland functions (see [16]), where p is a polynomial
Infinitely Smooth:	
$\phi(r) = \sqrt{1+(\varepsilon r)^2}$	multiquadric (MQ) RBF
$\phi(r) = \dfrac{1}{1+(\varepsilon r)^2}$	inverse quadratic (IQ) RBF

We are in this study primarily concerned with the accuracy of RBF approximations. For some "basic" RBF approximations, and also for some RBF approximations which incorporate boundary enhancements (to be discussed later), there exist remote possibilities that singular linear systems can arise. Comments on this issue can be found in [12], and will not be discussed further here.

3. Cubic RBFs and their connection to splines.

3.1. Infinite interval. In the absence of boundaries, there is a simple relationship between cubic B-splines (denoted by $B_3(x)$; the non-zero cubic spline with the narrowest support) and cubic RBFs:

THEOREM 3.1. *On a unit-spaced 1-D grid*

$$B_3(x) = \frac{1}{12}\left[|x+2|^3 - 4\ |x+1|^3 + 6\ |x|^3 - 4\ |x-1|^3 + |x-2|^3\right]$$

This is a special case of $B_k(x)$ as follows (assuming k odd):

$$B_k(x) = \frac{1}{2\ k!}\sum_{\nu=0}^{k+1}(-1)^\nu \binom{k+1}{\nu} \left|x + \frac{k+1}{2} - \nu\right|^k \ .$$

As is conventional, the normalization factor is here chosen so that $\int_{-\frac{k+1}{2}}^{\frac{k+1}{2}} B_k(x)dx = 1$.

COROLLARY 3.2. *We can translate between a B-spline expansion $\sum a_k B_3(x+k)$ and a cubic RBF expansion $\sum \lambda_k |x+k|^3$ by means of*

$$\lambda_k = \frac{1}{12}\left[a_{k-2} - 4a_{k-1} + 6a_k - 4a_{k+1} + a_{k+2}\right] \ .$$

Figure 3.1 summarizes the key properties of some cubic spline/cubic RBF expansions on an infinite interval. In addition to Corollary 3.2, we note:

- For a cardinal function (equal to one at one node point and zero at all others), the oscillations in the cubic RBF approximation decay exponentially as we move out from the center. The amplitude ratio of successive oscillations approaches $-2+\sqrt{3} \approx -0.2679$.

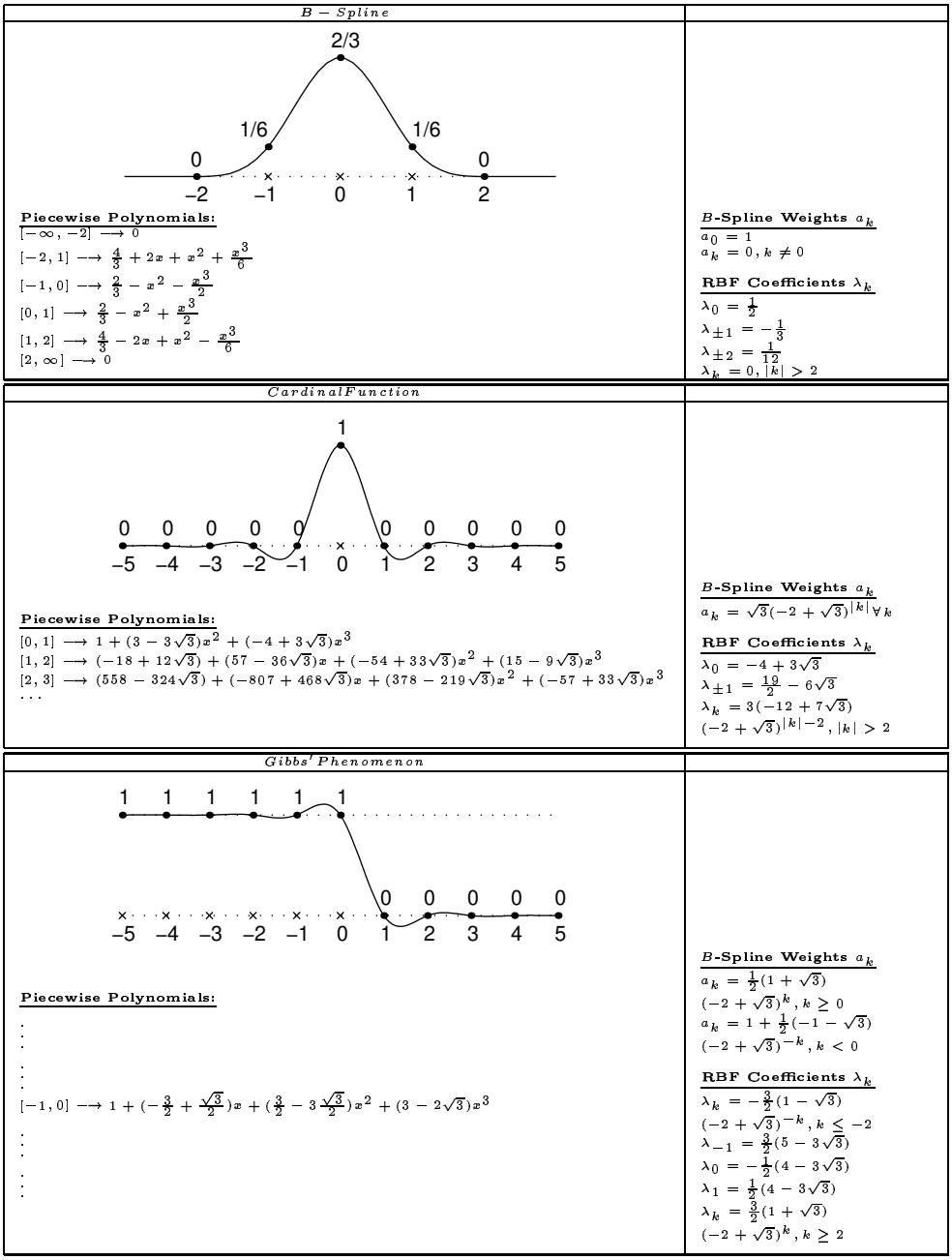


FIG. 3.1. *Examples of cubic splines / cubic RBF approximations on an infinite interval*

- For a step function, the Gibbs overshoot reaches a maximum of $\frac{1}{6}(8+2\sqrt{2}-\sqrt{3}-\sqrt{6}) \approx 1.1078$ at $x = \frac{1}{6}(-3-\sqrt{3}+\sqrt{6}) \approx -0.3804$. This can be compared to a maximum value of approximately 1.1411 in the case of trigonometric interpolation [6]. This latter value is also what arises for polynomial interpolation of increasing order (and interval length). For truncated Fourier expansions, the value is $\frac{1}{2} + \frac{1}{\pi}\int_0^\pi \frac{\sin t}{t} dt \approx 1.0895$.

3.2. Boundaries. Since RBFs behave badly at the ends of an interval (as we saw in Figure 1.2), and quite good end conditions have been devised for cubic splines (e.g. natural spline and Not-a-Knot conditions), it is of obvious interest to see how these carry over from cubic splines to cubic RBFs.

- **Equivalence of cubic spline and cubic RBF**
  When using cubic splines to approximate a given function one must choose two extra conditions to make the solution unique. It is natural to ask if these conditions can be chosen to reproduce a cubic RBF approximation. The following describes how this can be done:

  Suppose for simplicity that

  $$-1 = x_0 < x_1 < \ldots < x_N = 1. \tag{3.1}$$

  Every term in the RBF sum for $s(x)$ is a cubic which flips its sign at some point within [-1,1]. Hence, if $s(x) = ax^3 + bx^2 + cx + d$ for $x \geq 1$, we must have $s(x) = -ax^3 - bx^2 - cx - d$ for $x \leq -1$. This gives

  $$\begin{array}{lll} s(1) & = a+b+c+d & \qquad s(-1) \quad = a-b+c-d \\ s'(1) & = 3a+2b+c & \qquad s'(-1) \quad = -3a+2b-c \\ s''(1) & = 6a+2b & \qquad s''(-1) \quad = 6a-2b \end{array}$$

  Eliminating a, b, c, d leads to the two end conditions

  $$\begin{array}{ll} s''(1) & = 2s'(1) - s'(-1) - \frac{3}{2}(s(1)+s(-1)) \\ s''(-1) & = s'(1) - 2s'(-1) - \frac{3}{2}(s(1)+s(-1)), \end{array} \tag{3.2}$$

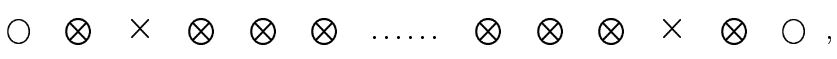
  Imposing these conditions (3.2) on a cubic spline will recreate the basic cubic RBF. These seemingly strange end conditions coupling the two sides together are the cause of the oscillations seen in the cubic RBF approximation shown in Figure 1.2.

- **Natural cubic spline**
  A natural cubic spline is obtained by choosing as the extra two conditions that the second derivative at each end be 0. The equivalent "natural" cubic RBF $s(x)$ is obtained by letting $s(x) = a+bx+\sum \lambda_i |x - x_i|^3$ and enforcing $\sum \lambda_i = 0$, $\sum \lambda_i x_i = 0$. This can be seen as follows (assuming again (3.1)):

  For $x \geq 1$ (i.e. $x \geq x_i$), $s(x) = a + bx + \sum \lambda_i (x - x_i)^3$. Therefore $s''(x) = 6\sum \lambda_i (x - x_i) = 6x \sum \lambda_i - 6 \sum \lambda_i x_i = 0$ (because of the imposed constraints). Hence $s''(1) = 0$, and similarly $s''(-1) = 0$. Within (-1,1), $s(x)$ is a piecewise cubic with jumps in the third derivative at the data locations. The function $s(x)$ satisfies all the requirements of the natural spline. Since this is well known to be unique, it must be equal to $s(x)$.

- **Not-a-Knot cubic spline**
  If the two extra condtions for the cubic spline are chosen so that there is no jump in the third derivative at the first and last interior data points (assuming the points are arranged in ascending order), then the cubic spline is called a Not-a-Knot cubic spline. In cubic RBF terms, this means moving the centers at those points outside the interval. In other words, the set of points at which interpolation conditions are imposed is slightly different from the set RBF centers.

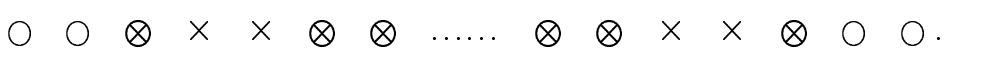
This method is illustrated as follows



where the ◯'s represent the centers and the ×'s the data locations. The approximation in the interior is independent of how far the centers are moved outside of the interval.

4. Other types of RBFs. Cubic RBFs are very convenient for analysis in 1-D because of their very close connection to cubic splines. However, many other choices of RBFs are available. We give below a few brief comments on some of these.

4.1. Quintic. Just as cubic RBF approximations are closely related to cubic splines, quintic RBF approximations are similarly related to quintic splines. The natural spline will in this case be defined by $s'''(-1) = s^{(4)}(-1) = 0$, $s'''(1) = s^{(4)}(1) = 0$ and is realized via RBFs when interpolating with $s(x) = \alpha + \beta x + \gamma x^2 + \sum_{i=0}^{N} \lambda_i |x - x_i|^5$ under the constraints $\Sigma\lambda_i = 0$, $\Sigma\lambda_i x_i = 0$, $\Sigma\lambda_i x_i^2 = 0$. Outside the interval, $s(x)$ will grow at most quadratically with x (compared to at most linearly in the cubic case). The Not-a-Knot boundary condition will in the quintic case have a distribution of centers and interpolation nodes



The first two interior node locations on each side have their centers moved outside the interval (again, their exact final locations do not matter, as far as the approximation within the interval is concerned).

In general, errors when using quintic RBF approximations will be two orders of h better than for cubic RBFs. However, Figure 4.1(a) shows that the condition numbers of the quinctic RBF interpolation matrices are orders of magnitude greater than their cubic counterparts.

4.2. Thin plate splines (TPS). Like cubic and quintic RBFs these are parameter free. Their justification includes extensive theoretical accuracy results and a variational theory in 2-D [3], [12]. Figure 4.1(a) also shows that the TPS RBF interpolation matrices are very well conditioned.

4.3. Wendland functions. Unlike the RBFs mentioned previously these are compactly supported, and their exact form depends on the number of space dimensions of the approximation. Two examples of Wendland functions are $\phi_{1,2}(r) = (1-r)_+^5(8r^2 + 5r + 1)$ and $\phi_{2,2}(r) = (1-r)_+^6(35r^2 + 18r + 3)$, constructed for use in 1-D and 2-D, respectively. These RBFs require some compromise between wide support/good accuracy and good sparsity/low accuracy, an interesting realization of which is the multiscale iterative method described in [4]. For an illustration of how the conditioning of the RBF interpolation matrix using the Wendland function $\phi_{1,2}$ (denoted by Wend12) increases as the support radius (SR) and the number of centers/data values increases, see Figure 4.1(b).

4.4. Multiquadric (MQ). These were first applied by Hardy [9] and perform well in applications [8], [14]. MQ RBFs are infinitely smooth and involve a free parameter. There is a trade off in the choice of this parameter: small ε leads to good accuracy, while large ε provides good conditioning. Figure 4.1(c) shows how the condition number of MQ RBF interpolation matrices increases with decreasing ε and increasing N.

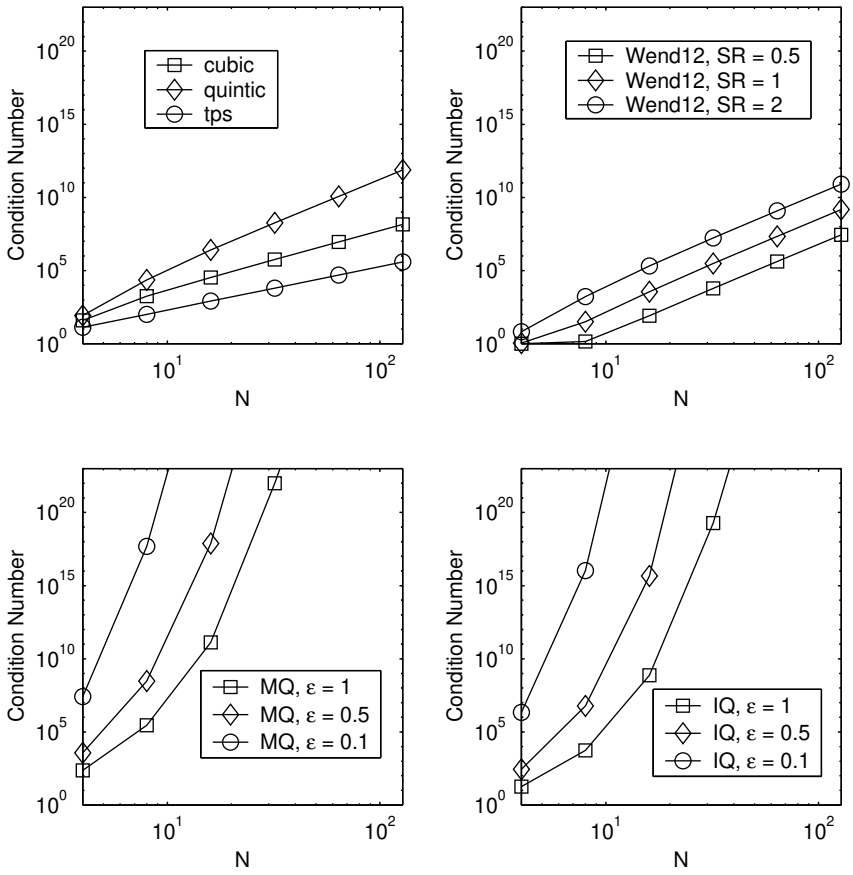


FIG. 4.1. *Condition numbers of RBF interpolation matrices as a function of the number of centers/data nodes in 1-D.*

4.5. Inverse quadratic (IQ). This case is of interest since

- it contains a free parameter (like MQ) which can to be adjusted for best trade off between accuracy and conditioning, and
- its form is algebraically simple enough to allow more closed-form analysis than, say, multiquadrics.

An example of the latter is the following closed-form expression of the RBF coefficients for a cardinal function on an unbounded, unit-spaced grid:

$$\lambda_k = \frac{(-1)^k \, \varepsilon \, \sinh \frac{\pi}{\varepsilon}}{\pi^2} \int_0^\pi \frac{\cos k\xi}{\cosh \frac{\xi}{\varepsilon}} d\xi. \tag{4.1}$$

This can be derived from [7, equation (7)].

We can also find the cardinal function itself, for arbitrary ε, as

$$s(x,\varepsilon) = \frac{2 \sinh \frac{\pi}{\varepsilon} \, \sin \pi x}{\pi(\cosh \frac{2\pi}{\varepsilon} - \cos 2\pi x)} \left\{ \frac{\sinh \frac{\pi}{\varepsilon} \, \cos \pi x}{x} + \cosh \frac{\pi}{\varepsilon} \int_0^\pi \sin x\xi \, \tanh \frac{\xi}{\varepsilon} \, d\xi \right\}. \tag{4.2}$$

The key ingredient in working out $s(x,\varepsilon) = \sum_{k=-\infty}^{\infty} \lambda_k/(1+(\varepsilon(k-x))^2)$ is the formula

$$\sum_{k=-\infty}^{\infty} \frac{(-1)^k \cos k\xi}{1+(\varepsilon(k-x))^2} =$$

$$\frac{2\pi}{\varepsilon} \cdot \frac{\cos \pi x \cos x\xi \cosh \frac{\xi}{\varepsilon} \sinh \frac{\pi}{\varepsilon} + \sin \pi x \sin x\xi \cosh \frac{\pi}{\varepsilon} \sinh \frac{\xi}{\varepsilon}}{\cosh \frac{2\pi}{\varepsilon} - \cos 2\pi x}, \quad \xi \in [-\pi, \pi]. \tag{4.3}$$

Figure 4.1(d) illustrates the effect of ε on the conditioning of the RBF interpolation matrix. One way the ill-conditioning manifests itself is in rapid growth of the expansion coefficients λ_i as $\varepsilon \to 0$. Equation (4.1) implies

$$\lambda_k = \frac{\varepsilon^2 (-1)^k \sinh\left(\frac{\pi}{\varepsilon}\right)}{2\pi \cosh\left(\frac{k\pi\varepsilon}{2}\right)} + O(\varepsilon)$$

uniformly in k. As $\varepsilon \to 0$ each coefficient λ_k will grow exponentially:

$$\lambda_k \approx \frac{\varepsilon^2 (-1)^k e^{(\pi/\varepsilon)}}{4\pi} \tag{4.4}$$

Despite the fact that the system is ill-conditioned, especially as $\varepsilon \to 0$, the RBF approximation remains bounded and well defined (indeed, for the cardinal data, (4.2) shows that $\lim_{\varepsilon \to 0} s(x,\varepsilon) = \frac{\sin \pi x}{\pi x}$). Issues regarding limits for $\varepsilon \to 0$ are discussed more in [2].

5. Edge effects and possible remedies in 1-D. All three types of end conditions for cubic splines/RBFs mentioned in Section 3 produce $O(h^4)$ convergence in the interior, but they differ significantly in their accuracy near the ends. We list below a few additional ways one might use to improve the edge accuracy. Later in this section, we will test all the different approaches, when combined with different RBF functions, in 1-D. Section 6 will contain similar tests for 2-D.

5.1. Adding polynomial terms. Although natural spline (enforcing the usually incorrect $s''(x) = 0$ at both ends) is quite inaccurate, the RBF implementation of it can be generalized. Instead of adding 1, x as available basis functions and enforcing $\sum \lambda_i = 0$, $\sum \lambda_i x_i = 0$ we can add 1, x, x^2,...,x^p and enforce $\sum \lambda_i = 0$, $\sum \lambda_i x_i = 0$, $\sum \lambda_i x_i^2 = 0$,..., $\sum \lambda_i x_i^p = 0$ where p takes any of the values 0, 1, 2, ... , N (and where N is one less than the number of grid points x_i, $i = 0, 1, \ldots, N$). The case $p = 1$ corresponds to a natural spline, and $p = N$ corresponds to classical polynomial interpolation (the $N+1$ constraints now force all $N+1$ RBF coefficients to vanish), Figure 5.1 shows the results of a numerical experiment using cubic RBFs with varying p. The front curve shows $f(x) = -\arctan(5(x+\frac{1}{2}))$, and the following curves different approximations obtained by interpolating $f(x)$ at 11 equispaced points over [-1,1] (i.e. $N = 10$). The first of the approximations is basic cubic RBF (no end corrections); behind this follow the results for $p = 0, 1, \ldots, 10$.

Figure 5.2 illustrates the accuracy of the derivative approximation to this function $f(x)$ across [-1,1] as the degree of the global polynomial is varied. If the function is not well resolved (as is the case here), increasing p hurts the accuracy. On the other hand, we would expect—and will indeed see in 2-D—that for a highly resolved smooth function which is well approximated by low degree polynomials, this approach can significantly reduce boundary errors.

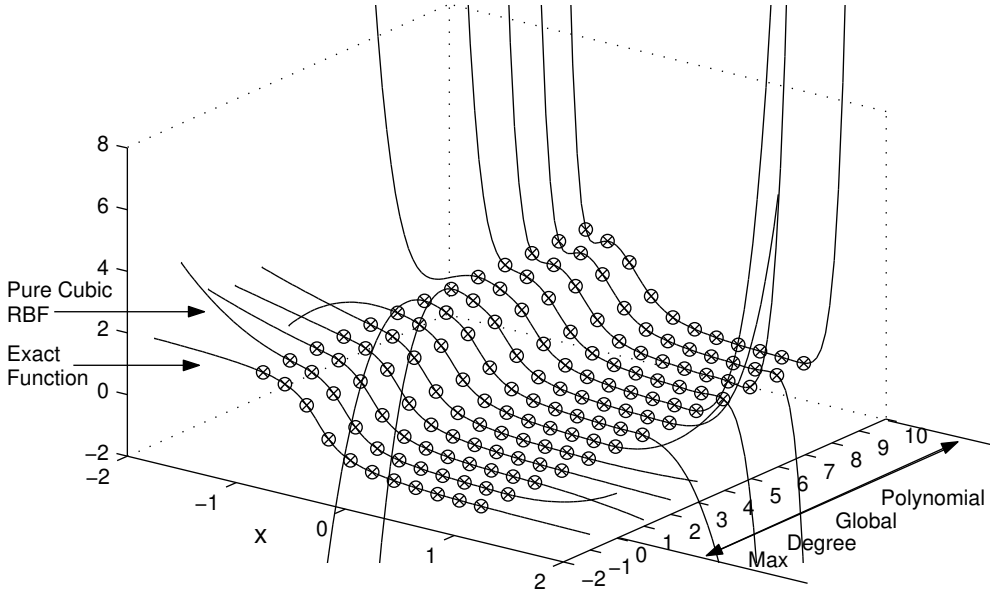


FIG. 5.1. *Interpolation/extrapolation of the function* $f(x) = -\arctan(5(x+\frac{1}{2}))$ *(shown as front curve) when using different combinations of cubic RBFs with global polynomial terms (and matching constraints).*

5.2. Super Not-a-Knot (SNaK). The idea behind SNaK is to modify Not-a-Knot by shifting the outermost two centers entirely outside the domain.

The centers vs. data points for SNaK are illustrated by

$$\bigcirc\ \bigcirc\ \times\ \times\ \otimes\ \otimes\ \otimes\ \ldots\ldots\ \otimes\ \otimes\ \otimes\ \times\ \times\ \bigcirc\ \bigcirc\,.$$

One can now show for cubic RBFs that

- The RBF approximation across the interval (indeed up to the innermost of the translated centers) is entirely independent of how far the centers have been shifted out (hence within the interval, the result is identical to a standard Not-a-Knot spline), and
- $O(h^4)$ accuracy holds also a distance h outside the main interval (assuming the shifted centers are still further out). This improves on the Not-a-Knot accuracy of $O(h^3)$ and natural spline accuracy of $O(h^2)$.

Figure 5.3 graphically compares the SNaK method against regular Not-a-Knot and basic cubic RBF approximation. The increased order of accuracy is not well visible in this type of graph (we will however see it in a following 2-D test case).

5.3. Boundary clustering of nodes. The idea here is borrowed from polynomial interpolation in 1-D. High-degree equispaced interpolation features a disastrous Runge phenomenon (usually exponential blow-up towards the end of an interval, as seen in the $p = 10$ case from Figure 5.1). Chebyshev interpolation simply clusters the nodes denser at the boundaries, and entirely avoids the problem; this is a highly regarded approach for the accurate approximation of smooth functions and forms the foundation for non-periodic pseudospectral methods. The left part of Table 5.1 shows different node distributions that can be generated by varying a parameter γ, such that $\gamma = 0$ corresponds to an equispaced grid and $\gamma = 0.5$ to

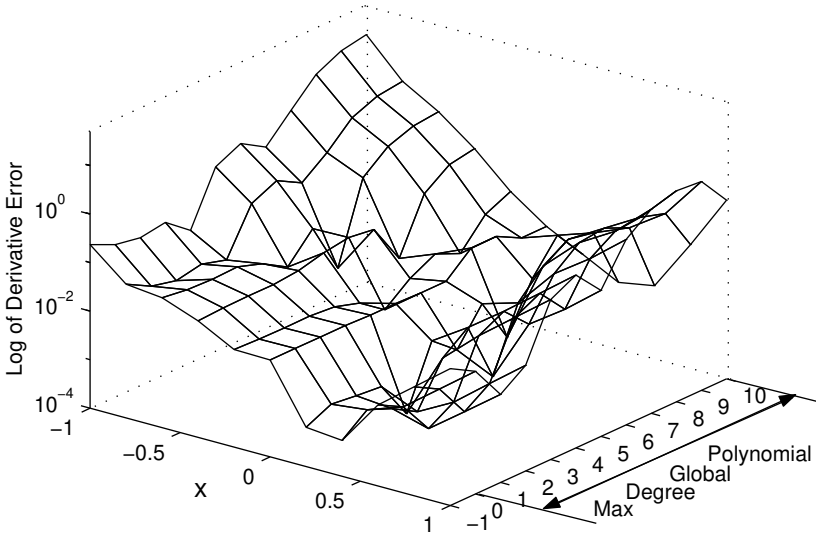


FIG. 5.2. *Log of the absolute error in the derivative approximation of the function* $f(x) = -\arctan(5(x+\frac{1}{2}))$ *when using different combinations of cubic RBFs with global polynomial terms (and matching constraints). The approximation was calculated using 11 equispaced centers/data values across* $[-1,1]$.

γ	Illustration of the distribution of 41 data values over $[-1,1]$	Max error in maginude MQ, $\varepsilon = 3$	Max Error in magnitude IQ, $\varepsilon = 3$
0.0		$4.03 \cdot 10^{-6}$	$2.29 \cdot 10^{-5}$
0.1		$3.91 \cdot 10^{-7}$	$2.85 \cdot 10^{-6}$
0.2		$1.69 \cdot 10^{-6}$	$1.65 \cdot 10^{-6}$
0.3		$1.52 \cdot 10^{-6}$	$1.39 \cdot 10^{-6}$
0.4		$6.61 \cdot 10^{-6}$	$1.09 \cdot 10^{-5}$
0.5		$2.41 \cdot 10^{-5}$	$2.32 \cdot 10^{-5}$

TABLE 5.1

Distribution of 41 data values over $[-1,1]$ *for different values of* γ *and the corresponding maximum (in magnitude) error across the interval in the MQ and IQ approximations to the function* $f(x) = -\arctan(5(x+\frac{1}{2}))$

a Chebyshev-spaced grid (details on how these various distributions are calculated with γ are given in [6, Section 3.3]). As the table shows, increasing γ lowers the resolution at the center of the interval. In the case of polynomial interpolation, this is a small price to pay for a vast edge improvement.

The right part of Table 5.1 shows the maximum (in magnitude) difference between the function $f(x) = -\arctan(5(x+\frac{1}{2}))$ and the MQ and IQ RBF approximations using the different distributions of nodes. We can see from the table that both approximations benefit from some amount of boundary clustering. A Chebyshev polynomial interpolant (based on

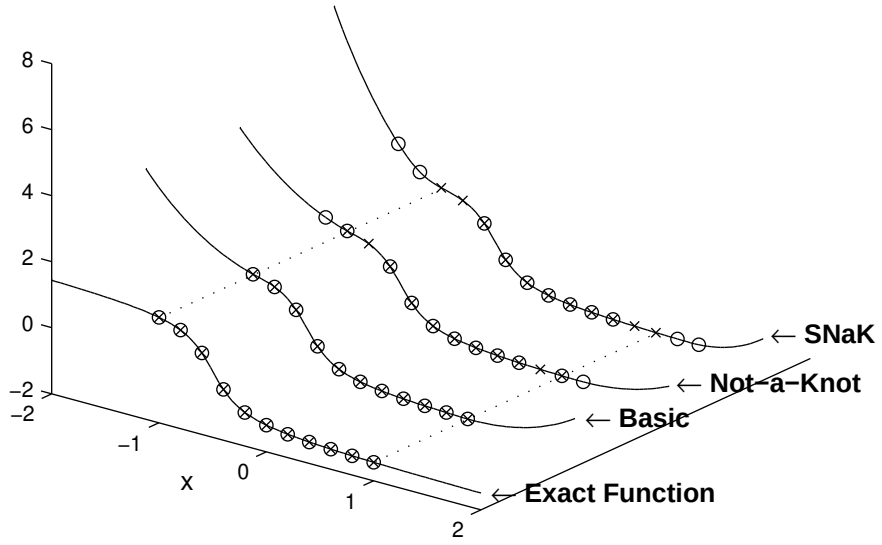


FIG. 5.3. *Comparison of interpolation/extrapolation of the function* $f(x) = -\arctan(5(x+\frac{1}{2}))$ *for basic, Not-a-Knot, and Super Not-a-Knot (SNaK) RBFs*

the $\gamma = 0.5$ grid) for this test case features a maximum error of $1.28 \cdot 10^{-5}$. Thus both MQ and IQ RBFs do here about an order of magnitude better than the highly regarded Chebyshev approximation, and this without needing to resort to nearly as severe edge clustering (a benefit to the stability restriction in a time-dependent problem).

5.4. Comparisons between the different edge improvement methods in 1-D. All the edge improvement methods we have introduced apply not only to cubic RBFs, but also to all other RBFs (although smooth RBFs technically do not have "knots", we keep the notation of Not-a-Knot and SNaK for these RBFs as well). Figure 5.4 compares the errors when the different boundary correction methods are combined with different RBFs for our $f(x) = -\arctan(5(x + 1/2))$ test case. Whereas Figures 5.1 and 5.3 showed approximations based on 11 nodes, we show here errors when using 21 nodes. The function is very coarsly sampled; the top row of subplots shows significant interior errors. None of the edge corrections can be expected to reduce these (as the results confirm). We do however see reductions of edge errors in most cases. Since the main interest in using RBFs is in two or more dimensions, we postpone more extensive comparisons to the next section, where we use scattered 2-D node distributions.

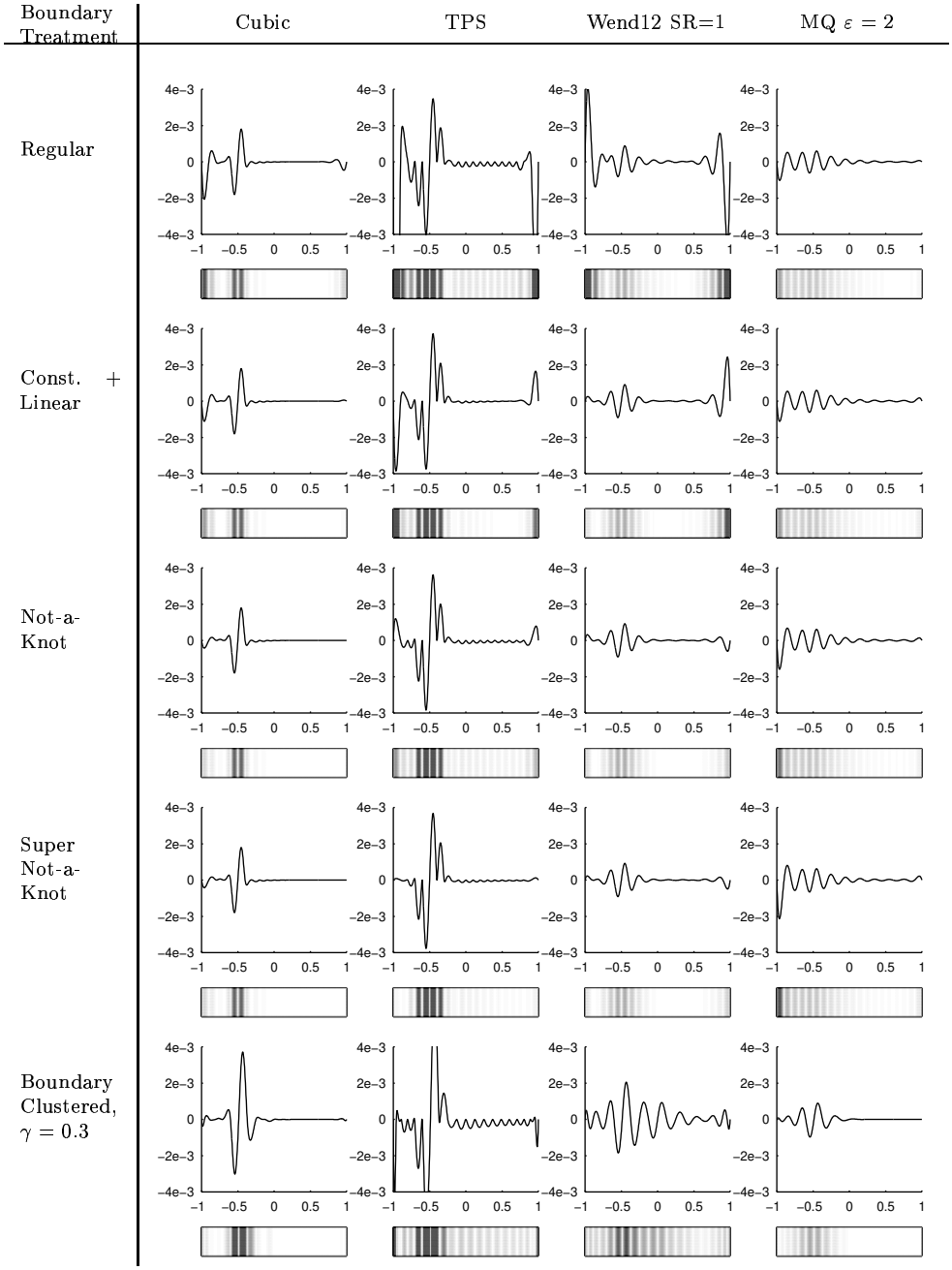
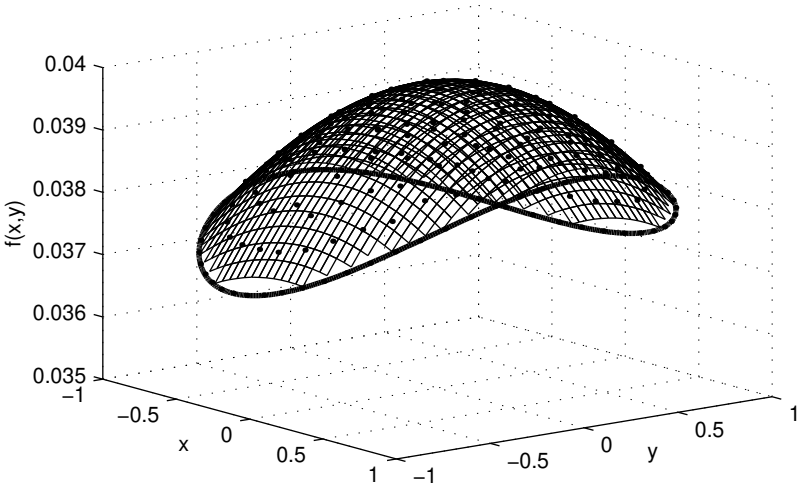
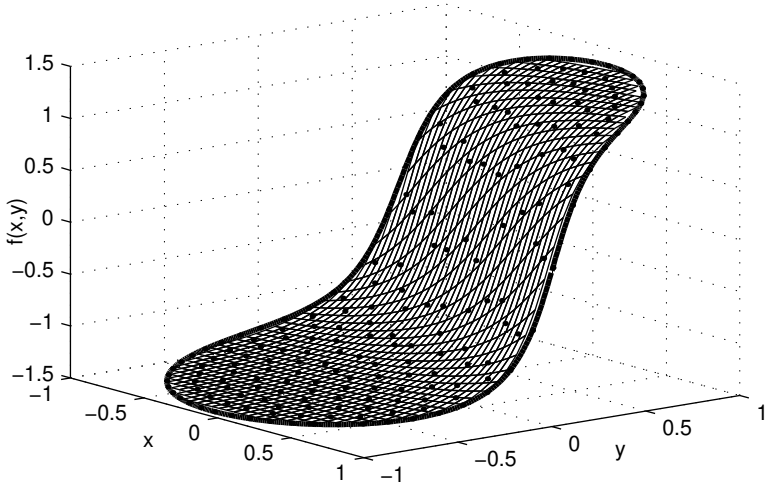


FIG. 5.4. *Comparison of different RBFs in 1-D with varying boundary treatments. The top picture in each row shows a plot of the error in the RBF approximation to* $f(x) = -\arctan(5(x + 1/2))$ *using 21 data points across* $[-1, 1]$. *The bottom picture shows a gray scale image of the absolute error in the RBF approximation across the interval. Errors range from 0 (white) to 0.002 (black).*



(a) $f(x) = \frac{1}{25+(x-1/5)^2+2y^2}$



(b) $f(x) = \arctan(2(x + 3y - 1))$

FIG. 6.1. *Test functions for the 2D experiments. The black solid circles represent the data locations used in all experiments shown in Figure 6.3 and 6.4 except the boundary clustering experiment.*

6. Edge effects and overall accuracy in 2-D. It is only in two (and higher) dimensions that the geometric flexibility of the RBF approximations becomes important enough to justify the relatively high computational cost of the approach. We make below some observations about the different edge enhancement methods, and compare how they hold up when applied to the test functions given in Figure 6.1(a) and (b).

6.1. Inclusion of low-order polynomial terms. One can choose different types of extra terms, and also different types of constraints (as long as their number agree). Low-order polynomial-type constraints arise however very naturally if one wants to regularize the far field RBF approximation.

Example: Consider $\phi(r) = r^3$. Show that the far field values of

$$s(x,y) = \sum \lambda_i ((x-x_i)^2 + (y-y_i)^2)^{3/2}$$

are minimized if one successively enforces

$$\begin{array}{l} \sum \lambda_i = 0, \\ \sum \lambda_i x_i = 0, \quad \sum \lambda_i y_i = 0, \\ \sum \lambda_i x_i^2 = 0, \quad \sum \lambda_i x_i y_i = 0, \quad \sum \lambda_i y_i^2 = 0, \\ \cdots \end{array} \tag{6.1}$$

Solution: If we set $x = r\ \cos\theta$, $y = r\ \sin\theta$ and then expand (6.1) in powers of r around $r = \infty$, we get

$$\begin{array}{l} s(x,y) = \sum \lambda_i ((x-x_i)^2 + (y-y_i)^2)^{3/2} = \\ \\ r^3 \left\{\sum \lambda_i\right\} + \\ r^2 \left\{(-3\cos\theta)\ \sum \lambda_i x_i + (-3\sin\theta) \sum \lambda_i y_i\right\} + \\ r^1 \left\{\frac{3}{4}(3+\cos 2\theta) \sum \lambda_i x_i^2 + (3\cos\theta\ \sin\theta) \sum \lambda_i x_i y_i + \frac{3}{4}(3-\cos 2\theta) \sum \lambda_i y_i^2\right\} + \\ \cdots \end{array}$$

The expansion continues with terms for r^0, r^{-1}, $r^{-2}, \ldots$ giving, for each power of r, another row which vanishes if and only if the conditions in the matching row of (6.1) are satisfied. □

The algebra proceeds essentially the same for all standard ϕ-functions, and results (at least in all cases we have tested) in the same constraints (6.1). Because the derivation was based on expanding in powers of r around $r = \infty$, one might think that thin plate splines might lead to different types of constraints (given the fact that $r^2 \log r$ has a branch point at $r = \infty$). However, it transpires that, although some $\log r$-factors also enter, the conditions associated with the smallest far field again become (6.1).

Given the form of the constraints in (6.1), it seems quite natural to let the extra functions to include be

$$\begin{array}{llll} 1, & & & \\ x, & y, & & \\ x^2, & xy, & y^2, & \\ \cdots & & & \end{array}$$

The case of including constant and linear extra functions, with matching constraints, is illustrated in the second row of Figures 6.3 and 6.4. These approximations were computed using the distribuition of data points and centers shown in Figure 6.2(a).

6.2. Not-a-Knot and Super Not-a-Knot. Not-a-Knot and Super Not-a-Knot worked very well for the piecewise smooth RBFs in 1-D. For our 2-D experiments, we started with the distribution of 200 data points shown in Figure 6.2(a). The data points are irregularly distributed, but with a fairly uniform density both along the domain boundary and throughout the interior. The Figures 6.2(b) and 6.2(c) show how we moved out some centers (open circles) from inside to outside the domain for our Not-a-Knot and Super Not-a-Knot implementaions, respectively. The numerical experiments with these implementations are shown in the third and fourth row of Figures 6.3 and 6.4.

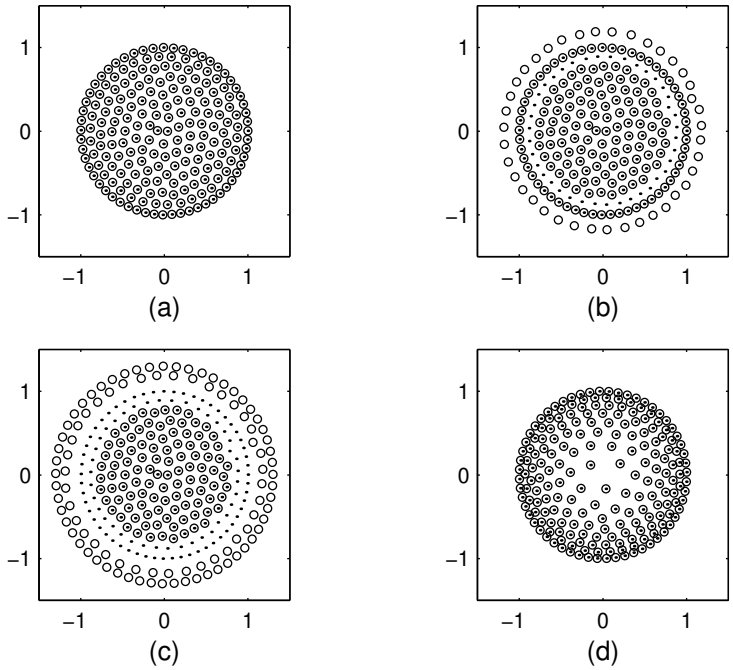


FIG. 6.2. *Different data/center distributions used in the numerical experiments. The solid circles represent the locations of the data while the open circle represent the locations of the centers.*

6.3. Boundary clustering of nodes. Following the ideas of Section 5.3, the nodes are distributed so that they are denser at the boundaries. Figure 6.2(d) shows the distribution used in our numerical experiments. The last row of Figures 6.3 and 6.4 shows the various RBF approximations using this distribution of data.

6.4. Comparisons between the different edge improvement methods in 2-D. Figures 6.3 and 6.4 illustrate the errors when different RBFs and edge enhancement techniques are applied to the two test cases shown in Figure 6.1. The first case ($f(x) = 1/(25 + (x - 1/5)^2 + 2y^2)$) is very smooth, and resolving it over 200 scattered nodes is representative of the situation when using RBFs for high-accuracy solutions of PDEs. The top row of subplots in Figure 6.3 shows that edge errors dominate for all the choices of RBFs. The next four rows of subplots show that all of the boundary correction approaches are effective. The "constant plus linear" works well in all cases of RBFs. The small residual ring still visible for TPS, Wend22 and MQ suggest that possibly including further terms might improve the boundary situation even more. Both variations of Not-a-Knot are highly effective for all RBF cases apart from Wend22. The boundary clustering idea needs to be used with caution—it appears to be the most "delicate" approach of those tested. Too much boundary clustering hurts overall accuracy through depletion of points in the interior.

It would be of interest to test the different boundary correction methods in more complex geometries. Quite possibly, the Not-a-Knot type conditions—being more "local" in character than including "constant plus linear" global functions—would adapt better to the task of edge correction along lengthy and irregular boundaries.

The test function $f(x) = \arctan(2(x + 3y - 1))$ is (like our 1-D arctangent) too rough to be well resolved on the present scattered set of nodes. The top row in Figure 6.4 shows

interior errors dominating. None of the boundary correction methods offer any help against this—indeed, boundary clustering, with its associated interior node depletion, is outright counterproductive in this context.

At first glance, it might appear that the errors in this last test case are disappointingly large. We show next that this is not the case—they are in fact remarkably small.

6.5. Accuracy comparison for RBFs against some other techniques. It needs to be stressed that the errors we have seen in 6.3 and (especially) 6.4 actually are very good. The test function $f(x, y) = \arctan(2(x + 3y - 1))$ turns out to be surprisingly difficult to fit well with any method which uses data at about 200 node points. Figure 6.5 compares the errors for two different approaches:

- Multiquadric RBFs, 200 nodes, irregular grids (shown in 6.2(a) and (c)), with $\varepsilon = 3$.
- Polar grid pseudospectral (regular grid with 16 points Chebyshev distributed across each $r \in [-1, 1]$, 14 angles $\theta \in [0, \pi)$; a total of 224 distinct points).

In the first case (top row in Figure 6.5) the MQ method is enhanced with the Super Not-a-Knot boundary treatment. The error is very small (around 10^{-7})), but even so larger than that of the Fourier-Chebyshev method which for very smooth functions becomes extremely accurate (for this case the error is around 10^{-9}). However, for the second test function, MQ is seen to be far more accurate than Fourier-Chebyshev. That is extremely encouraging—the latter method is well established as being of particularly high accuracy in the very advantageous case of perfectly regular grids. (Although here somewhat less costly and better conditioned than MQ, we should note that the Fourier-Chebyshev method does not generalize to irregular domains).

7. Concluding observations. Even if a scattered node set is relatively coarse, approximations based on smooth RBFs are found to be highly effective in approximating smooth functions (compared even to Chebyshev pseudospectral methods on regular grids). In cases of very fine resolution, edge errors start to dominate over interior errors. For this situation, several edge enhancement techniques have been studied here. They all prove to be effective, at insignificant or no extra computational cost (with the SNaK method generally the best for all our RBF choices apart from the Wendland function $\phi_{2,2}(r)$). The correction approaches ameliorate or altogether eliminate the usually notorious problem of approximating function and derivative values near edges of computational domains.

The idea of deploying some centers outside the main domain of interest appears also in some recent papers, e.g. [1] and [5], although arrived at from different initial considerations.

REFERENCES

[1] K. Balakrishnan, R. Jureshkumar, and P. Ramachandran, *An operator splitting-radial basis function method for the solution of transient nonlinear Poisson problems*, This one, (2001), p. in this issue.

[2] T. A. Driscoll and B. Fornberg, *Interpolation in the limit of increasingly flat radial basis functions*, This one, (2001), p. in this issue.

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

[4] G. E. Fasshauer, *Solving partial differential equations with radial basis functions: multilevel methods and smoothing*, Adv. Comp. Math., (1999), pp. 139–159.

[5] A. Fedoseyeve, M. Friedman, and E. Kansa, *Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary*, This one, (2001), p. in this issue.

[6] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, 1996.

[7] B. Fornberg and M. Ghrist, *Spatial finite difference approximations for wave-type equations*, SIAM J. Numer. Anal., 37 (1999), pp. 105–130.

[8] R. Franke, *Scattered data interpolation: tests of some methods*, Math. Comp., 38 (1982), pp. 181–200.
[9] R. L. Hardy, *Multiquadric equations of topograpy and other irregular surfaces*, J. Geophy. Res., 76 (1971), pp. 1905–1915.
[10] Y. C. Hon and X. Z. Mao, *An efficient numerical scheme for Burgers' equation*, Appl. Math. Comput., 95 (1998), pp. 37–50.
[11] E. J. Kansa, *Multiquadrics – a scattered data approximation scheme with applications to computational fluid-dynamics – II: Solutions to parabolic, hyperbolic and elliptic partial differential equations*, Comput. Math. Appl., 19 (1990), pp. 147–161.
[12] M. J. D. Powell, *The theory of radial basis function approximation in 1990*, in Advances in Numerical Analysis, Vol. II: Wavelets, Subdivision Algorithms and Radial Functions, W. Light, ed., Oxford University Press, Oxford, UK, 1990, pp. 105–210.
[13] M. J. D. Powell, *Univariate multiquadric interpolation: some recent results*, in International Conference on Curves and Surfaces, Chamonix, June 1990.
[14] S. Rippa, *Interpolation and smoothing of scattered data by radial basis functions*, Master's thesis, Tel Aviv University, Israel, 1984.
[15] R. Schaback, *Radial basis functions viewed from cubic splines*, in Proceedings of Manheim Conference, Birkhauser, 1997.
[16] H. Wendland, *Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree*, Adv. in Comput. Math., 4 (1995), pp. 389–396.

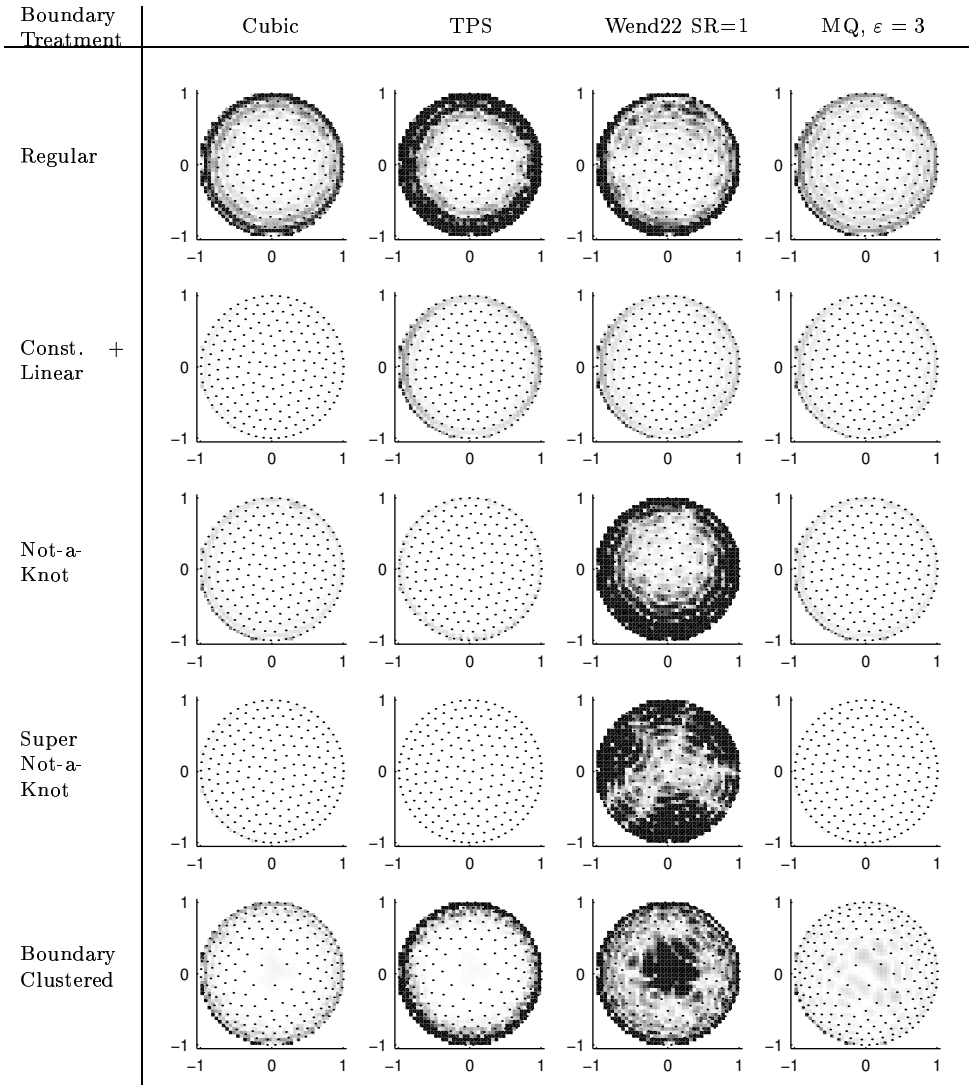


FIG. 6.3. *Comparison of the (absolute) error of different RBFs with varying boundary treatments for the function* $f(x,y) = \frac{1}{25+(x-1/5)^2+2y^2}$ *inside the unit circle. The black solid circles are the locations of the data values. Errors range from 0 (white) to* $2.0 \cdot 10^{-5}$ *(black).*

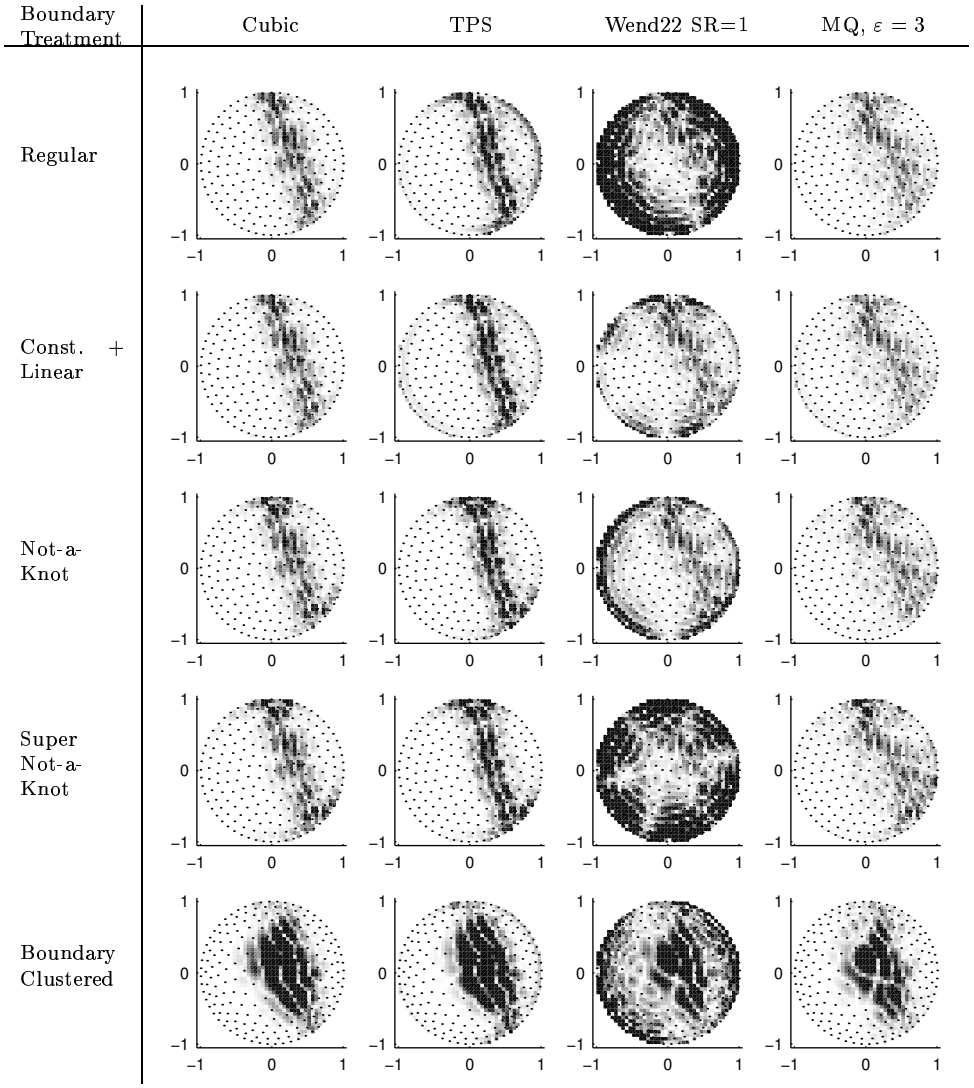


FIG. 6.4. *Comparison of the (absolute) error of different RBFs with varying boundary treatments for the function* $f(x, y) = \arctan(2(x + 3y - 1))$ *inside the unit circle. The black solid circles are the locations of the data values. Errors range from 0 (white) to 0.01 (black).*

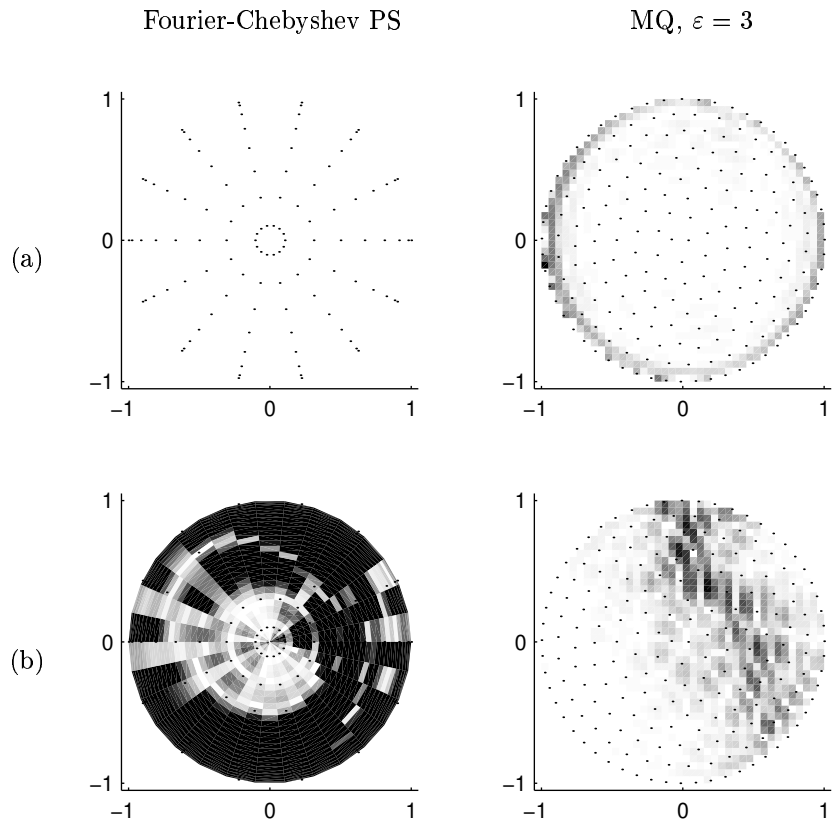


FIG. 6.5. *Comparison of the (absolute) error of Fourier-Chebyshev pseudospectral approximation at 224 data locations (shown by black solid circles) to the (absolute) error of the MQ RBF approximation at 200 scattered data locations (shown by black solid circles). Part (a) shows the errors, ranging from 0 (white) to* $2.0 \cdot 10^{-7}$ *(black), to the function* $f(x, y) = 1/(25 + (x - 1/5)^2 + 2y^2)$. *The Super Not-a-Knot boundary treatment has been applied to MQ RBF approximation in this part. Part (b) shows the errors, ranging from 0 (white) to 0.01 (black), to the function* $f(x, y) = \arctan(2(x + 3y - 1))$. *No boundary treatment has been applied to the MQ RBF approximation in this part.*

Appendix B: Stable Computation of Multiquadric Interpolants for All Values of the Shape Parameter

Bengt Fornberg * and Grady Wright †
University of Colorado
Department of Applied Mathematics
CB-526, Boulder, CO 80309, USA

Abstract

Spectrally accurate interpolation and approximation of derivatives used to be practical only on highly regular grids in very simple geometries. Since radial basis function (RBF) approximations permit this even for multivariate scattered data, there has been much recent interest in practical algorithms to compute these approximations effectively.

Several types of RBFs feature a free parameter (e.g. c in the multiquadric (MQ) case $\phi(r) = \sqrt{r^2 + c^2}$). The limit of $c \to \infty$ (increasingly flat basis functions) has not received much attention because it leads to a severely ill-conditioned problem. We present here an algorithm which avoids this difficulty, and which allows numerically stable computations of MQ RBF interpolants for all parameter values. We then find that the accuracy of the resulting approximations, in some cases, becomes orders of magnitude higher than was the case within the previously available parameter range.

Our new method provides the first tool for the numerical exploration of MQ RBF interpolants in the limit of $c \to \infty$. The method is in no way specific to MQ basis functions and can—without any change—be applied to many other cases as well.

Keywords: Radial basis functions, RBF, multiquadrics, ill-conditioning

AMS subject classification: 41A21, 65D05, 65E05, 65F22

1 Introduction

Linear combinations of radial basis functions (RBFs) can provide very good interpolants for multivariate data. Multiquadric (MQ) basis functions, generated by $\phi(r) = \sqrt{r^2 + c^2}$ (or in the notation used in this paper, $\phi(r) = \sqrt{1 + (\varepsilon r)^2}$ with $\varepsilon = 1/c$), have proven to be particularly successful [1]. However, there have been three main difficulties with this approach: severe numerical ill-conditioning for a fixed N (the number of data points) and small ε, similar ill-conditioning problems for a fixed ε and large N, and high computational cost. This study shows how the first of these three problems can be resolved.

*fornberg@colorado.edu. This work was supported by NSF grants DMS-9810751 (VIGRE), DMS-0073048, and a Faculty Fellowship from the University of Colorado at Boulder.

†wrightg@colorado.edu. This work was supported by an NSF VIGRE Graduate Traineeship under grant DMS-9810751.

Large values of the parameter ε are well known to produce very inaccurate results (approaching linear interpolation in the case of 1-D). Decreasing ε usually improves the accuracy significantly [2]. However, the direct way of computing the RBF interpolant suffers from severe ill-conditioning as ε is decreased [3]. Several numerical methods have been developed for selecting the "optimal" value of ε (e.g. [4, 5, 6]). However, because of the ill-conditioning problem, they have all been limited in the range of values that could be considered, having to resort to high-precision arithmetic, for which the cost of computing the interpolant increases to infinity as $\varepsilon \to 0$ (timing illustrations for this will be given later). In this study, we present the first algorithm which not only can compute the interpolant for the full range $\varepsilon \geq 0$, but it does so entirely without a progressive cost increase as $\varepsilon \to 0$.

In the highly special case of MQ RBF interpolation on an infinite equispaced Cartesian grid, Buhmann and Dyn [7] showed that the interpolants obtain spectral convergence for smooth functions as the grid spacing goes to zero. For this same case, but with a fixed grid spacing, Baxter [8] showed the MQ RBF interpolant in the limit of $\varepsilon \to 0$ to cardinal data (equal to one at one data point and zero at all others) exists and goes to the multidimensional *sinc* function—just as the case would be for a Fourier spectral method. Limiting ($\varepsilon \to 0$) interpolants on scattered finite data sets were studied by Driscoll and Fornberg [9]. They noted that, although the limit usually exists, it can fail to do so in exceptional cases. The present numerical algorithm handles both of these situations. It also applies—without any change—to many other types of basis functions. The cases we will give computational examples for are

Name of RBF	Abbreviation	Definition
Multiquadrics	MQ	$\phi(r) = \sqrt{1 + (\varepsilon r)^2}$
Inverse Quadratics	IQ	$\phi(r) = \dfrac{1}{1 + (\varepsilon r)^2}$
Gaussians	GA	$\phi(r) = e^{-(\varepsilon r)^2}$

Note that in all the above cases the limits of flat basis functions correspond to $\varepsilon \to 0$.

The main idea of the present method is to consider the RBF interpolant at a fixed $\underline{x}$

$$s(\underline{x}, \varepsilon) = \sum_{j=1}^{N} \lambda_j \, \phi(\|\underline{x} - \underline{x}_j\|) \tag{1}$$

(where $\|\cdot\|$ is the 2-norm) not only for real values of ε, but as an analytic function of a complex variable ε. Although not explicitly marked, λ_j and ϕ are now functions of ε. In the large area around $\varepsilon = 0$, $s(\underline{x}, \varepsilon)$ becomes a meromorphic function of ε and can therefore be written as

$$s(\underline{x}, \varepsilon) = (\text{rational function in } \varepsilon) + (\text{power series in } \varepsilon) \tag{2}$$

The present algorithm numerically determines (in a stable way) the coefficients to the rational function and the power series. This allows us to use (2) for computing the RBF interpolant effectively numerically right down to $\varepsilon = 0$. The importance of this entirely new capability is expected to be as a tool to investigate properties of RBF approximations and not, at the present time, to interpolate any large experimental data sets.

Although not pursued here, there are a number of important and unresolved issues relating to the limit of the RBF interpolant as $\varepsilon \to 0$, for which the present algorithm will now allow numerical explorations. For example, it was shown by Driscoll and Fornberg [9] that the limiting interpolant in 1-D is simply the Lagrange interpolating polynomial. This, of course,

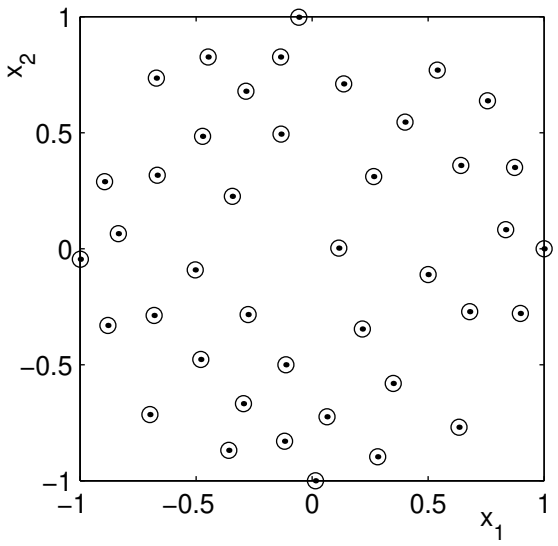


Figure 1: Distribution of 41 data points for use in the first test problem.

forms the foundation for finite difference and pseudospectral methods. The equivalent limit ($\varepsilon \to 0$) can now be studied for scattered data in higher dimensions. This is a situation where there, in general, does not exist any unique lowest-degree interpolating polynomial and, consequently, spectral limits have not received much attention.

The rest of this paper is organized as follows: Section 2 introduces a test example which we will use to describe the new method. In Section 3, we illustrate the structure of $s(\underline{x}, \varepsilon)$ in the complex ε-plane (the distribution of poles etc.). Section 4 describes the steps in the numerical method, which then are applied to our test problem in Section 5. Section 6 contains additional numerical examples and comments. We vary the number of data points and also give numerical results for some other choices of RBFs. One of the examples we present there features a situation where $\varepsilon \to 0$ leads to divergence. We finish by giving a few concluding remarks in Section 7.

2 First test problem

Figure 1 shows 41 data points $\underline{x}_j$ randomly scattered over the unit disk (in the $\underline{x}-$plane where $\underline{x}$ is a two-vector with components x_1, x_2). We let our data at these points be defined by the function

$$f(\underline{x}) = f(x_1, x_2) = \frac{59}{67 + (x_1 + \frac{1}{7})^2 + (x_2 - \frac{1}{11})^2} \tag{3}$$

The task is to compute the MQ RBF interpolant (i.e. (1) with $\phi(r) = \sqrt{1 + (\varepsilon r)^2}$) at some location $\underline{x}$ inside the unit disk. We denote the data by $y_j = f(\underline{x}_j)$, $j = 1, 2, \ldots, 41$. The immediate way to perform the RBF interpolation would be to first obtain the expansion

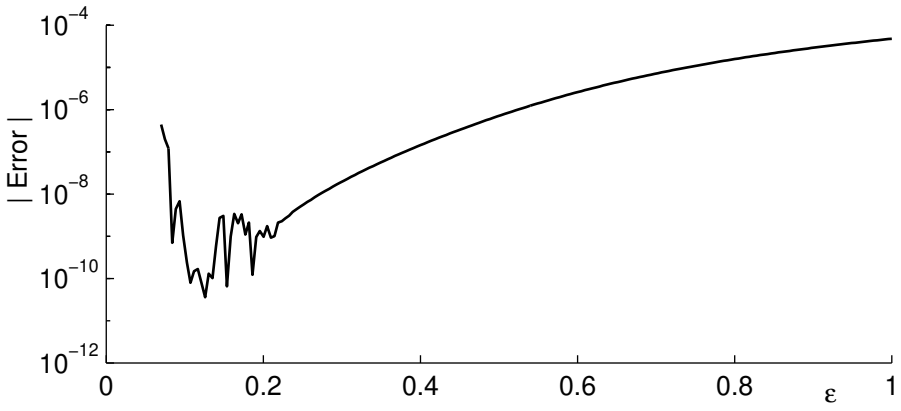


Figure 2: The error (in magnitude) as a function of ε in the interpolant $s(\underline{x}, \varepsilon)$ of (3) when $s(\underline{x}, \varepsilon)$ is computed directly using (6). We have chosen $\underline{x} = (0.3, -0.2)$.

coefficients λ_j by solving

$$\begin{bmatrix} \\ A(\varepsilon) \\ \\ \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_{41} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_{41} \end{bmatrix} \tag{4}$$

where the elements of $A(\varepsilon)$ are $a_{j,k} = \phi(\|\underline{x}_j - \underline{x}_k\|)$. The RBF interpolant, evaluated at $\underline{x}$, is then written as

$$s(\underline{x}, \varepsilon) = \sum_{j=1}^{41} \lambda_j \phi(\|\underline{x} - \underline{x}_j\|). \tag{5}$$

or equivalently

$$s(\underline{x}, \varepsilon) = \begin{bmatrix} \quad B(\varepsilon) \quad \end{bmatrix} \begin{bmatrix} \\ A(\varepsilon) \\ \\ \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_{41} \end{bmatrix} \tag{6}$$

where the elements of $B(\varepsilon)$ are $b_j = \phi(\|\underline{x} - \underline{x}_j\|)$.

Figure 2 shows the magnitude of the error $s(\underline{x}, \varepsilon) - f(\underline{x})$ where $\underline{x} = (0.3, -0.2)$ as a function of ε when computed directly via (4) and (5). This computation clearly loses its accuracy (in 64-bit floating point precision) when ε falls below approximately 0.2. The drop in error as ε approaches 0.2 (from above) suggests that computations for lower values of ε could be very accurate if the numerical instability could be overcome. The reason the onset of ill-conditioning occurs so far from $\varepsilon = 0$ is that the matrix $A(\varepsilon)$ approaches singularity very rapidly as ε decreases. Using Rouché's theorem, we can find that in this test case $\det(A(\varepsilon)) = \alpha \cdot \varepsilon^{416} + O(\varepsilon^{418})$ (where the coefficient α is non-zero). Regardless of this rapid approach to singularity, we usually find that $s(\underline{x}, \varepsilon)$ exists and is bounded as $\varepsilon \to 0$. This means an extreme amount of numerical cancellation occurs for small ε when evaluating $s(\underline{x}, \varepsilon)$.

In the notation of (6), our task is to then determine the row vector

$$\begin{bmatrix} \quad C(\varepsilon) \quad \end{bmatrix} = \begin{bmatrix} \quad B(\varepsilon) \quad \end{bmatrix} \begin{bmatrix} \\ A(\varepsilon) \\ \\ \end{bmatrix}^{-1} \tag{7}$$

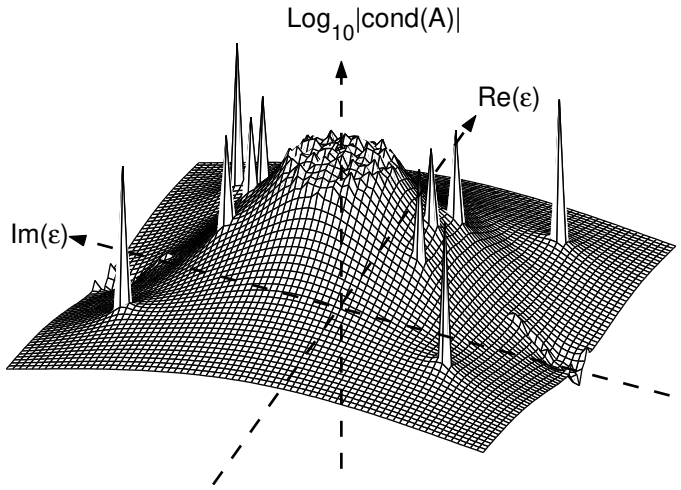


Figure 3: Logarithm (base 10) of the condition number for $A(\varepsilon)$ as a function of the complex variable ε. The domain of the plot is a square with sides of length $2 \cdot 0.75$ centered at the origin and the range of the plot varies from 0 to 10^{20}. Note near $\varepsilon = 0$ the $\log_{10}$ of the condition number of $A(\varepsilon)$ goes to infinity. However, due to numerical rounding, no values greater than 10^{20} were recorded.

for all $\varepsilon \geq 0$. To do this, we need an algorithm which bypasses the extremely ill-conditioned direct formation of $A(\varepsilon)^{-1}$ and computation of the product $B(\varepsilon) \cdot A(\varepsilon)^{-1}$ for any values of ε less than approximately 0.3. The algorithm we present in Section 4 does this by directly computing $C(\varepsilon)$ around some circle in the complex ε-plane where $A(\varepsilon)$ is well-conditioned. This will allow us to determine the coefficients in (2) and therefore determine $s(\underline{x}, \varepsilon)$ for small ε-values.

Note that in practice, we often want to evaluate the interpolant at several points. This is most easily done by letting $B(\varepsilon)$ (and thus $C(\varepsilon)$) in (7) contain several rows—one for each of the points.

3 Test problem viewed in a complex $\varepsilon-$plane

Figure 3 shows the $\log_{10}$ of the condition number of $A(\varepsilon)$ when ε is no longer confined to the real axis. We see that the ill-conditioning is equally severe in all directions as ε approaches zero in the complex plane. Furthermore, we note a number of sharp spikes. These correspond to complex ε-values for which $A(\varepsilon)$ is singular (none of these can occur on the real axis according to the non-singularity result by Micchelli [10]).

The linear system (4) could have been solved by Cramer's rule. That shows that $s(\underline{x}, \varepsilon)$, apart from the branch point singularities on the imaginary axis due to zeros of $\phi(r)$, can only have poles as singularities (thus justifying the analytic form stated (2)). The structure of $s(\underline{x}, \varepsilon)$ in the complex ε-plane is shown in Figure 4. The lined area marks where the ill-conditioning is too severe for direct computation of $s(\underline{x}, \varepsilon)$ in 64-bit floating-point. The solid circles mark simple poles. There are also branch points on the imaginary axis (marked by $\times$'s). For evaluation of $s(\underline{x}, \varepsilon)$ on the unit disk, these never get closer to the origin than $\pm\frac{i}{2}$.

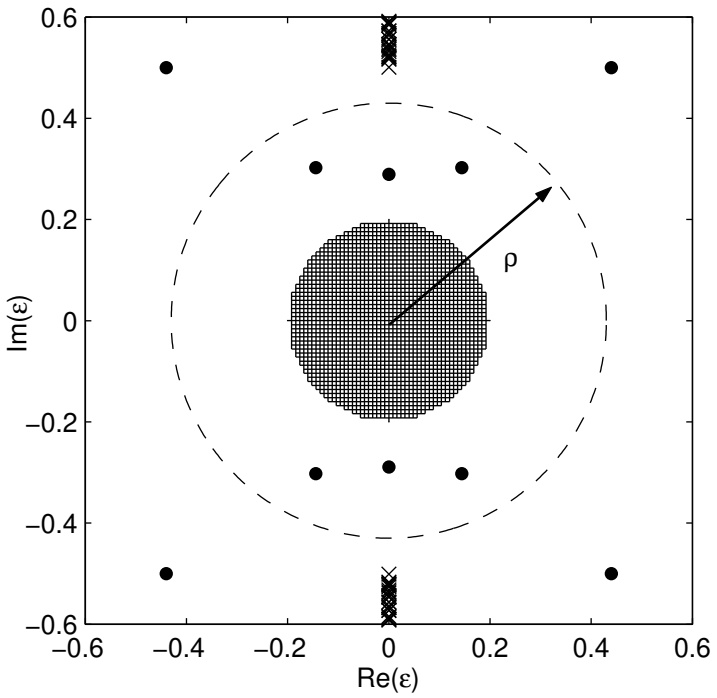


Figure 4: Structure of $s(\underline{x}, \varepsilon)$ in the complex ε-plane. The approximate area with ill-conditioning is marked with a line pattern; poles are marked with solid circles and branch points with $\times$'s.

The most notable feature of the analytic character of $s(\underline{x}, \varepsilon)$ is that the origin is usually only a removable singularity, a consequence of (2) together with the fact that the limit of $C(\varepsilon)$ as $\varepsilon \to 0$ usually exists.

The dashed line in Figure 4 indicates a possible contour (a circle) we can use in our method. Everywhere along such a circle, $s(\underline{x}, \varepsilon)$ can be evaluated directly with no particular ill-conditioning problems. Had there been no poles inside the circle, plain averaging of the $s(\underline{x}, \varepsilon)$-values around the circle would have given us $s(\underline{x}, 0)$.

It should be pointed out that if we increase the number of data points N too much (e.g. $N > 100$ in this example) the ill-conditioning region in Figure 4 will grow so that it contains some of the branch point singularities (starting at $\varepsilon = 0.5i$), forcing us to choose a circle that falls within this ill-conditioned region. However, we can still find $s(\underline{x}, \varepsilon)$ everywhere inside our circle for no worse conditioning than at ε just below 0.5.

To complete the description of our algorithm, we next discuss how to

- detect and compensate for the poles located inside our circle (if any), and
- compute $s(\underline{x}, \varepsilon)$ at any ε-point inside the circle (and not just at its center)

based only on ε-values around the circle.

4 Numerical method

We first evaluate $s(\underline{x}, \varepsilon)$ at equidistant locations around the circle of radius ρ that was shown in Figure 4, and then take the (inverse) fast Fourier transform (FFT) of these values. This produces the vector

d_0	ρd_1	$\rho^2 d_2$	$\rho^3 d_3$	...		...	$\rho^{-3} d_{-3}$	$\rho^{-2} d_{-2}$	$\rho^{-1} d_{-1}$

(here ordered as is conventional for the output of an FFT routine). From this (with $\varepsilon = \rho e^{i\theta}$) we have essentially obtained the Laurent expansion coefficients for $s(\underline{x}, \varepsilon)$. We can thus write

$$s(\underline{x}, \varepsilon) = \ldots + d_{-3}\varepsilon^{-3} + d_{-2}\varepsilon^{-2} + d_{-1}\varepsilon^{-1} + d_0 + d_1\varepsilon^1 + d_2\varepsilon^2 + d_3\varepsilon^3 + \ldots \,. \tag{8}$$

This expansion is convergent within some strip around the periphery of the circle. If there are no poles inside the circle all the coefficients in (8) with negative indices vanish, giving us the Taylor part of the expansion

$$s(\underline{x}, \varepsilon) = d_0 + d_1\varepsilon^1 + d_2\varepsilon^2 + d_3\varepsilon^3 + \ldots \,. \tag{9}$$

We can then use this to evaluate $s(\underline{x}, \varepsilon)$ numerically for any value of ε inside the circle.

The presence of any negative powers in (8) indicates that $s(\underline{x}, \varepsilon)$ has poles inside the circle. To account for the poles so that we can evaluate $s(\underline{x}, \varepsilon)$ for any value of ε inside the circle, we re-cast the terms with negative indices into Padé rational form. This is accomplished by first using the FFT data to form

$$q(\eta) = d_{-1}\eta + d_{-2}\eta^2 + d_{-3}\eta^3 + \ldots \,. \tag{10}$$

Next, we expand $q(\eta)$ in Padé rational form (see for example [11]), and then set

$$r(\varepsilon) = q(1/\varepsilon).$$

Since $s(\underline{x}, \varepsilon)$ can only posses a finite number of poles inside the circle, the function $r(\varepsilon)$ together with (9) will entirely describe $s(\underline{x}, \varepsilon)$ in the form previously stated in (2):

$$s(\underline{x}, \varepsilon) = \{r(\varepsilon)\} + \{d_0 + d_1\varepsilon + d_2\varepsilon^2 + \ldots\}.$$

This expression can be numerically evaluated to give us $s(\underline{x}, \varepsilon)$ for any value of ε inside the circle.

An automated computer code needs to monitor several consequences of the fact that we are working with finite and not infinite expansions. These are:

- $s(\underline{x}, \varepsilon)$ must be sampled densely enough so that the coefficients for the high negative and positive powers of ε returned from the FFT are small.
- When turning (10) into Padé rational form, we must choose the degrees of the numerator and denominator (which can be chosen to be equal) so that they match or exceed the total number of poles within our circle. (Converting the Padé expansion back to Laurent form and comparing coefficients offers an easy and accurate test that the degrees were chosen sufficiently high).
- The circular path must be chosen so that it is inside the closest branch point on the imaginary axis (equal to i/D where D is the maximum distance between points), but still outside the area where direct evaluation of $s(\underline{x}, \varepsilon)$ via (6) is ill-conditioned.
- The circular path must not run very close to any of the poles.

5 Numerical method applied to the test problem

We choose for example $M = 128$ points around a circle of radius $\rho = 0.42$ (as shown in Figure 4). This requires just $M/4 + 1 = 33$ evaluations of $s(\underline{x}, \varepsilon)$ due to the symmetry between the four quadrants. We again take $\underline{x} = (0.3, -0.2)$. Following the inverse FFT (and after "scaling away" ρ), we cast the terms with negative indices to Padé rational form to obtain:

$$r(\varepsilon) = \frac{-3.3297 \cdot 10^{-11} - 5.9685 \cdot 10^{-10}\varepsilon^2 - 1.8415 \cdot 10^{-9}\varepsilon^4 + 0 \cdot \varepsilon^6}{1.0541 \cdot 10^{-3} + 2.4440 \cdot 10^{-2}\varepsilon^2 + 2.2506 \cdot 10^{-1}\varepsilon^4 + \varepsilon^6} \ . \tag{11}$$

(The highest degree term in the numerator is zero because the expansion (10) contains no constant term). Combining (11) with the Taylor series approximation, we compute, for example, $s(\underline{x}, \varepsilon)$ at $\varepsilon = 0.1$:

$$s(\underline{x}, 0.1) \approx \{\ r(0.1)\} \ + \left\{ \sum_{k=0}^{32} d_{2k} (0.1)^{2k} \right\} \approx \ \ 0.87692244095761.$$

Note that the only terms present in the Taylor and Padé approximations are even, due to the four-fold symmetry of $s(\underline{x}, \varepsilon)$ in the complex ε-plane.

Table 1 compares the error in $s(\underline{x}, \varepsilon)$ when computed in standard 64-bit floating point with the direct method (6) and when computed with the present algorithm. The comparisons were made with $s(\underline{x}, \varepsilon)$ computed via (6) with high-precision arithmetic, using 60 digits of accuracy. The last part of the table compares the error in the approximation of (3), when $s(\underline{x}, \varepsilon)$ is computed using the present algorithm.

Magnitude of the error in $s(\underline{x}, \varepsilon)$
when computed using the direct method

	$\varepsilon = 0$	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.12$	$\varepsilon = 0.25$
	∞	$3.9 \cdot 10^{-3}$	$1.0 \cdot 10^{-6}$	$4.9 \cdot 10^{-10}$	$1.4 \cdot 10^{-9}$	$4.6 \cdot 10^{-11}$

Magnitude of the error in $s(\underline{x}, \varepsilon)$
when computed using the Contour-Padé algorithm

$\frac{M}{4}+1$	$\varepsilon = 0$	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.12$	$\varepsilon = 0.25$
33	...	$1.1 \cdot 10^{-13}$	$1.0 \cdot 10^{-13}$	$8.4 \cdot 10^{-14}$	$7.1 \cdot 10^{-14}$	$1.1 \cdot 10^{-13}$
65	...	$1.3 \cdot 10^{-13}$	$1.4 \cdot 10^{-13}$	$1.4 \cdot 10^{-13}$	$1.4 \cdot 10^{-13}$	$1.2 \cdot 10^{-13}$
129	...	$2.1 \cdot 10^{-13}$	$2.0 \cdot 10^{-13}$	$1.8 \cdot 10^{-13}$	$1.6 \cdot 10^{-13}$	$5.6 \cdot 10^{-14}$

Magnitude of the error $s(\underline{x}, \varepsilon) - f(\underline{x})$
when $s(\underline{x}, \varepsilon)$ is computed using the Contour-Padé algorithm

$\frac{M}{4}+1$	$\varepsilon = 0$	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.12$	$\varepsilon = 0.25$
33	$5.3 \cdot 10^{-11}$	$5.2 \cdot 10^{-11}$	$2.5 \cdot 10^{-11}$	$2.3 \cdot 10^{-12}$	$2.5 \cdot 10^{-13}$	$5.5 \cdot 10^{-9}$

Table 1: Comparison of the error in $s(\underline{x}, \varepsilon)$ when computed using the direct method and the Contour-Padé algorithm.

Figure 5 graphically compares the results of the Contour-Padé algorithm using $M/4+1 = 33$ to those using the direct method (6). Like Table 1, the figure clearly shows that the

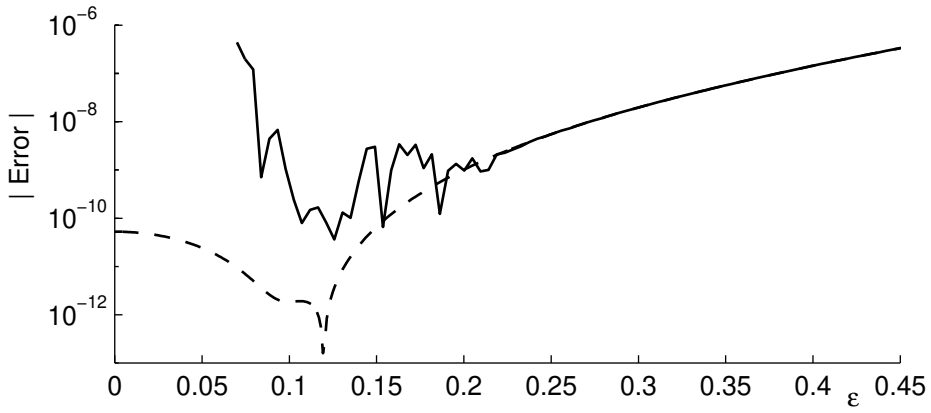


Figure 5: The error (in magnitude) as a function of ε in the interpolant $s(\underline{x}, \varepsilon)$ of (3). The solid line shows the error when $s(\underline{x}, \varepsilon)$ is computed using (6) and the dashed line shows the error when $s(\underline{x}, \varepsilon)$ is computed using the Contour-Padé algorithm presented in Section 4. Again, we have chosen $\underline{x} = (0.3, -0.2)$.

Contour-Padé algorithm allows the RBF interpolant to be computed in a stable manner for the full range of ε. (The increased error in the results of the Contour-Padé algorithm as ε falls below 0.12 is not due to any loss in computational accuracy; it is a genuine feature of the RBF interpolant, and will be discussed in a separate study).

We next compare the computational effort required to compute $s(\underline{x}, \varepsilon)$ using the direct method (6) and the Contour-Padé algorithm. To obtain the same level of accuracy (around 12 digits) with the direct method as the present algorithm provides requires the use of high-precision arithmetic. The table below summarizes the time required for computing the interpolant via the direct method using MATLAB's variable-precision arithmetic (VPA) package. All computations were done on a 500 MHz Pentium III processor.

ε	Digits Needed	Time for finding λ_j	Time for evaluating $s(\underline{x}, \varepsilon)$ at each $\underline{x}$
10^{-2}	42	172.5 sec.	1.92 sec.
10^{-4}	74	336.3 sec.	2.09 sec.
10^{-6}	106	574.6 sec.	2.31 sec.
10^{-8}	138	877.1 sec.	2.47 sec.

Note that in this approach, changing ε will necessitate an entirely new calculation.

With the Contour-Padé algorithm, the problem can be done entirely in standard 64-bit floating point. A summary of the time required to compute the various portions of the algorithm using MATLAB's standard floating point follows:

Portion of the Algorithm	Time
Finding the expansion coefficients around the ε circle and the poles for the Padé rational form	0.397 sec.
Evaluating $s(\underline{x}, \varepsilon)$ at a new $\underline{x}$ value.	0.0412 sec.
Evaluating $s(\underline{x}, \varepsilon)$ at a new ε value.	0.0022 sec.

Note that these times hold true regardless of the value of ε.

6 Some additional examples and comments

Looking back at the description of the Contour-Padé algorithm, we see that it only relied on computing $s(\underline{x}, \varepsilon)$ around a contour and was in no way specific to the MQ RBF. In the first part of this section we present some additional results of the algorithm and make some comments not only for the MQ RBF, but also for the IQ and GA RBFs.

We consider RBF approximations of (3) sampled at the 62 data points $\underline{x}_j$ shown in Figure 6. To get a better idea of how the error behaves over the whole region (i.e. the unit disk), we compute the root-mean-square (RMS) error of the approximations over a dense set of points covering the region. In all cases, we use the Contour-Padé algorithm with $M = 512$ points around the contour.

Figure 7 (a) shows the structure in the complex ε-plane for $s(\underline{x}, \varepsilon)$ based on the MQ RBF (we recall that the pole locations are entirely independent of $\underline{x}$). Unlike the example from Section 2 which resulted in 6 poles for $s(\underline{x}, \varepsilon)$, we see from the figure that the present example only results in 2 poles within the contour (indicated by the dashed line). Figure 8 (a) compares the resulting RMS error as a function of ε when the MQ RBF approximation is computed directly via (6) and computed using the Contour-Padé algorithm. The figure shows that the direct computation becomes unstable when ε falls below approximately 0.28. Most algorithms for selecting the optimal value of ε (based on RMS errors) would thus be limited from below by this value. However, the Contour-Padé algorithm allows us to compute the approximation accurately for every value of ε. As the figure shows, the true optimal value of ε is approximately 0.119. The RMS error in the approximation at this value is approximately $2.5 \cdot 10^{-12}$, whereas the RMS error in the approximation at $\varepsilon = 0.28$ is approximately $6.0 \cdot 10^{-9}$.

Figure 7 (b) shows the structure in the complex ε-plane for $s(\underline{x}, \varepsilon)$ based on the IQ RBF. We notice a couple of general differences between the structures based on the IQ RBF and MQ RBF. First, the IQ RBF leads to a slightly better conditioned linear system to solve. Thus, the approximate area of ill-conditioning is smaller. Second, the IQ basis function contains a pole, rather than a branch point, when $\varepsilon = \pm i/r$. Thus, for evaluation on the unit disk, there will be trivial poles (of unknown strengths) on the imaginary ε-axis that can never get closer to the origin than $\pm\frac{i}{2}$. For our 62 point distribution and for an evaluation point $\underline{x}$ that does not correspond to any of the data points, there could be up to $2 \cdot 62 = 124$ trivial poles on the imaginary axis. If we combine these with the non-trivial poles that arise from singularities in the $A(\varepsilon)$ matrix, this will be too many for the Contour-Padé algorithm to "pick up". So, as in the MQ case, the choice of our contour is limited by $1/D$, where D is the maximum distance between the points (e.g. $1/D = 1/2$ for evaluation on the unit disk). One common feature we have observed in the structures of $s(\underline{x}, \varepsilon)$ for the IQ and MQ cases is that the location of

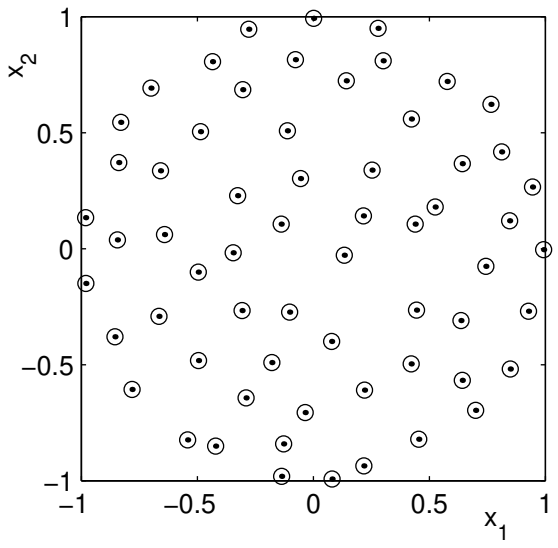


Figure 6: Distribution of 62 data points for use in the example from Section 6.

the poles due to singularities of the $A(\varepsilon)$ matrix are usually in similar locations (cf. the solid circles in Figure 7 (a) and (b)).

Figure 8 (b) compares the resulting RMS error as a function of ε when the IQ RBF approximation is computed directly via (6) and computed using the Contour-Padé algorithm. Again, we see that the direct computation becomes unstable when ε falls below approximately 0.21. This is well above the optimal value of approximately 0.122. Using the Contour-Padé algorithm, we find that the RMS error in the approximation at this value of ε is approximately $2.5 \cdot 10^{-12}$, whereas the RMS error at $\varepsilon = 0.21$ is approximately $2.5 \cdot 10^{-9}$.

Figure 7 (c) shows the structure in the complex ε-plane for $s(\underline{x}, \varepsilon)$ based on the GA RBF. It differs significantly from the structures based on the IQ and MQ RBFs. The first major difference is that the GA RBF possesses no singularities in the finite complex ε-plane (it has an essential singular point at $\varepsilon = \infty$). Thus, the contour we choose is not limited by the maximum distance between the points. However, the GA RBF grows as ε moves farther away from the real axis. Thus, the contour we choose for evaluating $s(\underline{x}, \varepsilon)$ is limited by the ill-conditioning that arises for large imaginary values of ε. This limiting factor has signficantly less impact than the "maximum distance" limiting factor for the MQ and IQ RBFs, and makes the Contour-Padé algorithm based on GA RBF able to handle larger data sets (for example, it can easily handle approximations based on 100 data points in the unit disk when the computations are done in standard 64-bit floating point). Indeed, Figure 7 (c) shows that the contour we used for the GA RBF approximation is much farther away from the ill-conditioned region around $\varepsilon = 0$ than the corresponding contours for the MQ and IQ approximations. The second difference for the GA RBF is that it leads to a linear system that approaches ill-conditioning faster as ε approaches zero [3]. The final difference we note (from also looking at additional examples) is that the pole structure of $s(\underline{x}, \varepsilon)$ based on the GA RBF often differs quite significantly from those based on the MQ and IQ RBFs.

Figure 8 (c) compares the resulting RMS error as a function of ε when the GA RBF approximation is computed directly via (6) and computed using the Contour-Padé algorithm.

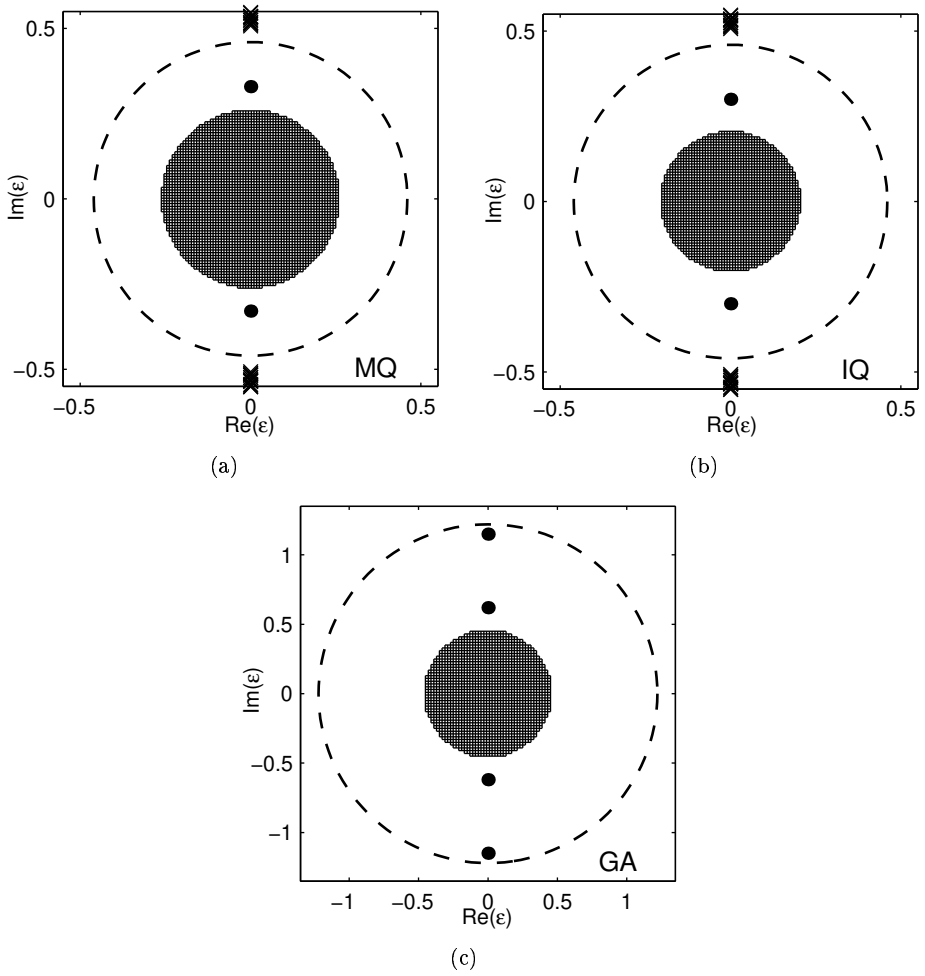
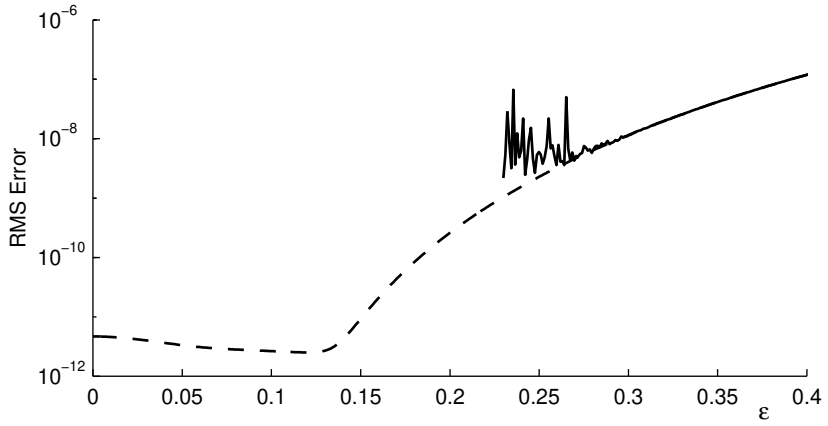
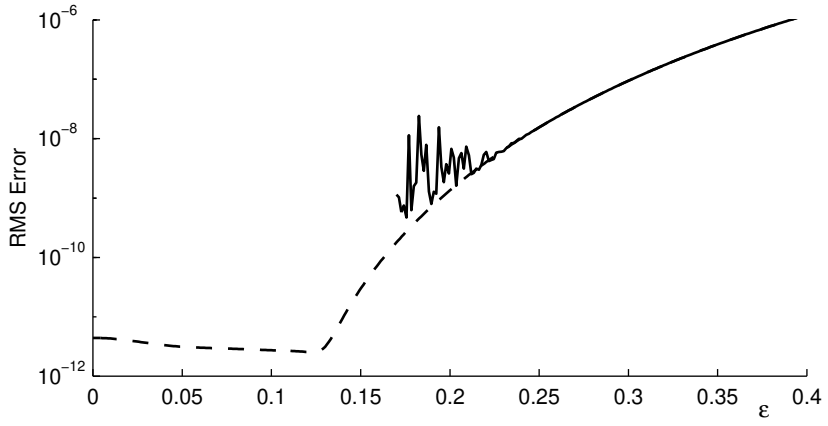


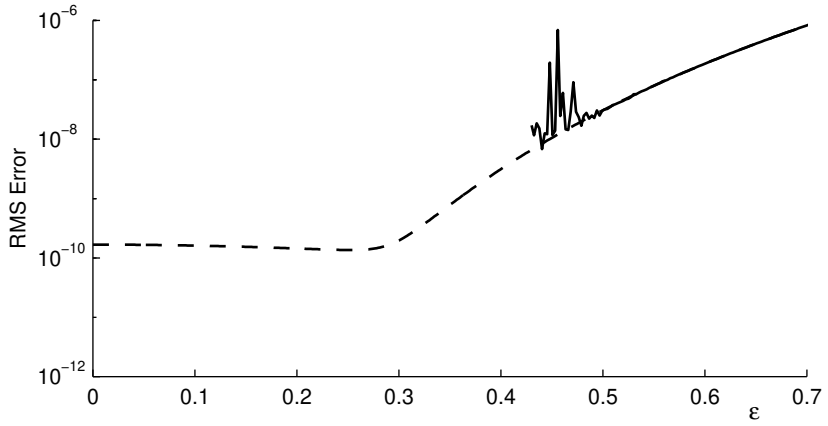
Figure 7: The structures of $s(\underline{x}, \varepsilon)$ in complex ε-plane for the 62 data points shown in Figure 6 in the case of (a) MQ RBF (b) IQ RBF and (c) GA RBF (note the different scale). The approximate region of ill-conditioning is marked with a line pattern, the poles are marked with solid circles, and singularites due to the basis functions themselves (i.e. branch points for the MQ RBF and poles for the IQ RBF) are marked with with $\times$'s. The dashed lines indicate the contours that were used for computing $s(\underline{x}, \varepsilon)$ for each of the three cases.



(a) MQ RBF



(b) IQ RBF



(c) GA RBF

Figure 8: The RMS error in the (a) MQ, (b) IQ, and (c) GA RBF approximations $s(\underline{x}, \varepsilon)$ of (3). The solid line shows the error when $s(\underline{x}, \varepsilon)$ is computed using (6) and the dashed line shows the error when $s(\underline{x}, \varepsilon)$ is computed using the Contour-Padé algorithm. Note the different scale for the GA RBF results.

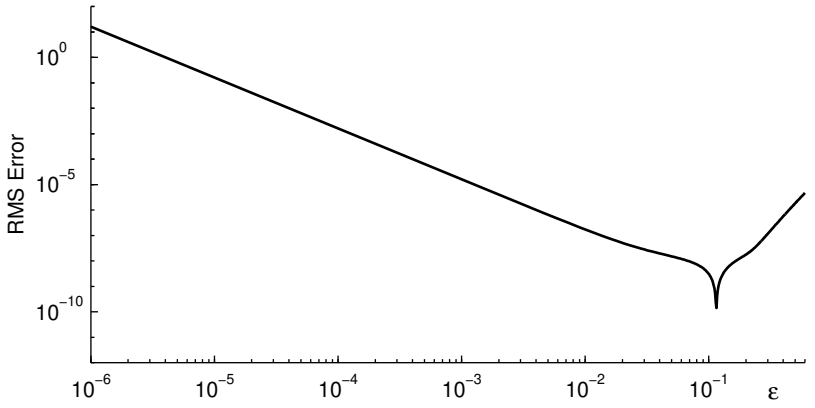


Figure 9: The RMS error in the MQ RBF approximation $s(\underline{x}, \varepsilon)$ of (3) for the case of a 5×5 equispaced Cartesian grid over $[0,1] \times [0,1]$.

The figure shows that instability in the direct method arises when ε falls below 0.48. Again, this is well above the optimal value of $\varepsilon = 0.253$. The Contour-Padé algorithm produces an RMS error of approximately $1.4 \cdot 10^{-10}$ at this value, whereas the RMS error at $\varepsilon = 0.48$ is approximately $2.0 \cdot 10^{-8}$.

We next explore a case where the limit of $s(\underline{x}, \varepsilon)$ as $\varepsilon \to 0$ fails to exist. As was reported in [9], the 5×5 equispaced Cartesian grid over $[0,1] \times [0,1]$ leads to divergence in $s(\underline{x}, \varepsilon)$ of the type $O(\varepsilon^{-2})$. To see how the Contour-Padé algorithm handles this situation, we consider the the 5×5 grid as our data points $\underline{x}_j$ and compute the MQ RBF approximation to (3) (although the choice of data values is irrelevant to the issue of convergence or divergence; as we know from (6) this depends only on the properties of the matrix $C(\varepsilon) = B(\varepsilon) \cdot A(\varepsilon)^{-1}$). Figure 9 shows a $\log - \log$ plot of RMS error where the MQ RBF approximation has been evaluated on a much denser grid over $[0,1] \times [0,1]$. In agreement with the high-precision calculations reported in [9], we again see a slow growth towards infinity for the interpolant. The reason is that this time there is a double pole right at the origin of the $\varepsilon-$plane (i.e. $\varepsilon = 0$ is not, in this case, a removable singularity). The Contour-Padé algorithm automatically handles this situation correctly, as Figure 9 shows.

To get a better understanding of how the interpolant behaves for this example, we use the algorithm to compute all the functions $d_k(\underline{x})$ in the small ε-expansion

$$s(\underline{x}, \varepsilon) = d_{-2}(\underline{x})\varepsilon^{-2} + d_0(\underline{x}) + d_2(\underline{x})\varepsilon^2 + d_4(\underline{x})\varepsilon^4 + \ldots \tag{12}$$

Figure 10 displays the first 6 $d_k(\underline{x})$-functions over the unit square. Note the small vertical scale on the figure for the $d_{-2}(\underline{x})$ function. This is consistent with the fact that divergence occurs only for small values of ε (cf. Figure 9). Each surface in Figure 10 shows markers (solid circles) at the 25 data points. The function $d_0(\underline{x})$ exactly matches the input function values at those points (and the other functions are exactly zero there). It also gives very accurate approximation to the actual function; the RMS error is $1.27 \cdot 10^{-8}$.

We omit the results for the IQ and GA RBF interpolants for this example, but note that the IQ also leads to divergence in $s(\underline{x}, \varepsilon)$ of the type $O(\varepsilon^{-2})$ (as reported in [9]), whereas the GA RBF actually leads to convergence.

We conclude this section with some additional comments about other techniques we tried

related to computing the interpolant for small values of ε.

It is often useful (and sometimes necessary) to augment the RBF interpolant (1) with low order polynomial terms (see for example [12]). The addition of these polynomial terms gives the RBF interpolation matrix (a slightly modified version of the $A(\varepsilon)$ matrix found in (4)) certain desirable properties, e.g. (conditional) positive or negative definiteness [10]. The Contour-Padé algorithm can—without any change—be used to compute the RBF interpolant also with the inclusion of these polynomial terms. We have found, however, that the behavior of the interpolant is not significantly affected by such variations. For example, we found that the pole structure of $s(\underline{x}, \varepsilon)$ is not noticeably affected, and there is no significant gain in accuracy at the "optimal" ε value (however, for larger values of ε, there can be some gains).

Since the RBF interpolant can usually be described for small values of ε by (12) but without the ε^{-2} term, one might consider using Richardson/Romberg extrapolation at larger values of ε to obtain the interpolant for $\varepsilon = 0$. However, this idea is not practical. Such extrapolation is only effective if the first few expansion terms strongly dominate the later ones. This would only be true if we are well inside the expansion's radius of convergence. As Figures 4 and 8 indicate, this would typically require computations at ε values that are too small for acceptable conditioning.

7 Concluding remarks

The shape parameter ε in RBF interpolation plays a significant role in the accuracy of the interpolant. The highest accuracy is often found for values of ε that make the direct method of computing the interpolant suffer from severe ill-conditioning. In this paper we have presented an algorithm that allows stable computation of RBF interpolants for all values of ε, including the limiting case (if it exists) when the basis functions become perfectly flat. This algorithm has also been successfully used in [13] for computing RBF based solutions to elliptic PDEs for the full range of ε-values.

The key to the algorithm lies in removing the restriction that ε be a real parameter. By allowing ε to be complex, we not only obtain a numerically stable algorithm, but we also gain a wealth of understanding about the interpolant, and we can use powerful tools to anlayze it, such as Cauchy integral formula, contour integration, Laurent series, and Padé approximations.

References

[1] R. Franke. Scattered data interpolation: tests of some methods. *Math. Comput.*, 38:181–200, 1982.

[2] W.R. Madych. Miscellaneous error bounds for multiquadric and related interpolants. *Comput. Math. Appl.*, 24:121–138, 1992.

[3] R. Schaback. Error estimates and condition numbers for radial basis function interpolants. *Adv. Comput. Math.*, 3:251–264, 1995.

[4] R. E. Carlson and T. A. Foley. The parameter R^2 in multiquadric interpolation. *Comput. Math. Appl.*, 21:29–42, 1991.

[5] T. A. Foley. Near optimal parameter selection for multiquadric interpolation. *J. Appl. Sci. Comput.*, 1:54–69, 1994.

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

[7] M. D. Buhmann and N. Dyn. Spectral convergence of multiquadric interpolation. In *Proceedings of the Edinburgh Mathematical Society*, volume 36, pages 319–333, Edinburgh, 1993.

[8] B. J. C. Baxter. On the asymptotic behaviour of the span of translates of the multiquadric $\phi(r) = (r^2 + c^2)^{1/2}$ as $c \to \infty$. *Comput. Math. Appl.*, 24:1–6, 1994.

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

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

[11] C. M. Bender and S. A. Orszag. *Advanced Mathematical Methods for Scientists and Engineers*. McGraw-Hill, 1978.

[12] I. R. H. Jackson. Radial basis functions: a survey and new results. Report no. DAMTP NA16, University of Cambridge, 1988.

[13] E. Larsson and B. Fornberg. A numerical study of some radial basis function based solution methods for elliptic PDEs. *Comput. Math. Appl.* To appear.

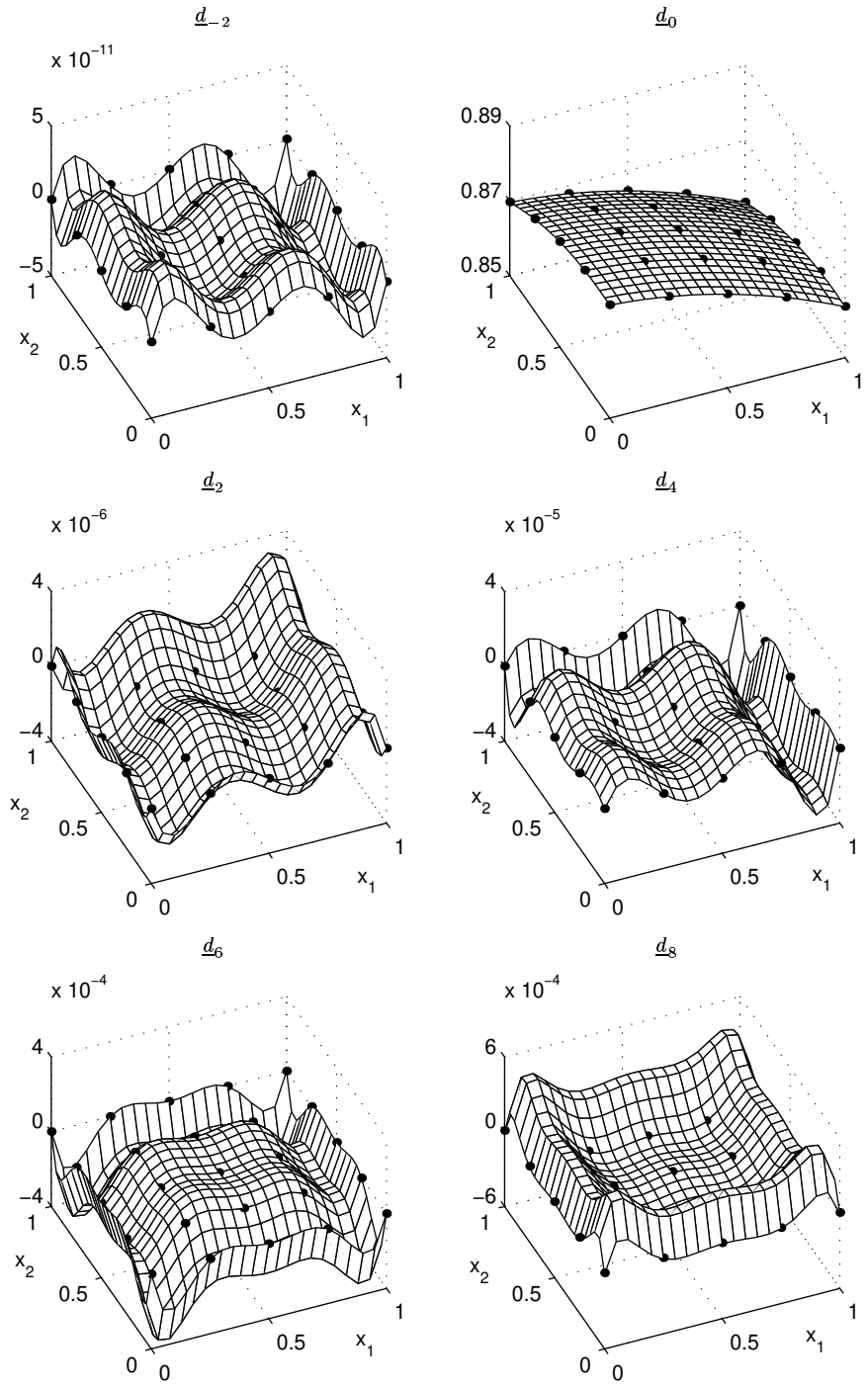


Figure 10: The first 6 terms from the expansion (12) of $s(\underline{x}, \varepsilon)$. The solid circles represent the 5×5 equi-spaced Cartesian grid that served as the input data points for the interpolant.

APPENDIX C:
SOME OBSERVATIONS REGARDING INTERPOLANTS IN THE LIMIT OF FLAT RADIAL BASIS FUNCTIONS

BENGT FORNBERG*, GRADY WRIGHT†, AND ELISABETH LARSSON‡

Abstract. Radial basis functions (RBFs) form a primary tool for multivariate interpolation. Some of the most commonly used radial functions feature a shape parameter, allowing them to vary from being nearly flat (ε small) to sharply peaked (ε large). The former limit can be particularly accurate when interpolating a smooth function based on scattered data. This study discusses theoretical and computational aspects of the $\varepsilon \to 0$ limit, and includes the conjecture that Gaussian RBF interpolants will never diverge in this limit.

Key words. Radial basis functions, RBF, multivariate interpolation.

AMS subject classifications. 41A10, 41A63, 41A58, 65D05, 65F22, 65F40

1. Introduction. When collocating n pieces of data in one dimension, one first chooses some set of basis functions $\psi_k(x)$, and then determines expansion coefficients λ_k such that the linear combination

$$s(x) = \sum_{k=1}^{n} \lambda_k \psi_k(x)$$

satisfies all the constraints. For many choices of $\psi_k(x)$, interpolation is guaranteed to be non-singular whenever the data points are distinct. With certain point distributions, it may furthermore be possible to choose basis functions that possess some orthogonality properties, e.g., Fourier modes on a periodic interval, or Chebyshev polynomials on a finite interval. In more than one dimension, the situation is very different. There no longer exist any basis functions $\psi_k(\underline{x})$ so that non-singularity for interpolation can be guaranteed for more than $n = 1$ data point [1]. The RBF approach circumvents this problem by following a somewhat different strategy. Instead of using a sequence of (typically increasingly oscillatory) basis functions that are independent of the data point locations $\underline{x}_k$, one uses instead translates of one single non-oscillatory function $\phi(\|\underline{x}\|)$:

$$s(\underline{x}) = \sum_{k=1}^{n} \lambda_k \phi(\|\underline{x} - \underline{x}_k\|) \ , \tag{1.1}$$

where $\|\cdot\|$ is the standard Euclidean vector norm. Table 1.1 shows a few of the many choices available for $\phi(r)$. Existence and uniqueness of the interpolants $s(\underline{x})$ are discussed for example in [2, 3, 4]. For all three choices of smooth basis functions listed (IQ, GA, MQ), these are ensured for arbitrary point distributions. To ensure these for the piecewise smooth basis functions listed, (1.1) may require some modifications [2].

*University of Colorado, Department of Applied Mathematics, 526 UCB, Boulder, CO 80309, USA (fornberg@colorado.edu). The work was supported by NSF grants DMS-9810751 (VIGRE), DMS-0073048, and by a Faculty Fellowship from University of Colorado at Boulder.

†University of Colorado, Department of Applied Mathematics, 526 UCB, Boulder, CO 80309, USA (wrightg@colorado.edu). The work was supported by NSF grant DMS-9810751 (VIGRE).

‡Uppsala University, Information Technology, Department of Scientific Computing, Box 337, SE-751 05 Uppsala, Sweden (bette@it.uu.se). The work was supported by a postdoctoral grant from STINT, The Swedish Foundation for International Cooperation in Research and Higher Education, and by a grant from The Swedish Research Council.

Type of basis function	$\phi(r)$
Piecewise smooth RBFs	
Piecewise polynomial (R_n)	$\|r\|^n$, n odd
Thin Plate Spline (TPS_n)	$\|r\|^n \ln r$, n even
Infinitely smooth RBFs	
Multiquadric (MQ)	$\sqrt{1+(\varepsilon r)^2}$
Inverse quadratic (IQ)	$\frac{1}{1+(\varepsilon r)^2}$
Gaussian (GA)	$e^{-(\varepsilon r)^2}$

TABLE 1.1
Some commonly used radial basis functions

The piecewise smooth $\phi(r)$, such as cubics and TPS will, as the number of data points is increased, give an algebraic rate of convergence (to a smooth function). The rate reflects the severity of the irregularity of $\phi(r)$ at the origin, and it typically increases with the number of space dimensions [5]. In contrast to this, the infinitely smooth RBFs often lead to spectral convergence (when proper attention is paid to boundary effects), i.e. $O(e^{-\text{const.}/h})$ (or $O(e^{-\text{const.}/h^2})$ in case of GA RBFs) where h is a 'typical' distance between neighboring data locations [6, 7, 8].

The value of the shape parameter ε in the smooth RBFs will influence the constants in this estimate. Many studies have been devoted to experimentally establishing suitable values of ε for different situations [9, 10, 11]. Although small but non-zero values of ε usually are optimal, the limit of flat radial functions ($\varepsilon \to 0$) has recently been found to have a number of intriguing features:

- For arbitrarily spaced data in 1-D, the limiting interpolant usually agrees with Lagrange's interpolation polynomial [12]. In higher-D, the limiting interpolant (when it exists) will again be a low degree multivariate polynomial. This means that RBFs can be a tool for generalizing, to irregular grids and domains, the 'classical' spectral methods (which typically are based on 1-D high-order polynomial interpolants, cf. [13]).
- Small values of ε have been found to yield very accurate results when interpolating smooth functions [14], solving elliptic PDEs with RBFs [15], and approximating data on low-dimensional manifolds within high-dimensional spaces [16].
- The direct method of solving for the RBF interpolant via the expansion coefficients λ_k in (1.1) becomes extremely ill-conditioned as $\varepsilon \to 0$ [7]. However, recently a numerically stable algorithm has been found that largely overcomes this ill-conditioning problem and allows for the stable computation of the RBF interpolants for the full range of ε [14]. Although the present algorithm appears to be limited to relatively small data sets, it still demonstrates that the ill conditioning is not in any way intrinsic to RBF interpolation, but only an artifact of certain implementations. A perfectly stable algorithm for any number of points may very well be feasible. In any case, RBF interpolants based on up to around 100 data points (in 2-D; more in higher-D) can at present be explored numerically for all values of ε, including in the limit of $\varepsilon \to 0$.

In very special cases divergence can occur when $\varepsilon \to 0$, as was first noted in [12] for a case when all the data points were given on a finite Cartesian grid. However, for randomly scattered data, there has never been found an instance in which the RBF interpolant fails to exist in the limit of $\varepsilon \to 0$. One of the goals of this paper is to try to shed more light on the nature of such exceptional situations. One key tool for that is the simple closed-form expression for the

RBF interpolant to cardinal data (equal to one at one node point, and zero at the others) that is given in Section 2. This formula is used to determine the interpolants in the examples in Section 3. These examples cause us to become particularly interested in the Gaussian (GA) RBFs, for which we formulate a conjecture. In Section 4, we relate the observed features of RBF interpolants to those of multivariate interpolation polynomials. We also summarize some new theoretical results regarding limiting interpolants for scattered data, which will be presented in more detail separately [17]. Although we primarily are interested in real values of ε, considering complex values of ε not only provides additional insights, but also leads to the only computational algorithm that is available for $\varepsilon \to 0$ studies [14]. In Section 5, we very briefly review this algorithm, and we then use it to re-visit and extend some of the examples from Section 3. An often considered variation of (1.1) is $s(\underline{x}) = \alpha + \sum_{k=1}^{n} \lambda_k \phi(\|\underline{x} - \underline{x}_k\|)$ together with the constraint $\sum_{k=1}^{n} \lambda_k = 0$. Our brief study of this in Section 6 does not suggest that it offers any particular advantages in the limit of $\varepsilon \to 0$. The final Section 7 contains some concluding remarks.

2. Closed-form expression for the RBF interpolant. With data values f_k at locations $\underline{x}_k$, $k = 1, 2, \ldots, n$, the RBF interpolant becomes

$$s(\underline{x}) = \sum_{k=1}^{n} \lambda_k \, \phi\left(\|\underline{x} - \underline{x}_k\|\right) \ . \tag{2.1}$$

The coefficients λ_k can be obtained by solving the linear system

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \ , \tag{2.2}$$

where the entries of the matrix A are $A_{i,j} = \phi\left(\left\|\underline{x}_i - \underline{x}_j\right\|\right), i = 1, \ldots, n, \ , j = 1, \ldots, n$. As mentioned in the introduction, this linear system is guaranteed to be non-singular for the IQ, GA, and MQ RBFs (assuming the data locations are unique). For some other choices of $\phi(r)$, it may be necessary to augment (2.1) with some low order polynomial terms and additional constraints in order to guarantee that (2.2) is non-singular (see for example [2]).

THEOREM 2.1. *For cardinal data* $f_k = \begin{cases} 1 & \text{if } k = 1 \\ 0 & \text{otherwise} \end{cases}$, *the RBF interpolant of the form (2.1) becomes*

$$s(\underline{x}) = \frac{\det \begin{bmatrix} \phi(\|\underline{x} - \underline{x}_1\|) & \phi(\|\underline{x} - \underline{x}_2\|) & \cdots & \phi(\|\underline{x} - \underline{x}_n\|) \\ \phi(\|\underline{x}_2 - \underline{x}_1\|) & \phi(\|\underline{x}_2 - \underline{x}_2\|) & \cdots & \phi(\|\underline{x}_2 - \underline{x}_n\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\underline{x}_n - \underline{x}_1\|) & \phi(\|\underline{x}_n - \underline{x}_2\|) & \cdots & \phi(\|\underline{x}_n - \underline{x}_n\|) \end{bmatrix}}{\det \begin{bmatrix} \phi(\|\underline{x}_1 - \underline{x}_1\|) & \phi(\|\underline{x}_1 - \underline{x}_2\|) & \cdots & \phi(\|\underline{x}_1 - \underline{x}_n\|) \\ \phi(\|\underline{x}_2 - \underline{x}_1\|) & \phi(\|\underline{x}_2 - \underline{x}_2\|) & \cdots & \phi(\|\underline{x}_2 - \underline{x}_n\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\underline{x}_n - \underline{x}_1\|) & \phi(\|\underline{x}_n - \underline{x}_2\|) & \cdots & \phi(\|\underline{x}_n - \underline{x}_n\|) \end{bmatrix}} \tag{2.3}$$

Proof. Solving (2.2) by Cramer's rule and substituting into (2.1) leads, in just a few lines of algebra, to (2.3). However, the easiest demonstration is probably by inspection. By

expanding the determinant in the numerator along its first row, we see that the expression for $s(\underline{x})$ indeed becomes of the form of (2.1). Furthermore, it will evaluate to one for $\underline{x} = \underline{x}_1$ (since then the two determinants become equal) and to zero for $\underline{x} = \underline{x}_k, k \neq 1$ (since then two rows in the determinant in the numerator become equal). □

One immediate consequence of this result is the following:

THEOREM 2.2. *If* $\lim_{\varepsilon \to 0} s(\underline{x})$ *exists, it will be a (multivariate) finite degree polynomial in* $\underline{x}$.

Proof. If we expand $\phi(\|\underline{x} - \underline{x}_k\|)$ in powers of ε^2, the coefficient for ε^{2m} will be a polynomial of degree (at most) $2m$ in the components of $\underline{x}$. The same will therefore hold for the determinant in the numerator of (2.3), and the ratio in (2.3) will be of the form

$$s(\underline{x}) = \frac{\varepsilon^{2p}\{\text{pol. degree } 2p\} + \varepsilon^{2p+2}\{\text{pol. degree } 2p{+}2\} + \dots}{\varepsilon^{2q}\{\text{constant}\} + \varepsilon^{2q+2}\{\text{constant}\} + \dots}$$

where p and q are positive integers. Since $s(x_1) = 1$, $p > q$ is impossible. If $p < q$, the limit fails to exist. Otherwise (i.e. when $p = q$) it will become a polynomial of degree (at most) $2p$ in the components of $\underline{x}$.□

Theorem 2.2 appears to have been discovered independently a number of times during the last decade or so. However, we have been unable to locate it in any previous reference. It was however shown in [12] that, in 1-D and subject to some minor constraint on $\phi(r)$, the limit is the lowest order interpolation polynomial, i.e. of degree $n-1$ in case of n data points (and that, failing these constraints, the limit would still be of polynomial type, but that the degree could be higher). The situation for higher-D can be considerably more complex, as will be discussed below (and analyzed further in [17]).

3. A collection of examples with closed-form solutions for the $\varepsilon \to 0$ limit. The first several examples in this section concern the situation when increasingly many points are located along a straight line. In most of these cases, closed-form analysis is possible, offering key insights and motivation for the more general discussion in Section 4 (where we examine polynomial unisolvency and interpolants for scattered points in more dimensions).

3.1. Three points along a straight line: Evaluation along the line. Let the three points be located at $\{x_1, x_2, x_3\}$ and the corresponding data values be $\{1, 0, 0\}$ (The results become equivalent for $\{0, 1, 0\}$ and $\{0, 0, 1\}$ and, since the interpolation procedure is linear, also for arbitrary data). With a radial function

$$\phi(r) = a_0 + a_1(\varepsilon r)^2 + a_2(\varepsilon r)^4 + \dots$$

we get

$$\det \begin{bmatrix} \phi(|x - x_1|) & \phi(|x - x_2|) & \phi(|x - x_3|) \\ \phi(|x_2 - x_1|) & \phi(|x_2 - x_2|) & \phi(|x_2 - x_3|) \\ \phi(|x_3 - x_1|) & \phi(|x_3 - x_2|) & \phi(|x_3 - x_3|) \end{bmatrix} =$$

$$= 2a_1(a_1^2 - 6a_0a_2)(x - x_2)(x - x_3)(x_1 - x_2)(x_1 - x_3)(x_2 - x_3)^2\varepsilon^6 + O(\varepsilon^8)$$

and

$$\det \begin{bmatrix} \phi(|x_1 - x_1|) & \phi(|x_1 - x_2|) & \phi(|x_1 - x_3|) \\ \phi(|x_2 - x_1|) & \phi(|x_2 - x_2|) & \phi(|x_2 - x_3|) \\ \phi(|x_3 - x_1|) & \phi(|x_3 - x_2|) & \phi(|x_3 - x_3|) \end{bmatrix} =$$

$$= 2a_1(a_1^2 - 6a_0a_2)(x_1 - x_2)^2(x_1 - x_3)^2(x_2 - x_3)^2\varepsilon^6 + O(\varepsilon^8).$$

On the assumption that $a_1 \neq 0$ and $a_1^2 - 6a_0a_2 \neq 0$, the ratio becomes

$$s(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + O(\varepsilon^2),$$

i.e. we have recovered the Lagrange interpolation polynomial. According to the main theorem in [12], the same will be the case for any number of points along a line, as long as certain inequalities hold for the Taylor coefficients a_k of the radial function. The two inequalities encountered here are two of an infinite set that is explicitly given in [12]. Although a firm proof is still lacking, current evidence strongly suggest that all of these are satisfied, for example by MQ, IQ, and GA RBFs. We will in the following assume that these inequalities hold when we refer to this main theorem of [12].

3.2. Three points along a straight line: Evaluation off the line. When evaluating the interpolant in the previous example at a location (x, y), the first determinant becomes

$$\det \begin{bmatrix} \phi(\sqrt{(x-x_1)^2+y^2}) & \phi(\sqrt{(x-x_2)^2+y^2}) & \phi(\sqrt{(x-x_3)^2+y^2}) \\ \phi(|x_2 - x_1|) & \phi(|x_2 - x_2|) & \phi(|x_2 - x_3|) \\ \phi(|x_3 - x_1|) & \phi(|x_3 - x_2|) & \phi(|x_3 - x_3|) \end{bmatrix} =$$

$$= 2a_1 \left[(a_1^2 - 6a_0a_2)(x - x_2)(x - x_3) + (a_1^2 - 2a_0a_2)y^2\right] \cdot$$

$$\cdot (x_1 - x_2)(x_1 - x_3)(x_2 - x_3)^2 \, \varepsilon^6 \; + \; O(\varepsilon^8)$$

producing the interpolant

$$s(x, y) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + \frac{a_1^2 - 2a_0a_2}{a_1^2 - 6a_0a_2} \, \frac{y^2}{(x_1 - x_2)(x_1 - x_3)} + O(\varepsilon^2). \tag{3.1}$$

For $y = 0$, i.e. along the x-axis, we recover the previous result. However, a y^2-term is now also present, with a coefficient that depends on the choice of RBF. The factor $\frac{a_1^2 - 2a_0a_2}{a_1^2 - 6a_0a_2}$ takes the values $\frac{1}{2}$, $\frac{1}{5}$, and 0 for MQ, IQ, and GA respectively. Thus the $\varepsilon \to 0$ limits are now different for the different RBFs, but they still exist in all cases.

3.3. Three points not on a line. With the three points located at (x_k, y_k), $k = 1, 2, 3$, the ratio of the two determinants becomes

$$s(x, y) = \frac{4a_1^2(2A)(x\ (y_3 - y_2) - y\ (x_3 - x_2) + x_3y_2 - x_2y_3)\ \varepsilon^4 + O(\varepsilon^6)}{4a_1^2(2A)^2\ \varepsilon^4 + O(\varepsilon^6)} \tag{3.2}$$

where

$$2A = x_2y_1 - x_3y_1 - x_1y_2 + x_3y_2 + x_1y_3 - x_2y_3\ ,$$

i.e. $|A|$ is the area of the triangle spanned by the three point locations. When the three points are not collinear (i.e. when $A \neq 0$, and also assuming $a_1 \neq 0$), the limiting interpolant will be the plane that fits the data. As the points become increasingly collinear, the tilt of the plane increases without bound. When the points are collinear, both the coefficients for ε^4 in (3.2) vanish, and the $\varepsilon \to 0$ limit will instead follow from the ε^6 coefficients, as described in the two preceding examples. The limit thus behaves discontinuously with respect to the point locations.

3.4. Five or more points along a straight line. It turns out that increasing from three to four points along a line (say, the x-axis) offers no new phenomenon. For five points along a line, the result along the same line (like for any number of points) becomes the Lagrange interpolation polynomial. However, evaluating at a location (x, y) off the x-axis now gives

$$s(x,y) = \frac{4y^2}{(x_1-x_2)(x_1-x_3)(x_1-x_4)(x_1-x_5)} \cdot \cdot \frac{(a_1a_2^2 - 3a_1^2a_3 + 3a_0a_2a_3)}{(6a_2^3 + 225a_0a_3^2 + 70a_1^2a_4 - 30a_1a_2a_3 - 420a_0a_2a_4)} \frac{1}{\varepsilon^2} + O(1) \tag{3.3}$$

(assuming $2a_2^2 - 5a_1a_3 \neq 0$; both this assumption and the denominator above being non-zero belong to the set of inequalities discussed above). The ratio involving the coefficients a_k, $k = 1, \ldots, 4$ in (3.3) takes for MQ, IQ, and GA the values $\frac{1}{28}$, $\frac{1}{149}$ and 0 respectively. If for example $\phi(r) = (1+(\varepsilon r)^2)^{\beta/2}$ (an 'unconditionally positive definite' case with non-singularity of the interpolant guaranteed for any $\beta < 0$, cf. [2, 3]), the assumption $2a_2^2 - 5a_1a_3 \neq 0$ becomes $\beta \neq 0$, $\beta \neq 2$, $\beta \neq 7$, and the coefficient ratio becomes always non-zero: $-3/(\beta^3 - 19\beta^2 + 99\beta - 165)$, i.e. divergence as $\varepsilon \to 0$.

Increasing to 7, 9, ... points, we typically get divergence like $O(1/\varepsilon^4)$, $O(1/\varepsilon^6)$, etc. For non-divergence, increasingly numerous and intricate requirements on the coefficients a_k need to hold. GA are remarkable in satisfying them all, as follows from the theorem below:

THEOREM 3.1. *With any number of data points along a straight line, GA interpolants will not diverge as $\varepsilon \to 0$ when evaluated anywhere on or off the line.*

Proof. Along the x-axis, (i.e. $y = 0$), the GA interpolant takes the form $s(x,\varepsilon) = \sum_k \lambda_k e^{-\varepsilon^2(x-x_k)^2}$. At (x, y), the value becomes $r(x,y,\varepsilon) = \sum_k \lambda_k e^{-\varepsilon^2((x-x_k)^2+y^2)} = e^{-\varepsilon^2 y^2} s(x,\varepsilon)$. Both of these factors remain bounded as $\varepsilon \to 0$ (obviously for the first one, and the second one converges to Lagrange's interpolation polynomial, according to the main theorem in [12]). Therefore, the product also remains bounded. □

Both numerical and some analytical evidence suggest that certain other RBFs share the property of GA interpolants shown in Theorem 3.1, e.g. $\phi(r) = J_0(\varepsilon r)$ and $\phi(r) = \frac{\sin(\varepsilon r)}{\varepsilon r}$. Since these functions are oscillatory, they have seldom been considered for practical RBF work (with one reason for this being that oscillatory RBFs can never provide non-singular interpolation for all data sets in all dimensions). However, with finite-sized data sets and small ε, the oscillations would never be seen, and it could be that oscillatory radial functions deserve some further considerations.

In the doubly infinite case of $x_k = k$, $k = -\infty, \ldots, -1, 0, 1, \ldots, \infty$, the situation is again different—there is no divergence (as $\varepsilon \to 0$) off the line for any of the smooth RBF choices [18].

3.5. Some generalizations to higher dimensions. The result in Theorem 3.1 can be extended as follows:

THEOREM 3.2. *In the case when the data points are laid out in a finite rectangular lattice (in any number of dimensions), GA interpolants will not diverge as $\varepsilon \to 0$.*

Proof. For notational simplicity, we consider the 2-D case. Let the lattice be $\{x_i, y_j\}$, $i = 1, \ldots m$, $j = 1, \ldots n$. It suffices to show the result for cardinal data, i.e. if for some fixed i and j it holds that

$$f = \begin{cases} 1 & \text{when } x = x_i \text{ and } y = y_j \\ 0 & \text{otherwise} \end{cases} .$$

Dimension of hyperplane: $d =$	1	2	3	4	5	6	7	8
Lowest number of points to feature divergence off the hyperplane; $N_d =$	5	11	21	36	57	85	121	166

TABLE 3.1

Lowest number of scattered data points in a hyperplane to cause divergence off the plane

Consider the 1-D interpolants

$$r(x,\varepsilon) = \sum_{k=1}^{m} \lambda_k e^{-\varepsilon^2 (x-x_k)^2} \quad \text{satisfying} \quad r(x,\varepsilon) = \begin{cases} 1 & x = x_i \\ 0 & \text{otherwise} \end{cases}$$

$$s(y,\varepsilon) = \sum_{l=1}^{n} \mu_l e^{-\varepsilon^2 (y-y_l)^2} \quad \text{satisfying} \quad s(y,\varepsilon) = \begin{cases} 1 & y = y_j \\ 0 & \text{otherwise} \end{cases}$$

The product $r(x,\varepsilon) \cdot s(y,\varepsilon) = \sum_{k=1}^{m} \sum_{l=1}^{n} \lambda_k \mu_l e^{-\varepsilon^2 (x-x_k)^2 - \varepsilon^2 (y-y_l)^2}$ satisfies all that is required of the 2-D GA interpolant, and is therefore identical to it. Since both of the factors $r(x,\varepsilon)$ and $s(y,\varepsilon)$ remain bounded as $\varepsilon \to 0$ (by the main theorem in [12]), so does their product. □

Equation (3.3) showed that, for five points along a line ($d = 1$), divergence typically occurs when the interpolant is evaluated off the line. Similarly, for scattered points in a plane ($d = 2$), eleven points usually leads to divergence when the interpolant is evaluated off that plane. The data in Table 3.1, obtained with the Contour- Padé algorithm described in Section 5, suggests that $N_d - 1 = \binom{d+3}{3}$.

All instances of divergence that we have encountered occur in situations where 'polynomial unisolvency' (to be defined later) fails. Heuristically, this becomes the case when the data points are located in such a way that multivariate polynomial interpolation leaves some low order coefficient(s) completely undetermined. For example, with points only along the x-axis, coefficients for powers of y are undetermined. Similarly, if we place all the points along a section of a parabola, some other low order polynomial coefficients will be undetermined. This is illustrated next.

3.6. Points placed along a parabola. We let the locations for n points be $x_k = \frac{k-1}{n-1}$, $y_k = x_k^2$, $k = 1, 2, \ldots, n$. This leaves a polynomial interpolant undetermined with respect to several low order polynomials, such as $y - x^2$, $x(y - x^2)$, $y(y - x^2)$, etc. With cardinal data (equal to one at the first point and zero at the remaining ones) we get in this case different limits starting when $n = 4$ (one step later than for points along a line). The leading powers of ε in the expansions in the numerator and denominator of (2.3) increase quite rapidly with n. Table 3.2 illustrates that, and also gives the value of the interpolants in the $\varepsilon \to 0$ limit, evaluated at the point (0,1). For $n = 8$, both MQ and IQ feature a numerator starting with ε^{30} and denominator starting with ε^{32}, producing divergence according to $\frac{117649}{23040} \frac{1}{\varepsilon^2} + \frac{72202965}{65536} + O(\varepsilon^2)$ and $\frac{117649}{127080} \frac{1}{\varepsilon^2} + \frac{643338441829}{538310880} + O(\varepsilon^2)$ respectively. The GA interpolant remains bounded; both leading terms are $O(\varepsilon^{32})$, and the limit value is $\frac{6864}{5}$. Numerical evidence, to be given in Section 5, shows that for n higher still, MQ and IQ interpolants again diverge (when evaluated away from the parabola), whereas no case of divergence was seen for GA interpolants.

We conclude this section with a conjecture:

CONJECTURE 3.3. *Gaussian (GA) RBF interpolants will never diverge as* $\varepsilon \to 0$.

n	2	3	4	5	6	7
Leading power of ε in numerator = leading power in denominator	2	4	8	12	18	24
Value of interpolant at (0,1)						
MQ	$\frac{1}{2}$	3	$\frac{69}{10}$	$\frac{89}{3}$	$\frac{12253}{176}$	$-\frac{8043}{40}$
IQ	$\frac{1}{2}$	3	$\frac{27}{4}$	$\frac{493}{15}$	$\frac{22575}{272}$	$\frac{6972}{25}$
GA	$\frac{1}{2}$	3	$\frac{20}{3}$	35	$\frac{189}{2}$	462

TABLE 3.2
Values of the interpolant at (0,1) in the example with points on a parabola

1-D	2-D	3-D
$a_0 \neq 0$	$a_0 \neq 0$	$a_0 \neq 0$
$a_1 \neq 0$	$a_1 \neq 0$	$a_1 \neq 0$
$6a_0a_2 - a_1^2 \neq 0$	$a_2 \neq 0$, $4a_0a_2 - a_1^2 \neq 0$	$a_2 \neq 0$, $10a_0a_2 - 3a_1^2 \neq 0$
$5a_1a_3 - 2a_2^2 \neq 0$	$a_3 \neq 0$, $9a_1a_3 - 4a_2^2 \neq 0$	$a_3 \neq 0$, $21a_1a_3 - 10a_2^2 \neq 0$

TABLE 4.1
Conditions on the expansion coefficients.

The strongest evidence in support of this conjecture comes from various experiments with the numerical algorithm that is described in Section 5.

4. Polynomial unisolvency, and some results for scattered points. An important concept in multivariate polynomial interpolation is unisolvency [1]. As we will see below, it also has some bearing on the RBF interpolation problem. The following theorem defines the concept and gives a necessary and sufficient condition.

THEOREM 4.1. *Let $\underline{x}_1$, $\underline{x}_2$, ..., $\underline{x}_n$ be n point locations, and let $p_1(\underline{x})$, $p_2(\underline{x})$, ..., $p_n(\underline{x})$ be n linearly independent polynomials. Then $\{\underline{x}_i\}$ is unisolvent with respect to $\{p_i\}$, i.e., there is a unique linear combination $\sum \beta_j p_j(\underline{x})$ which interpolates any data over the point set, if and only if det $(P) \neq 0$, where*

$$P = \begin{bmatrix} p_1(\underline{x}_1) & p_2(\underline{x}_1) & \cdots & p_n(\underline{x}_1) \\ p_1(\underline{x}_2) & p_2(\underline{x}_2) & \cdots & p_n(\underline{x}_2) \\ \vdots & \vdots & & \vdots \\ p_1(\underline{x}_n) & p_2(\underline{x}_n) & \cdots & p_n(\underline{x}_n) \end{bmatrix}.$$

The proof follows trivially from linear algebra arguments. The application of the theorem is easy in 1-D. With $p_j(x) = x^{j-1}$ we get the Vandermonde matrix, which is non-singular as soon as none of the data points coincide. In more than one space dimension, it is less obvious how to best determine whether a point set is unisolvent or not. However, using randomly scattered data points promotes unisolvency. Any regular features (such as the examples given in the previous section with points along a straight line or along a parabola) in a data set may lead to degeneration.

COROLLARY 4.2. *We can (also elementary) add that if det $(P) = 0$, then the nullspace of P will describe all the possible ambiguities in the resulting interpolant of the specified form.*

We know from Theorem 2.2 that if $\lim_{\varepsilon \to 0} s(\underline{x})$ exists, it is a finite order polynomial. It was proved in [12] that in 1-D, we recover the lowest order interpolating polynomial (under

some mild conditions on the expansion coefficients of the basis function, as briefly touched upon in Section 3). There are similar condition in more space dimensions. Table 4.1 gives the first four conditions in up to three space dimensions. Conditions of this type are needed for the proof of the theorem given below.

DEFINITION 4.3. *Let P_K be the set of all (multivariate) polynomials of degree $\leq K$.*

THEOREM 4.4. *If $\{p_i\}$ forms a basis for P_K and $\{\underline{x}_i\}$ is unisolvent with respect to $\{p_i\}$, then under some mild assumptions on the expansion coefficients of the radial basis function, the limiting RBF interpolant $\lim_{\varepsilon\to 0} s(\underline{x})$ is the unique interpolating polynomial of degree $\leq K$ to the given data.*

A proof and further discussion of the implications of the theorem can be found in [17] (an entirely different proof of the theorem can also be found in [19]). For now, we will only give a brief summary of some aspects relevant here. First of all, note that the number of data points must agree with the dimension of P_K for $\{p_i\}$ to be a basis. That is, in for example 2-D, $n = 1, 3, 6, 10, 15, 21, \ldots$ play a special role. If the unisolvency condition is not fulfilled, the interpolant may in the limit $\varepsilon \to 0$

- diverge,
- contain arbitrary elements from the nullspace of the matrix P,
- contain polynomial terms of higher degree than K.

EXAMPLE 4.5. *Interpolate $f(x, y) = x - y - 2xy - 2y^2$ at the six points $\underline{x}_i = (x_i, y_i) = \{(0,0),\ (0,\frac{1}{2}),\ (0,1),\ (1,0),\ (1,\frac{1}{2}),\ (1,1)\}$ with RBFs.* The natural choice of basis functions would be

$$\begin{aligned} &p_1 = 1, \\ &p_2 = x, \quad p_3 = y, \\ &p_4 = x^2, \quad p_5 = xy, \quad p_6 = y^2. \end{aligned}$$

This gives

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{4} \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{4} \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

This matrix is singular, and the nullspace of P is found to be $[0,\ 1,\ 0,\ -1,\ 0,\ 0]^T$. Therefore, the interpolant is undetermined with respect to any multiple of $1 \cdot p_2(\underline{x}) - 1 \cdot p_4(\underline{x}) = x - x^2 = x(1-x)$. If we try RBFs on this data set and let $\varepsilon \to 0$, we get

	$f(x, y) = x - y - 2xy - 2y^2$	original data
IQ:	$\frac{7}{5}x - y - \frac{2}{5}x^2 - 2xy - 2y^2$	
MQ:	$2x - y - x^2 - 2xy - 2y^2$	
GA:	$x - y - 2xy - 2y^2$	(recovers the original function)

We are getting different limits for all the three RBF types, but the differences are precisely of the type that was allowed to be undetermined according to the nullspace argument above. □

As shown in the examples from Section 3, with increasingly many points along a straight line or parabola, divergence of the IQ and MQ interpolants is preceeded by cases where the limiting interpolants are different. This result appears to be typical for many non-unisolvent

	d -	n - number of data points							
	dimension	2	3	5	10	20	50	100	200
Leading power of ε	1	2	6	20	90	380	2450	9900	39800
in both of the	2	2	4	12	40	130	570	1690	4940
determinants in (2.3)	3	2	4	10	30	90	360	980	2610
Leading inverse	1	2	4	8	18	38	98	198	398
power of ε in the	2	2	2	4	6	10	18	26	38
coefficients λ_k	3	2	2	4	4	6	10	14	18

TABLE 5.1
Powers of ε arising in cases of scattered data

cases. For example, if the points $(0, 1/4)$ and $(0, 3/4)$ are included in the above example, then both the IQ and MQ interpolants diverge like $O(\varepsilon^{-2})$ (the GA interpolant again converges to the original function).

5. Complex ε-plane considerations and the numerical Contour-Padé algorithm. The elements in the determinants in (2.3) are of size $O(1)$. Since expansions of the determinants for small ε typically start with some high power of ε, both matrices clearly become highly singular as $\varepsilon \to 0$. The matrix in the denominator is the same as the matrix A in (2.2). Thus, the coefficients λ_k must grow rapidly with decreasing ε. Since the sum in (2.1) is typically bounded, the sum must feature very severe cancellation of large quantities, and direct solution for $s(\underline{x})$ via (2.2) and (2.1) will be very ill conditioned. This heuristic argument can be made much more precise. In the case of scattered data, numerical evidence [17] strongly suggests the exponents that are shown in Table 5.1. These numbers are independent of the choice of MQ, IQ, or GA, and also hold for all other smooth RBFs that we have tested.

5.1. Numerical algorithm. The Contour-Padé algorithm in [14] is based on the fact that, in a complex ε-plane, the origin is a removable singularity (or, at worst, a low order pole) of the interpolant, which we now write as $s(\underline{x}, \varepsilon)$ in order to emphasize its dependence on ε. For a fixed $\underline{x}$, we compute $s(\underline{x}, \varepsilon)$ at equi-spaced ε-values around a circle centered at the origin in a complex ε-plane. Choosing the radius quite large, the direct approach using (2.2) and (2.1) works well. The next step is to take an FFT of these values. Some different cases will then arise:

- All negative Fourier coefficients vanish: The positive ones then provide the coefficients in the Taylor expansion

$$s(\underline{x}, \varepsilon) = s_0(\underline{x}) + \varepsilon^2 s_2(\underline{x}) + \varepsilon^4 s_4(\underline{x}) + \dots \tag{5.1}$$

  This expansion is then well suited for numerical computation of $s(\underline{x}, \varepsilon)$ for small values of ε (including $\varepsilon = 0$, when it just reduces to its first coefficient).
- Some negative Fourier coefficients are present: With a Padé procedure, we can find all the poles of $s(\underline{x}, \varepsilon)$ inside the computational circle. If there is no pole at the origin, we can either again express the interpolant in the form (5.1) or, if we want the radius of convergence to extend out to the computational circle and not just to the nearest pole, we can represent the answer as a Taylor series together with a rational function in ε. If there is a pole right at the origin, the only difference compared to the previous cases is that (5.1) will need to also include some term(s) with negative powers of ε. For example, if the origin is a pole of order four, (5.1) would need to be replaced with

$$s(\underline{x}, \varepsilon) = \frac{1}{\varepsilon^4} s_{-4}(\underline{x}) + \frac{1}{\varepsilon^2} s_{-2}(\underline{x}) + s_0(\underline{x}) + \varepsilon^2 s_2(\underline{x}) + \varepsilon^4 s_4(\underline{x}) + \dots \quad . \tag{5.2}$$

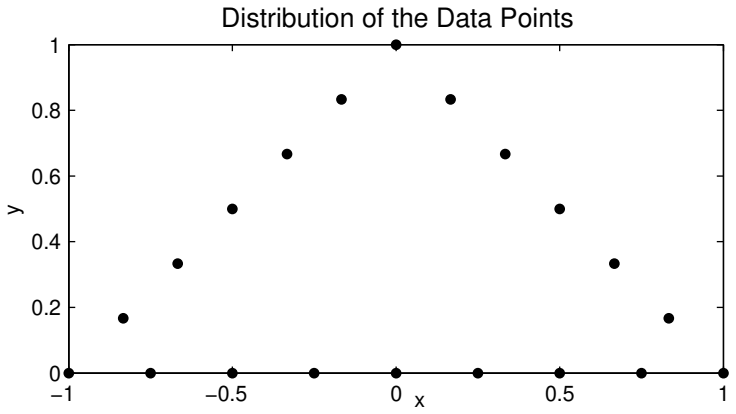


FIG. 5.1. *Distribution of the data points used for the example with computing expansion coefficients in Section 5.2.1 (The function value is one at the leftmost corner point, and zero at all the other points).*

Depending on what we are interested in, we can for example use the algorithm to generate

- The coefficients in (5.1) or in (5.2) (with an auxiliary rational function included, if desired).
- A display of the locations of the poles of $s(\underline{x}, \varepsilon)$ in the complex ε-plane. Since these originate from the A-matrix, they will be independent of $\underline{x}$ (they will depend only on $\phi(r)$ and on the data locations).

In the examples in Section 3, quite time consuming symbolic algebra (with *Mathematica*) was needed to obtain just the first one or two terms in (5.1) or (5.2) in cases of up to around $n = 5$ data points. In contrast, the numerical procedure gives, in seconds only, any number of expansion coefficients when n is up to around 100. Although this current size limit falls short of the sizes of some experimental data sets that one might want to use RBFs to analyze, the algorithm nevertheless vastly extends our ability to study phenomena related to $\varepsilon \to 0$ for RBF interpolants.

To illustrate this point, we use the Countor-Padé algorithm to compute the small ε expansion coefficients in (5.2) based on the 20 data points shown in Figure 5.1. As input data we use the cardinal function

$$f(x_j, y_j) = \begin{cases} 1 & \text{when } (x_j, y_j) = (-1, 0) \\ 0 & \text{otherwise} \end{cases} .$$

Since each expansion coefficient is a function of $\underline{x}$, we can display the coefficients over some domain in $\underline{x}$-space. Figure 5.2 displays—across the triangle—the first six expansion functions $s_k(\underline{x})$, $k = -4, -2, 0, 2, 4, 6$, computed in the case of IQ RBFs. It transpires that this triangular distribution of data points produces a fourth order pole in the interpolant at $\varepsilon = 0$. The data points allow for this possibility since they clearly fail the polynomial unisolvency condition stated in Theorem 4.1 (the interpolating polynomial would be undetermined with respect to any multiple of $x(1+x-y)(1-x-y)$). In each of the subplots of Figure 5.2, solid circles mark where the data points are located. We see that $s_0(\underline{x})$ exactly matches the input cardinal data (and provides a good approximation to the data) whereas all other expansion functions are zero at the data point locations. Note the small scale on vertical axis for the $s_{-4}(\underline{x})$ and $s_{-2}(\underline{x})$ functions. This is consistent with the fact that we only see divergence in the interpolant for very small values of ε. When evaluating the interpolant on the boundary of the triangle, the $s_{-4}(\underline{x})$ and $s_{-2}(\underline{x})$ functions are equal to zero. This is similar to the example

from Section 3.4 where 5 (or more) data points along a straight line cause divergence in the interpolant when evaluated off the line, but convergence when evaluated on the line. We also carried out the above experiment for the MQ and GA RBF. The interpolant based on MQ RBFs exhibits the same qualitative features as the one for the IQ case (for example, a fourth order pole at $\varepsilon = 0$). Like in all other cases that we have encountered, the interpolant based on GA RBFs has no pole at $\varepsilon = 0$; its expansion starts with the $s_0(\underline{x})$ term.

There are three options presently available for ensuring a convergent interpolant as $\varepsilon \to 0$:

(a) In place of the RBF interpolant $s(\underline{x}, \varepsilon)$ use $s_0(\underline{x})$ (also possibly incorporating the additional finite-ε corrections given by $s_2(\underline{x})$, $s_4(\underline{x})$, etc.).

(b) Avoid all data point distributions which are not consistent with the polynomial unisolvency requirements.

(c) Use GA RBFs.

The first option is presently only available by means of the Countor-Padé algorithm, and testing data sets for dangerous point distributions does not appear to be practical. Assuming our Conjecture 3.3 is correct, the last option may be the most convenient one.

5.2. Pole locations in the complex ε-plane. Our discussion thus far has been on the occurrence (and non-occurrence) of a pole in $s(\underline{x}, \varepsilon)$ at $\varepsilon = 0$. We now focus on poles that arise in the complex ε-plane for different types of point distributions, and show how this lends additional insight to the $\varepsilon = 0$ pole phenomenon.

5.2.1. Points approach cases where polynomial unisolvency fails. As we have demonstrated, a pole at $\varepsilon = 0$ is directly related to failure of the unisolvency condition. Below we revisit a couple of the examples from Section 3 to see how the poles in the complex ε-plane behave as the points approach a non-unisolvent set.

Three points approach being collinear. Let the data points be at locations $\{(0, y_0), (\frac{1}{2}, 0), (1, 0)\}$ with cardinal data $\{1, 0, 0\}$. With MQ, the exact result $\varepsilon = \pm\frac{2\,i\,A}{a\;b\;c}$ noted above gives

$$\varepsilon_{1,2} = \pm\ \frac{2\ i\ y_0}{\sqrt{(1+y_0^2)(1+4y_0^2)}}\ . \tag{5.3}$$

For small values of ε, we can alternatively expand the determinants in (2.3) to get

$$s(x, y, \varepsilon) = \frac{\frac{1}{4} y_0 y\ \varepsilon^4 + O(\varepsilon^6)}{\frac{1}{4} y_0^2\ \varepsilon^4 + \frac{1}{16}(1 - 2y_0^2)\ \varepsilon^6 + O(\varepsilon^8)} = \frac{y}{y_0} + O(\varepsilon^2). \tag{5.4}$$

When also y_0 is small, the first two terms in the denominator gives the approximate pole locations through

$$\frac{1}{4} y_0^2 + \frac{1}{16}\varepsilon^2 \approx 0 \quad \Rightarrow \quad \varepsilon_{1,2} \approx \pm\ 2 i y_0\ , \tag{5.5}$$

an excellent approximation to (5.3). Figure 5.3 (a) compares the approximation (5.5) to the exact (5.3) in the case of $y_0 = 0.01$. There is no visible discrepancy.

When the points approach being collinear (i.e. $y_0 \to 0$), the poles approach each other at the origin. When the points have become collinear, the ε^4-terms vanish from both the numerator and denominator in (5.4), and one can show that no poles remain.

Five points approach being collinear. Still considering MQ, we let now the point locations be $\{(0, y_0), (\frac{1}{4}, 0), (\frac{1}{2}, 0), (\frac{3}{4}, 0), (1, 0)\}$ with cardinal data $\{1, 0, 0, 0, 0\}$. For ε small, series expansion of the determinants gives

$$s(x, y, \varepsilon) = \frac{\frac{9\ y_0\ y}{1048576}\ \varepsilon^{14} + O(\varepsilon^{16})}{\frac{9\ y_0^2}{1048576}\ \varepsilon^{14} + O(\varepsilon^{16})} = \frac{y}{y_0} + O(\varepsilon^2).$$

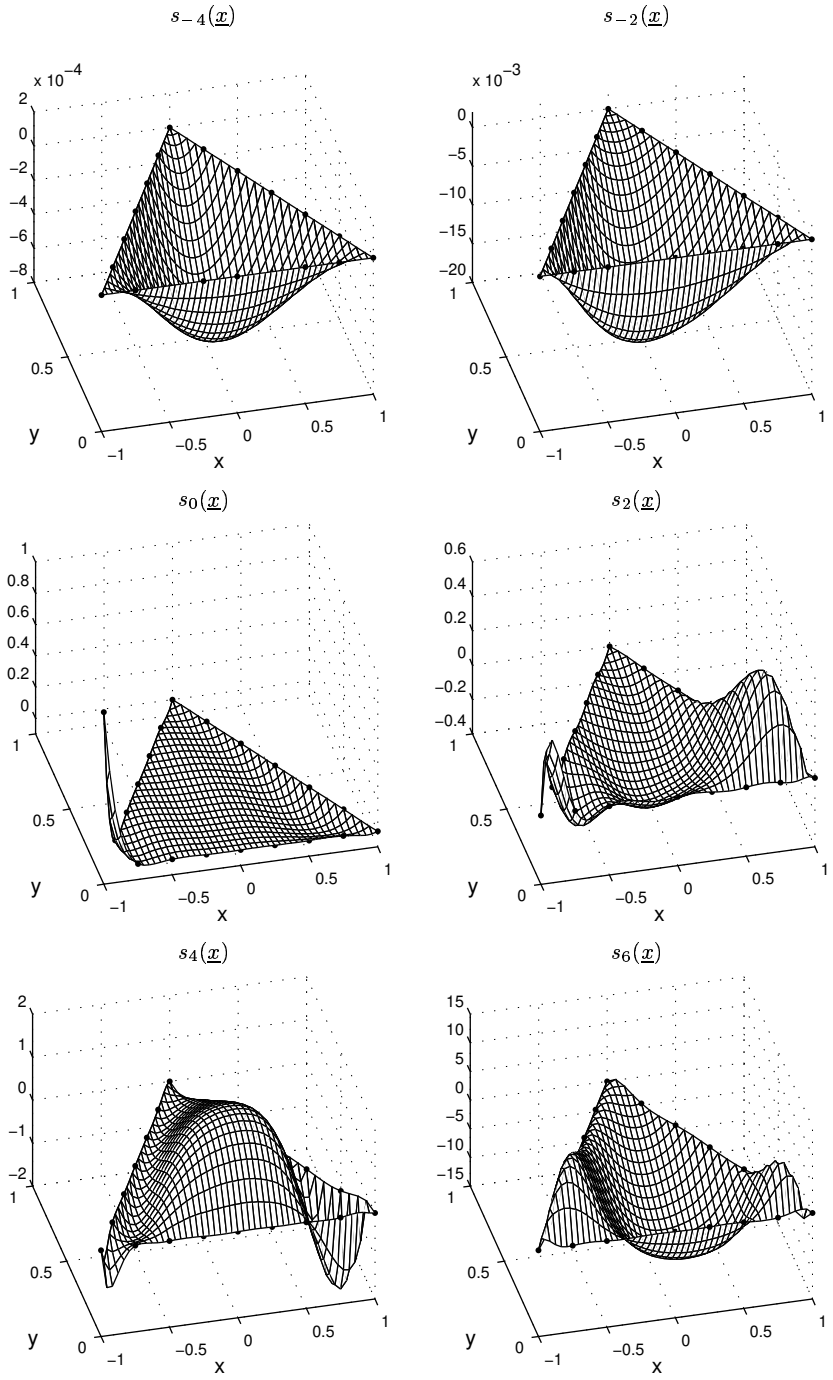


FIG. 5.2. *The first 6 terms from the expansion coefficients (5.2) of $s(\underline{x}, \varepsilon)$ for the example in Section 5.2.1. The solid circles mark the data points.*

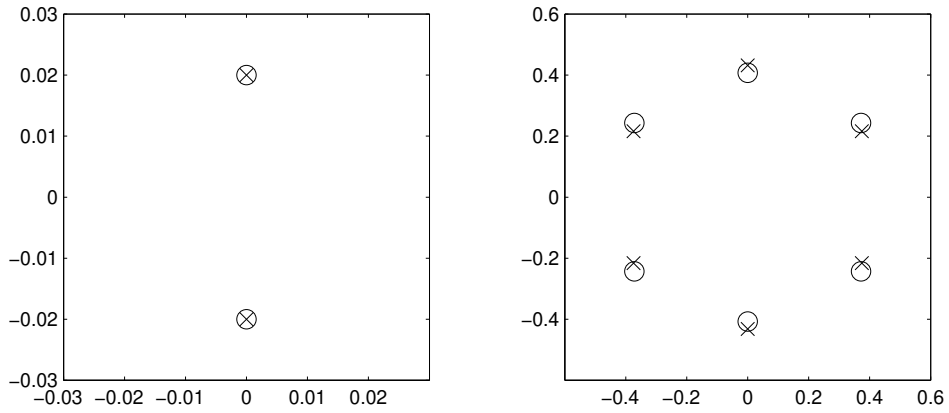


FIG. 5.3. *a and b. Exact and approximate locations of poles (circles and crosses, repsectively) when the data points approach being collinear: (a) three points, and (b) five points. The illustrations show the case of* $y_0 = 0.01$.

In more detail, the denominator is

$$\begin{aligned}\det_{\text{den}} = \frac{1}{4294967296}[36864y_0^2\varepsilon^{14} \quad - \quad 2304(15+4y_0^2)y_0^2\varepsilon^{16} + \\ + 288(85+50y_0^2+32y_0^4)y_0^2\varepsilon^{18} + \\ + 9(63-1460y_0^2-1528y_0^4-2080y_0^6-1024y_0^8)\,\varepsilon^{20}] + O(\varepsilon^{22})\end{aligned} \tag{5.6}$$

When ε is small, the terms for ε^{16} and ε^{18} will be small compared to the one for ε^{14}. The term for ε^{20} can be larger again (since it is lacking the factor y_0^2). The subsequent terms will again be decreasing. The determinant will therefore be zero also in the vicinity of the $\varepsilon-$values for which the terms with ε^{14} and ε^{20} add up to zero. This gives approximately

$$36864y_0^2 + 9\cdot 63\varepsilon^6 \approx 0 \quad \Rightarrow \quad \varepsilon_k \approx \frac{4\,y_0^{1/3}}{\sqrt[6]{63}}\, e^{\frac{\pi\, i\,(2k-1)}{6}}\,, \quad k = 1,2,\ldots,6. \tag{5.7}$$

Part b of Figure 5.3 compares, again in the case of $y_0 = 0.01$, the numerically determined pole locations for $s(x,y,\varepsilon)$ against the values given by the approximation (5.7). The agreement is again good. It is clear from (5.7) that all the six poles move in to the origin as $y_0 \to 0$. When the data points have become collinear, we see from (5.6) that the ε^{14}, ε^{16} and ε^{18} terms vanish, leaving the denominator expansion to start with ε^{20}. In the numerator, it transpires that only the ε^{14} and ε^{16} terms vanish, leaving us with

$$s(x,y,\varepsilon) = \frac{\frac{27\,y^2}{134217728}\varepsilon^{18} + O(\varepsilon^{20})}{\frac{567}{4294967296}\varepsilon^{20} + O(\varepsilon^{22})} = \frac{32\,y^2}{21}\,\frac{1}{\varepsilon^2} + O(1),$$

i.e. divergence as $\varepsilon \to 0$ entirely in accordance with (3.3).

5.2.2. Points are located so that polynomial unisolvency fails. We consider again the case with points on a parabola that was studied analytically in Section 3.6, but focus here on the poles that arise in the complex ε-plane. It is too algebraically complex to give the pole locations in closed-form, so we determine them numerically using the Contour-Padé algorithm.

The n data points are located at $x_k = \frac{k-1}{n-1}$, $y_k = x_k^2$, $k = 1,2,\ldots,n$. As soon as $n > 3$ the unisolvency condition fails. It was shown that for $n \geq 8$ there are cases where the interpolant

	$\lvert\varepsilon\rvert \leq 0.3$			$\varepsilon = 0$		
n	MQ	IQ	GA	MQ	IQ	GA
4	0	0	0	0	0	0
5	2	2	2	0	0	0
6	4	4	0	0	0	0
7	4	8	2	0	0	0
8	6	10	4	2	2	0
9	10	10	2	2	2	0
10	14	14	4	2	2	0
11	16	20	4	2	2	0
12	20	24	4	4	4	0

TABLE 5.2
The number of poles within the circle $|\varepsilon| = 0.3$ *and at* $\varepsilon = 0$ *for different RBFs*

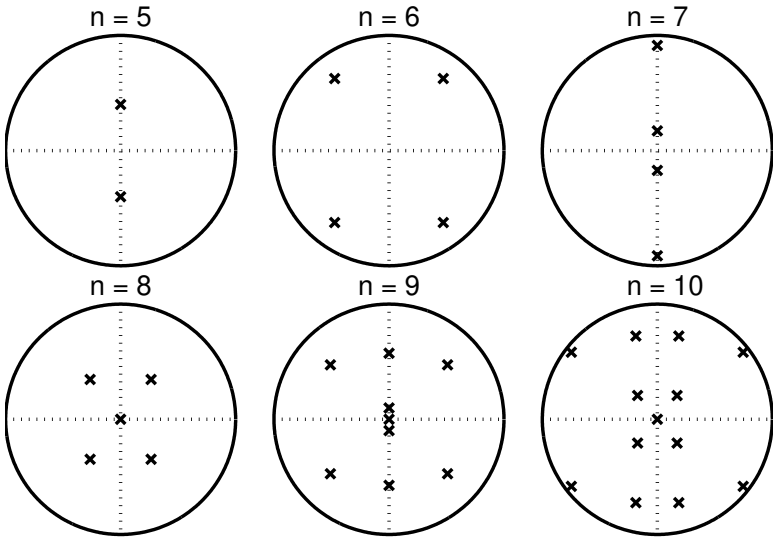


FIG. 5.4. *Pole locations inside the circle* $|\varepsilon| = 0.3$ *for points on a parabola using MQ*

diverges when using IQ or MQ. No divergent cases were found for GA (in accordance with our conjecture that GA interpolants never diverge as $\varepsilon \to 0$). Table 5.2 shows the number of poles inside the circle of radius 0.3 in the ε-plane and how many of those that are located at the origin (which leads to divergence). It is clear from the table that the GA RBF does behave differently from the other types of RBFs. We have not in any case observed poles at the origin for GA interpolants, and also the number of poles inside the fixed circle does not grow as fast. When n increases, it becomes more difficult to tell the exact number of poles, especially right at the origin. We have performed experiments for larger n that are not included here due to some numerical uncertainty in the precise numbers, but they seem to follow the same trends. The layout of the poles in the complex ε-plane for the different RBFs is shown in Figures 5.4–5.6.

5.2.3. Scattered points. It is only rarely possible to give all pole locations for $s(\underline{x}, \varepsilon)$ in closed-form. In the case of MQ with two arbitrarily located data points, the only poles appear at $\varepsilon = \pm\frac{\sqrt{2}\,i}{a}$ where a is the distance between the points. For three scattered points,

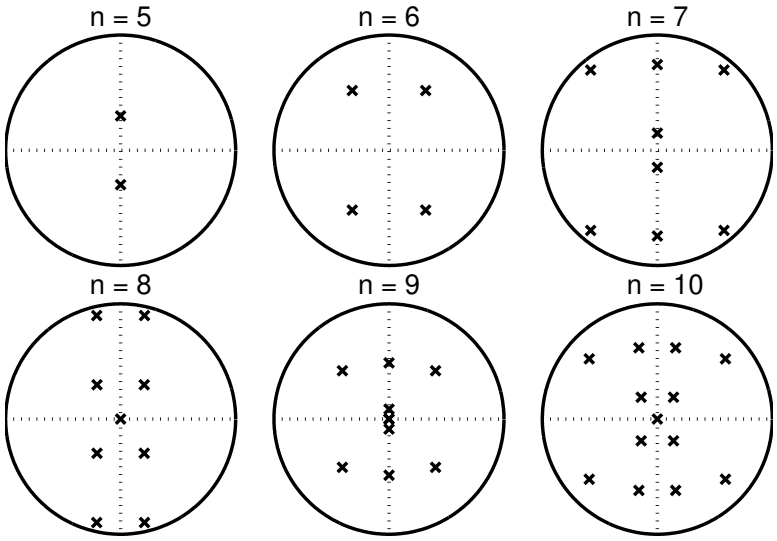


FIG. 5.5. *Pole locations inside the circle* $|\varepsilon| = 0.3$ *for points on a parabola using IQ*

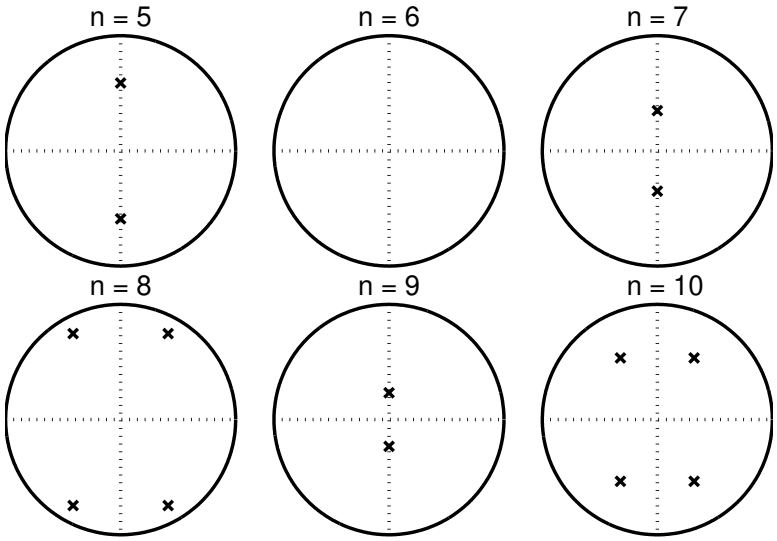
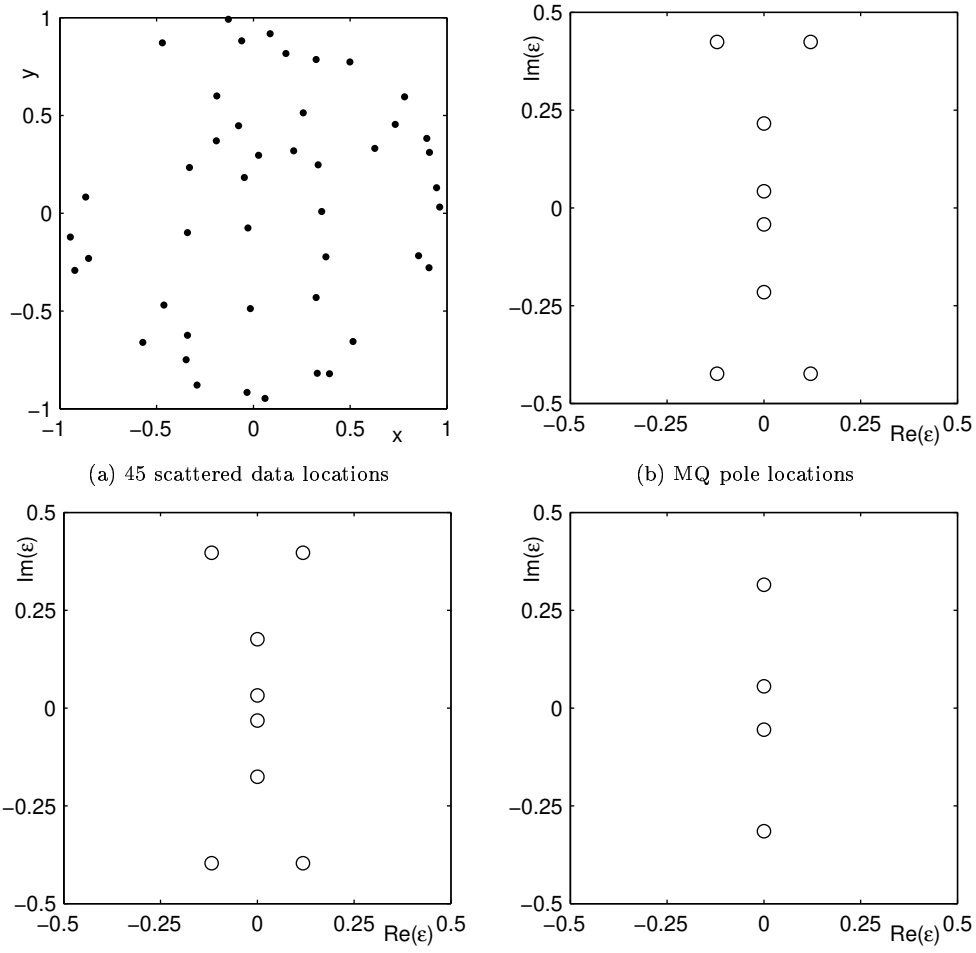


FIG. 5.6. *Pole locations inside the circle* $|\varepsilon| = 0.3$ *for points on a parabola using GA*

we find similarly $\varepsilon = \pm\frac{2\,i\,A}{a\,b\,c}$ where A is the area of the triangle formed by the points, and a, b, c are the lengths of its sides.

For several scattered data points, we can again use the Contour-Padé algorithm to determine the location of the poles of the RBF interpolant. As an example, we consider the 45 randomly scattered data points shown in Figure 5.7 (a). Using the Contour-Padé algorithm, we find that this distribution of data points produces the poles for the MQ, IQ, and GA interpolants as shown in Figure 5.7 (b–d). The figure shows several properties that we have observed also for other scattered data point distributions:

- The poles for the IQ and MQ interpolants are usually close together, whereas, the



(a) 45 scattered data locations

(b) MQ pole locations

(c) IQ pole locations

(d) GA pole locations

FIG. 5.7. *a to d. (a) Location of the 45 randomly scattered data points across the unit circle. (b–d) Location of the poles of the RBF interpolant in cases of MQ, IQ, and GA respectively.*

location of the poles for the GA interpolant is usually quite different. For example, the GA interpolant does not in this case have any poles off the imaginary axis while both the IQ and MQ interpolants do.

- There tends to be only a few poles near the origin even for larger numbers of scattered data points. The figure shows that each of the interpolants only has two poles that are near $\varepsilon = 0$.
- We have never observed any poles at the origin, i.e. divergence in the interpolants, for any of the main types of smooth RBFs in the case of scattered data points.

6. RBF approximation with a constant term included. In place of (1.1), one can for example consider interpolants of the form

$$s(\underline{x}) = \alpha + \sum_{k=1}^{n} \lambda_k \phi(\|\underline{x} - \underline{x}_k\|) \tag{6.1}$$

together with the constraint $\sum_{k=1}^{N} \lambda_k = 0$. Reasons for considering this, and also more general extensions of this kind, can include

- Positive definiteness of the linear system to solve for the coefficients λ_k in (1.1) (for distinct points and non-zero ε) is guaranteed for GA and IQ, but only for MQ if we include a constant term together with the constraint above.
- On infinite Cartesian grids, MQ and IQ can (for non-zero ε) *exactly* reproduce polynomials up to degrees d and $d-3$ (if $d \geq 3$) respectively, but GA cannot even reproduce a constant. Adding an explicit polynomial can provide such a feature, in case that is desired.
- Adding polynomial terms can in some cases improve the accuracy of interpolants, especially near boundaries. This and other more effective boundary improving methods are discussed in [20].

In the limit of $\varepsilon \to 0$, polynomials (of degrees increasing with n) will be reproduced automatically, even in the case of scattered finite point sets, so the last two of the observations above may then be of less significance. Although more general extensions than (6.1) may be of interest, we here limit our study to this form. It transpires that most of the results in the previous sections carry over with little difference. For example, Theorem 2.1 now becomes:

THEOREM 6.1. *For cardinal data* $f_k = \begin{cases} 1 & \text{if } k = 1 \\ 0 & \text{otherwise} \end{cases}$, *the RBF interpolant of the form (6.1) becomes*

$$s(\underline{x}) = \frac{\det \begin{bmatrix} \phi(\|\underline{x}-\underline{x}_1\|) & \phi(\|\underline{x}-\underline{x}_2\|) & \cdots & \phi(\|\underline{x}-\underline{x}_n\|) & 1 \\ \phi(\|\underline{x}_2-\underline{x}_1\|) & \phi(\|\underline{x}_2-\underline{x}_2\|) & \cdots & \phi(\|\underline{x}_2-\underline{x}_n\|) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi(\|\underline{x}_n-\underline{x}_1\|) & \phi(\|\underline{x}_n-\underline{x}_2\|) & \cdots & \phi(\|\underline{x}_n-\underline{x}_n\|) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}}{\det \begin{bmatrix} \phi(\|\underline{x}_1-\underline{x}_1\|) & \phi(\|\underline{x}_1-\underline{x}_2\|) & \cdots & \phi(\|\underline{x}_1-\underline{x}_n\|) & 1 \\ \phi(\|\underline{x}_2-\underline{x}_1\|) & \phi(\|\underline{x}_2-\underline{x}_2\|) & \cdots & \phi(\|\underline{x}_2-\underline{x}_n\|) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi(\|\underline{x}_n-\underline{x}_1\|) & \phi(\|\underline{x}_n-\underline{x}_2\|) & \cdots & \phi(\|\underline{x}_n-\underline{x}_n\|) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}} \tag{6.2}$$

Both versions of the proof for Theorem 2.1 carry over, so we omit the details here.

Regarding the case with three data points along a straight line, equation (3.1) needs to be replaced with

$$s(x,y) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} + \frac{1}{3} \cdot \frac{y^2}{(x_1-x_2)(x_1-x_3)} + O(\varepsilon^2)$$

on the assumptions that $a_1 \neq 0$ and $a_2 \neq 0$. Similarly, for five points along a line, (3.3) needs to be replaced by

$$s(x,y) = \frac{4\,y^2}{(x_1-x_2)(x_1-x_3)(x_1-x_4)(x_1-x_5)} \, \frac{a_2\,a_3}{(75\,a_3^2 - 140\,a_2 a_4)} \, \frac{1}{\varepsilon^2} \; + \; O(1),$$

this time on the same assumption as for its earlier counterpart, viz. $2a_2^2 - 5a_1a_3 \neq 0$. In this case, all the standard smooth RBF types (e.g. MQ, IQ, GA) lead to divergence. In the limit of $\varepsilon \to 0$, not only do the coefficients λ_k diverge to infinity, but so does the constant α in (6.1). Although the limiting interpolant along a line of data points will again become the Lagrange interpolation polynomial, there is no immediate counterpart to Theorem 3.1.

7. Conclusions. We have in this study been considering interpolants based on smooth RBFs featuring a shape parameter ε. It has been known for a long time that $\varepsilon > 0$ leads to a well-defined interpolant in the case of Gaussian RBFs and, thanks to more recent work of Micchelli [2] and others, equivalent results are known for many other cases. In this study, we are making a number of observations regarding the limit when the basis functions become increasingly flat ($\varepsilon \to 0$). We first note

- When the limit exists, it takes the form of a multivariate polynomial,
- The Contour-Padé algorithm permits numerical computations—without any deterioration of the conditioning—all the way down to the $\varepsilon \to 0$ limit.

We then make a number of observations that shed new light on this limit. In particular, we note that

- The existence of the limit, for most RBFs, depends very critically on the data point distributions. This is connected to the issue of 'polynomial unisolvency'.
- Gaussian RBFs appear to have the remarkable property of never leading to a divergent interpolant as $\varepsilon \to 0$. At least among other standard types of smooth RBFs, Gaussians are unique in this respect.

REFERENCES

[1] R. J. Y. McLeod and M. L. Baart. *Geometry and Interpolation of Curves and Surfaces.* Cambridge University Press, Cambridge, 1998.

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

[3] E. W. Cheney and W. A. Light. *A Course in Approximation Theory.* Brooks/Cole, New York, 2000.

[4] M. J. D. Powell. The theory of radial basis function approximation in 1990. In W. Light, editor, *Advances in Numerical Analysis, Vol. II: Wavelets, Subdivision Algorithms and Radial Functions*, pages 105–210. Oxford University Press, Oxford, UK, 1992.

[5] Z. Wu and R. Schaback. Local error estimates for radial basis function interpolation of scattered data. *I.M.A. J. Numer. Anal.*, 13:13–27, 1993.

[6] W. R. Madych and S. A. Nelson. Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation. *J. Approx. Theory*, 70:94–114, 1992.

[7] R. Schaback. Error estimates and condition numbers for radial basis function interpolants. *Adv. Comput. Math.*, 3:251–264, 1995.

[8] J. Yoon. Spectral approximation orders of radial basis function interpolation on the Sobolev space. *SIAM J. Math. Anal.*, 23(4):946–958, 2001.

[9] R. E. Carlson and T. A. Foley. The parameter R^2 in multiquadric interpolation. *Comput. Math. Appl.*, 21:29–42, 1991.

[10] T. A. Foley. Near optimal parameter selection for multiquadric interpolation. *J. Appl. Sci. Comput.*, 1:54–69, 1994.

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

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

[13] B. Fornberg. *A Practical Guide to Pseudospectral Methods.* Cambridge University Press, Cambridge, 1996.

[14] B. Fornberg and G. Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. *Comput. Math. Appl.*, 2003. Submitted.

[15] E. Larsson and B. Fornberg. A numerical study of some radial basis function based solution methods for elliptic PDEs. *Comput. Math. Appl.*, 2001. To appear.

[16] M. J. D. Powell. Radial basis function methods for interpolation to functions of many variables. DAMTP Report NA11, University of Cambridge, 2001.

[17] E. Larsson and B. Fornberg. Theoretical aspects of multivariate interpolation with increasingly flat radial basis functions. To be submitted.

[18] B. Fornberg and N. Flyer. Accuracy of radial basis function derivative approximations in 1-d. *Adv. Comput. Math.*, 2003. Submitted.

[19] R. Schaback. Multivariate interpolation by polynomials and radial basis functions. Submitted, 2002.

[20] B. Fornberg, T. A. Driscoll, G. Wright, and R. Charles. Observations on the behavior of radial basis functions near boundaries. *Comput. Math. Appl.*, 43:473–490, 2002.

Appendix D:
The Contour-Padé Toolbox for MATLAB

Grady Wright

1 Introduction

The Radial Basis Function (RBF) method is a primary tool for interpolating multidimensional data. RBF interpolants based on infinitely smooth basis functions, such as multiquadrics (MQs $\phi(r) = \sqrt{1+(\varepsilon r)^2}$) have proven very successful. However, three main issues have hindered their use: severe ill-conditioning for a fixed number of data points N and small values of ε, similar ill-conditioning problems for a fixed ε and large values of N, and high computational cost. The Contour-Padé algorithm of Fornberg and Wright [1] resolves the first of these problems.

Small values of ε are of interest because they sometimes greatly increase the accuracy of the RBF interpolants [2]. In addition, it was shown by Driscoll and Fornberg [3] that for finite 1-D data the RBF interpolant in the limit of $\varepsilon \to 0$ converges to the Lagrange interpolating polynomial. This result has been extended to higher dimensions independently by Schaback [4] and Larsson and Fornberg [5]. Some issues related to the behavior of the limiting RBF interpolants in higher dimensions are discussed in [6].

The Contour-Padé algorithm allows for the stable computation of RBF interpolants (based on infinitely smooth basis functions) for all values of ε, including $\varepsilon = 0$. In addition, there is no increase in computational cost as ε decreases. The basic idea of the algorithm comes from the fact that the RBF interpolant is a meromorphic function of ε in the vicinity of the origin of the complex ε-plane. This allows us to compute the RBF interpolant as the sum of a rational function in ε and a power series in ε. These functions are determined by evaluating the RBF interpolant around a contour (a circle) in the complex ε plane and by the use of Padé approximants (see [1] for more details). Although the current version of this algorithm works well only for up to around 100 2-D data points (with more in higher dimensions), it nevertheless forms a powerful (indeed, the *only*) numerical tool for studying the limit of the RBF interpolants as $\varepsilon \to 0$.

The Contour-Padé Toolbox for MATLAB contains an implementation of the Contour-Padé algorithm and provides users with a set of functions for computing RBF interpolants using the algorithm. It should be noted that the Contour-Padé algorithm is in no way specific to computing RBF interpolants. The algorithm can be used for computing values of any meromorphic function $C(\varepsilon)$ around $\varepsilon = 0$. Therefore, the toolbox contains a generic function for the algorithm that allows users to apply it for different problems. For example, one can use the generic function for computing RBF based solutions to Poisson's equation for all values of ε, see [7] for more information.

A description of the main functions in the Contour-Padé toolbox is presented in the next section.

Type of basis function	$\phi(r)$
Piecewise smooth RBFs	
Piecewise polynomial (R_n)	$\lvert r\rvert^n$, n odd
Thin Plate Spline (TPS_n)	$\lvert r\rvert^n \ln r$, n even
Infinitely smooth RBFs	
Multiquadric (MQ)	$\sqrt{1+(\varepsilon r)^2}$
Inverse quadratic (IQ)	$\frac{1}{1+(\varepsilon r)^2}$
Gaussian (GA)	$e^{-(\varepsilon r)^2}$

Table 1: Some commonly used radial basis functions

2 Description of the functions

There are two main functions in the toolbox: `rbfApprox` and `contourPade`. A description of both of these follows.

2.1 rbfApprox

The radial basis function (RBF) method involves creating an approximating function to a set of multidimensional scattered data points by taking linear combinations of a single basis function that only depends on the *radial* distance from its center. The basic RBF approximation to a set of n distinct data points $\{\underline{x}_k\}_{k=1}^n$ and corresponding data values $\{f_k\}_{k=1}^n$ is given as follows:

$$s(\underline{x}) = \sum_{k=1}^{n} \lambda_k \phi(\|\underline{x} - \underline{x}_k\|_2) \ , \tag{1}$$

where $\underline{x}, \underline{x}_k \in R^d$ and $\phi(r)$ is some scalar *radial* function. Requiring that $s(\underline{x}_j) = f_j$, $j = 1, \ldots, n$ leads to the following symmetric linear system for the expansion coefficients λ_j:

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} \\ \lambda \\ \\ \end{bmatrix} = \begin{bmatrix} \\ f \\ \\ \end{bmatrix} \ , \tag{2}$$

where the entries of A are given by $a_{j,k} = \phi(\|\underline{x}_j - \underline{x}_k\|_2)$. This linear system can be shown to be unconditionally nonsingular for a large class of basis functions $\phi(r)$ [8] (and for an even larger class of functions if (1) is modified to include some low order polynomial terms; see [8] for more details).

The types of basis functions available for use in the RBF method can be split into two main categories: piecewise smooth and infinitely smooth. Examples from each of these categories are given in Table 1. The accuracy of RBF approximations based on the infinitely smooth basis functions is far superior to those based on the piecewise smooth functions (spectral accuracy for the former vs. algebraic accuracy for the latter [9], [10], [11]). However, the conditioning of the linear system (2) is far superior for the piecewise smooth type than the infinitely smooth type. For the former, the condition number of the A matrix in (2) grows algebraically as the number of data points increases, while the latter leads to exponential growth.

Another property that differs between the two types of basis functions is that the infinitely smooth variety feature a free parameter ε for controlling their shape. By convention, decreasing values of ε correspond to the function becoming increasingly flat. As mentioned in the

introduction, the value of ε significantly affects the accuracy of the RBF approximations, with small values often leading to the most accurate approximation. The value of ε also affects the stability of the linear system (2). For a fixed number of data points, the condition number of the A matrix grows without bound as $\varepsilon \to 0$. This last attribute has made it impossible to explore RBF approximations for small values of ε including $\varepsilon = 0$. Until now.

By using the Contour-Padé algorithm, the `rbfApprox` function allows one to compute the RBF approximation to a given set of data for the full range of ε. As mentioned in the introduction, this function works well only for up to around 100 2-D data points (with more in higher dimensions). A description of how to use this function follows.

Here is how the function should be called:

```
[s, poles, rad] = rbfApprox(xd,xc,fd,xe,rbfFunc,snglrty,epsilon,rad)
```

A description of the inputs to and outputs from the function is given below.

Required Inputs

`xd:` The points at which the data values `fd` are given. `xd` must be a 2-D array and arranged so that the columns represent each dimension of the points. For example, for 25 points in 3-D, xd should be a 25x3 array. Note that `xd` must contain distinct points for an approximation to be computed.

`xc:` The points at which to center the basis functions. `xc` must be a 2-D array and arranged just like `xd`. The center points are most commonly chosen to be equal to the data points, i.e. `xc=xd`. For more information on the effects of choosing center points different from data points see, for example [12]. Note that `xc` must contain distinct points for an approximation to be computed. Supplying more centers than data points creates an underdetermined system and supplying more data points than centers creates an overdetermined system. In both cases, the system will be solved in the least squares sense. **NOTE:** The least squares solution is not implemented yet, so `xd` and `xc` must contain the same number of points.

`fd:` The data values corresponding to the points `xd`. To compute RBF approximations for different sets of data values corresponding to the same set of points `xd`, make `fd` a 2-D array with the columns containing the different sets of data values. For example, to compute two RBF approximations with one based on samplings of the function $f(x)$ at `xd` and the other on samplings of the function $g(x)$ at `xd`, `fd` would need to be arranged as follows:

```
fd = [f(xd) g(xd)];
```

`xe:` The points at which to evaluate the RBF approximation. `xe` must be a 2-D array and arranged just like `xd` and can contain an arbitrary number of points.

`rbfFunc:` Matlab m-file or inline expression representing the basis function to be used for creating the RBF approximation (see Table 1 for some examples of these basis functions). This function should be a matrix valued function. For example

```
mqrbf = inline('sqrt(1+(epsilon.*r).^2)');
```

creates the MQ RBF. This function will be called as follows so that it is evaluated with the value contained in `ep` and the distance matrix `rd`:

```
feval(rbfFunc,ep,rd);
```

`snglrty`: The ε-singularity of the basis function with $r = 1$ that is closest to the origin. For example, the MQ RBF with $r = 1$ has a branch point singularity at $\varepsilon = \pm i$. These are the closest (and only) singularities to the origin, therefore `snglrty` should be set to `i` or `-i`. The IQ RBF with r=1 has a pole singularity at $\varepsilon = \pm i$, so again, `snglrty` should be set to `i` or `-i`. For RBFs with no singularities in the finite-complex plan, e.g. the GA RBF, `snglrty` should be set to `nan`.

`epsilon`: The value(s) of the shape parameter ε at which to compute the RBF approximation; this can include $\varepsilon = 0$. The values should be arranged in a 1-D column vector.

Optional Inputs

`rad`: The radius to use for the circle in the Contour-Padé algorithm. Note that `rad` should be chosen so that it is less than (in magnitude) any of the ε-singularities present in the basis functions. For example, the MQ RBF has a branch point singularity at $\varepsilon = i/r$. Thus, `rad` should be chosen so that it is less than $i/\max(r)$, where $\max(r)$ is the maximum distance between any of the data points (`xd`) and center points (`xc`) and the evaluation points (`xe`) and the center points. In addition, `rad` should be chosen so that it avoids the computationally ill-conditioned region of computing the RBF approximation. If `rad` is not supplied then the function will attempt to find an appropriate one. Supplying this input will speed up the function.

Outputs

`s`: The RBF approximation(s) evaluated at the points `xe`. Each column of `s` contains the RBF approximation based on the corresponding column of data in `fd`. Each row of `s` contains the value of the RBF approximations at the point in the corresponding row of `xe`. If `epsilon` is a vector then `s` will be a 3-D array, with the third dimension containing the RBF approximations based on the different values of `epsilon`. For example, at the point given by the `j`th value in `xe`, the RBF approximation to the data in the `k`th column of `fd` and the `l`th value of ε in `epsilon` is given by `s(j,k,l)`.

`poles`: Column vector containing the poles of the RBF approximation that are inside the circle of radius `rad` in the complex ε-plane.

`rad`: The radius of the circle used for computing the RBF approximation with the Contour-Padé algorithm. Note that for values of ε greater than this value, the standard method of computing the RBF approximations is used, i.e. solve for the expansion coefficients and evaluate the RBF approximation.

As an example of how to use this function, we consider the problem presented in Section 2 of [1], i.e. for $0 \leq \varepsilon \leq 1$, compute the MQ RBF interpolant to the function

$$f(\underline{x}) = f(x_1, x_2) = \frac{59}{67 + (x_1 + \frac{1}{7})^2 + (x_2 - \frac{1}{11})^2} \tag{3}$$

at $(x_1, x_2) = (0.3, -0.2)$ based on the 41 data points shown in Figure 1.

In order to use the `rbfFunc` to solve this problem, we first declare (3) as an *inline* function:

```
>>f = inline('59./(67+(x1+1/7).^2 + (x2-1/11).^2)');
```

We also declare the MQ RBF as an *inline* function:

```
>>rbf = inline('sqrt(1+(ep*rd).^2)');
```

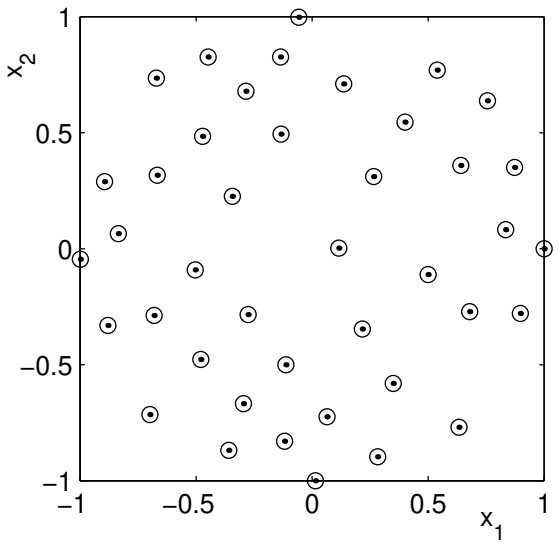


Figure 1: Locations of the 41 data points for sampling (3).

The data points shown in Figure 1 for sampling (3) have been saved in the file `N41DataPointsEx.mat` as variables `x1` and `x2` (both are column vectors). This file is included in the Contour-Padé toolbox. To load in the data points we use the following command:

```
>>load N41DataPointsEx;
```

Next, we declare variables to hold the data points, the evaluation point, and the values of ε:

```
>>xd = [x1 x2];     % Data points for sampling f.
>>xe = [0.3 -0.2]; % Point at which to evaluate the RBF interpolant.
>>ep = 0:0.01:1;   % Compute the interpolant for epsilon=0,0.01...,1.00
```

There are two more values we have to define before using the function, these are the `snglrty` and the `rad`. The MQ RBF has a branch point singularity at $\varepsilon = \pm i$ (when $r = 1$), so we let `snglrty=i`. Following what was done in [1], we assign 0.42 for the radius (we could also just let the function choose the value for the radius).

We are now ready to use the `rbfApprox` function. The following call computes the RBF approximation based on the values defined above:

```
>>[s, poles, rad] = rbfApprox(xd,xd,f(x1,x2),xe,rbf,i,ep,0.42);
```

Upon completion of the computations, `s` will be a 3-D array containing the MQ RBF approximation to (3) at all the values of ε contained in `ep`. The size of `s` will be $1 \times 1 \times 101$ since there was only one evaluation point (the number of entries in the first dimension), one set of data to compute the approximation two (the number of entries of the second dimension), and 101 values of ε to compute the approximation with (the number of entries in the third dimension). The magnitude of the error in the approximation at each value of ε can be computed as follows:

```
>>maxErr = squeeze(abs(s - f(xe(1,1),xe(1,2))));
```

These values can then be used to create a plot similar to that of Figure 5.1 in [1]. The following command will create the plot:

```
>>semilogy(ep',maxErr);
```

The variable `poles` contains the poles of the RBF approximation inside the circle of radius 0.42 in the complex ε-plane. We find that for the 41 data points used to compute the approximation there are 6 poles inside the circle of radius 0.42 (note that the location of the poles are independent of the evaluation point(s) `xe` and the function values `fd`, see [1] and [6]).

Since we specified a value of 0.42 for the radius of the circle, the value of `rad` will be 0.42. Had we let the `rbfApprox` function compute a value for the radius, `rad` would contain the value that was chosen. For a new set of data points, a recommended practice is to first let the `rbfApprox` function compute a value for the radius, then use this value on successive calls to the function with the same set of data points.

For more examples of how to use the `rbfApprox` function, please refer to the files

- `rbfApproxDemo1.m`,
- `rbfApproxDemo2.m`, and
- `rbfApproxDemo3.m`

which are included in the Contour-Padé toolbox. The first file contains the code for producing the example above, the second file shows how to use the `rbfApprox` function with multiple evaluation points, and the third file shows how to use the `rbfApprox` with multiple data sets.

2.2 contourPade

The Contour-Padé algorithm is a generic method for computing the values of any meromorphic function $C(\varepsilon)$ around $\varepsilon = 0$. The intended $C(\varepsilon)$ for the method are ones that cannot be directly evaluated in the vicinity of $\varepsilon = 0$ because they are numerically ill-conditioned there. However, they can be directly evaluated in a stable manner for larger values of ε. For example,

$$C(\varepsilon) = \frac{1 - \cos(\varepsilon)}{\varepsilon^2} + \frac{p}{\varepsilon^2 + q}\,, \tag{4}$$

where p and q are some real non-zero constants. This function is well-defined for all values of ε and can be directly evaluated in a stable manner for values of ε away from 0. Near $\varepsilon = 0$, however, the function is numerically ill-conditioned because of the numerical cancelation that occurs in the numerator of the first term and the subsequent magnification of this cancelation as a result of the division by a small (in magnitude) value of ε.

The RBF approximation (1) based on the infinitely smooth basis functions is another example of a function that is only numerically ill-conditioned near $\varepsilon = 0$, but that is (usually) well-defined there. To define the $C(\varepsilon)$ for this problem, we need to re-write (1). First, we denote the RBF approximation by $s(\underline{x}, \varepsilon)$ to explicitly indicate that it is a function of ε. Next, we combine (1) and (2) to arrive at the following form of the RBF approximation:

$$s(\underline{x}, \varepsilon) = \begin{bmatrix} B(\varepsilon) \end{bmatrix} \begin{bmatrix} A(\varepsilon) \end{bmatrix}^{-1} \begin{bmatrix} f \end{bmatrix}\,, \tag{5}$$

where the entries of $B(\varepsilon)$ are given by $b_j = \phi(\|\underline{x} - \underline{x}_j\|_2)$ for $j = 1, \ldots, n$ and $A(\varepsilon)$ is equivalent to the A in (2). Now, $C(\varepsilon)$ for the RBF approximation is given by the product $B(\varepsilon)A^{-1}(\varepsilon)$

(or $B(\varepsilon)A^{-1}(\varepsilon)f$ since f has no ε-dependence). While $A(\varepsilon)$ rapidly approaches singularity as $\varepsilon \to 0$, the product $B(\varepsilon)A^{-1}(\varepsilon)$ (usually) remains bounded.

The Contour-Padé algorithm can be used in both of the above examples to evaluate each $C(\varepsilon)$ for any ε near 0 in a stable manner. This is essentially accomplished by evaluating $C(\varepsilon)$ on a contour (a circle) enclosing the origin in the complex ε-plane where $C(\varepsilon)$ suffers from no ill-conditioning (see [1] for more details).

The `contourPade` function implements the Contour-Padé algorithm. It computes values of a user-defined $C(\varepsilon)$ for the complete range of ε. The function should be called as follows:

```
[C, poles] = contourPade(Cfunc,rad,epsilon,p1,p2,...)
```

Below is a description of each of the inputs. Note that the `rbfApprox` function uses this function to compute RBF approximations for the full range of ε.

Required Inputs

`Cfunc`: Matlab m-file or inline expression representing the analytic function $C(\varepsilon)$. This can be a matrix-valued function as is the case for the RBF problem (5). Here is an example of the way this function will be called so that $C(\varepsilon)$ is evaluated at $\varepsilon = 1$: `feval(Cfunc,1)`. **NOTE:** Currently, the `contourPade` function assumes `Cfunc` has a 4-fold symmetry in ε and if `Cfunc` produces a matrix then each entry in the matrix has the same pole locations in the complex ε-plane. These assumptions are satified by the RBF interpolation problem and allow the algorithm to run much faster. In later releases of this package, a version of the algorithm may be included with less restrictive assumptions.

`rad`: The radius of the countour (a circle) that `C` will be integrated around in the complex ε-plane. Note that `rad` should be chosen so that it avoids any values of ε where $C(\varepsilon)$ is ill-conditioned, e.g. poles and branch points.

`epsilon`: A scalar or vector containing the value(s) of ε that $C(\varepsilon)$ will be evaluated at using the Contour-Padé algorithm. Note that the value(s) should be less than or equal to `rad`.

Optional Inputs

`p1,p2,...`: Optional parameters `p1,p2,...` which, if present, get passed directly to `Cfunc` like `feval(Cfunc,epsilon,p1,p2,...)` or `Cfunc(epsilon,p1,p2,...)`.

Outputs

`C`: The value of $C(\varepsilon)$ at the supplied value(s) of ε. Since $C(\varepsilon)$ can be matrix-valued, `C` will be a 3-D array if `epsilon` is a vector. In this case, the third dimension represents the value of $C(\varepsilon)$ at the various values of ε. For example, the following code will print the value of $C(\varepsilon)$ at the `jth` ε-value (i.e. `epsilon(j)`): `C(:,:,j)`

`poles`: Column vector containing any poles of `C` that were detected by the Padé approximant inside the circle of radius `rad` in the complex ε-plane.

As an example of how to use this function, we consider computing the value of (4) for small values of ε including $\varepsilon = 0$.

To use the `contourPade` function to compute (4), we first declare (4) as an *inline* function in MATLAB:

```
>>Cfunc = inline('(1-cos(ep))./ep.^2 + p./(ep.^2+q)');
```

Next, we declare the values of ε we wish to compute the function at:

```
>>ep = 0:0.01:0.1;
```

Now, we are ready to use the `contourPade` function to compute the solution of (4) with $p = 1/8$ and $q = 1/4$. However, before we do this, we note that (4) does not suffer from any numerical ill-conditioning for any values of ε on the unit circle in the complex ε-plane. Thus, we let `rad=1`. The call to use the `contourPade` for this problem is as follows:

```
>>[C,poles] = contourPade(Cfunc,1,ep,1/8,1/4);
```

If we analytically compute the limit of (4) as $\varepsilon \to 0$ with these values of p and q, we find that $f(0) = 1$. This is the exact value computed by the `contourPade` function. To see this we can print out the corresponding $\varepsilon = 0$ value in `C` as follows:

```
>>C(1)
```

Note that since $C(\varepsilon)$ is a scalar-valued function this code works for printing out the value at $\varepsilon = 0$, however, the "correct" way to print this value out is `C(1,1,1)`. A comparision of the accuracy in the rest of the values of (4) computed by the `contourPade` function can be done by reformulating (4) in a more computationally stable form, i.e.

$$C(\varepsilon) = \frac{2\sin^2(\frac{\varepsilon}{2})}{\varepsilon^2} + \frac{p}{\varepsilon^2 + q}$$

When this is done, we find that the values of (4) as computed by the `contourPade` function have a maximum error of approximately 10^{-15}.

In addition to computing the values of (4), the `contourPade` function also numerically computed the location of the poles of (4) since they were located inside the circle of radius `rad=1` that we used. The location of the poles are $\pm i/2$ and by inspecting the value of `poles`, we find this is the exactly what was computed by the `contourPade` function.

For more examples of how to use the `contourPade` function, please refer to the files

- `contourPadeDemo1.m`, and
- `contourPadeDemo2.m`

which are included in Contour-Padé toolbox. The first file contains an example similar to the one above, and the second file contains an example with a matrix-valued $C(\varepsilon)$.

References

[1] B. Fornberg and G. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter*, submitted to Comput. Math. Appl. (2003).

[2] W. Madych, *Miscellaneous error bounds for multiquadric and related interpolants*, Comput. Math. Appl., 24 (1992), 121–138.

[3] T.A. Driscoll and B. Fornberg, *Interpolation in the limit of increasingly flat radial basis functions*, Comput. Math. Appl. 43 (2002), 413–422.

[4] R. Schaback, *Multivariate Interpolation by polynomials and radial basis functions*, submitted (2002).

[5] E. Larsson and B. Fornberg, *Theoretical aspects of multivariate interpolation with increasingly flat radial basis functions*, to be submitted (2003).

[6] B. Fornberg, G. Wright, and E. Larsson, *Some observations regarding interpolants in the limit of flat radial basis functions*, to appear in Comput. Math. Appl. (2003).

[7] E. Larsson and B. Fornberg, *A numerical study of some radial basis function based solution methods for elliptic PDEs*, to appear in Comput. Math. Appl. (2003).

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

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

[10] M.D. Buhmann and N. Dyn, *Spectral convergence of multiquadric interpolation*, Proceedings of the Edinburgh Mathematical Society, 36, Edinburgh (1993), 319–333.

[11] H. Wendland, *Gaussian Interpolation Revisited*, K. Kopotun, T. Lyche, and M. Neamtu (eds.), Trends in Approximation Theory, Vanderbilt University Press, Nashville, 2001, 427–436.

[12] B. Fornberg, T.A. Driscoll, G. Wright, and R. Charles, *Observations on the behavior of radial basis functions near boundaries*, Comput. Math. Appl. 43 (2002), 473–490.