Barycentric Rational Interpolation and Spectral Methods

by

Nathan Sharp

B.Sc., Principia College, 2010

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science

in the
Department of Mathematics
Faculty of Science

SIMON FRASER UNIVERSITY
Fall 2014

APPROVAL

Name: Nathan Sharp

Degree: Master of Science

Title of Thesis: Barycentric Rational Interpolation and Spectral Methods

Examining Committee:

Dr. Ralf Wittenberg
Associate Professor
Chair

Dr. Manfred Trummer
Senior Supervisor
Professor

Dr. Robert Russell
Supervisor
Emeritus Professor

Dr. Ben Adcock
Internal / External Examiner
Assistant Professor

Date Approved: November 24th, 2014

Partial Copyright Licence

Abstract

For the numerical solution of differential equations spectral methods typically give excellent accuracy with relatively few points (small N), but certain numerical issues arise with larger N. This thesis focuses on spectral collocation methods, also known as pseudo-spectral methods, that rely on interpolation at collocation points. A relatively new class of interpolants will be considered, namely the Floater-Hormann family of rational interpolants. These interpolants and their properties will be studied, including their use in differentiation by means of differentiation matrices based on rational interpolants in the barycentric form. Then, consideration will be given to the solution of singularly perturbed boundary value problems through the use of boundary layer resolving coordinate transformations. Finally, coupled systems of singularly perturbed boundary value problems will be investigated, though only with the standard Chebyshev collocation method.

To my mum and dad

"...there is nothing either good or bad, but thinking makes it so."

—Shakespeare, Hamlet

Acknowledgments

Firstly, I would like to thank my supervisors, especially Dr. Manfred Trummer, without whom this undertaking would not have been possible. His support throughout the entire process has been tremendous.

I would also like to thank my fellow graduate students, who not only provided academic support and camaraderie, but with some of whom I also became good friends. I would like to give special thanks to Xin, who answered many questions in almost all of my classes.

Finally, I would like to thank my family, particularly my mum and dad, who provided unconditional support throughout these past two years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In scientific computing, spectral methods are one of many methods used when solving differential equations numerically. They are methods that typically give excellent error for relatively small values of N ("points"), but also usually have numerical issues associated with their differentiation matrices. Spectral methods are usually thought of as numerical methods which have basis functions with support over the entire computational domain (or global basis functions). One solves the differential equations by choosing the coefficients of the basis functions. For example, with the equation $Lu(x) = f(x)$, where L is a differential operator, we would assume a solution $u_N(x) = \sum_0^N a_k \phi_k(x)$. Plugging this form of the solution into the differential equation, we can look a the residual, $R = f - Lu$. One can then minimize the residual in different ways, such as setting the error to zero at certain collocation points equal in number to the undetermined coefficients (pseudospectral or collocation method) or choosing basis functions orthogonal to the residual (Galerkin). There is much more to be said about these and other spectral methods; see [14] for more.

In this thesis, we only consider spectral collocation methods. First, we look at a relatively new class of interpolating functions, the Floater-Hormann family of rational interpolants. We will investigate their properties and their use in differentiation in conjunction with the differentiation matrices based on rational interpolants in the barycentric form. We then turn our attention to solving singularly perturbed boundary value problems (BVPs), making use of a technique developed by Tang and Trummer (boundary layer resolving spectral collocation, BLRSC, method) [34]. Here, the combination of the Floater-Hormann interpolants and BLRSC method is a new contribution. Finally, we look at systems of coupled singularly perturbed BVPs, though only using the standard Chebyshev points. We compare the BLRSC method (extended to coupled systems) to current methods in the relevant literature. A brief investigation into nonlinear systems is conducted as well and this is also a new contribution. We also pay attention to numerical issues and their remedies throughout the paper.

Chapter 2

Interpolation

2.1 Polynomial Interpolation

The basic idea of interpolation consists of using known values of a function to determine unknown ones in-between the known values. This typically involves creating a ”simple“ function that takes prescribed values at a finite set of discrete data points which can then be used to determine intermediate values.

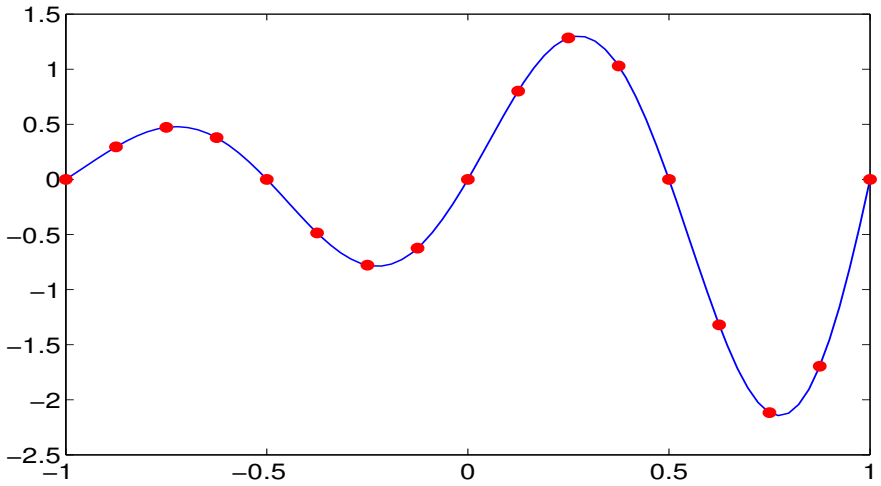

Figure 2.1: Illustrative example of interpolation. Red dots represent known values, through which a function (represented by the blue line) interpolates them, and then allows for evaluation at intermediate points.

This simple function is often a polynomial, which is easy to work with in problems such as

solving differential equations. Polynomial interpolation is a very useful starting approach since there always exists an unique interpolation polynomial of degree less than or equal to N for $N+1$ points, as described in the following well known theorem.

Theorem 1. *Given N +1 distinct nodes $x_0, x_1, \ldots, x_N$ and N +1 corresponding values $f_0, f_1, \ldots, f_N$, there exists an unique polynomial $p_N(x)$ of degree at most N for which*

$$p_N(x_j) = f_j \qquad \textit{for } j = 0, 1, \ldots, N.$$

There are a number of ways of constructing and representing this polynomial. One of the most common forms is the Lagrangian, which is

$$p_N(x) = \sum_{j=0}^{N} l_j(x) f_j \tag{2.1}$$

where

$$l_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^{N} \frac{x - x_k}{x_j - x_k}$$

therefore

$$l_j(x_i) = \begin{cases} 1 & \text{if } j = i \\ \\ 0 & \text{if } j \neq i \end{cases}$$

thus satisfying the interpolating condition. In this thesis, we will assume that the f_j are values of some underlying function and thus also use the notation $f(x_j)$.

Another common way of expressing this interpolating polynomial is the Newton form. The polynomial is expressed as

$$p_N(x) = \sum_{j=0}^{N} a_j \prod_{k=0}^{j-1} (x - x_k)$$

where the a_j are the divided differences

$$a_0 = [f_0] = f_0, \qquad a_j = [f_0, f_1, \ldots, f_j] = \frac{[f_1, f_2, \ldots, f_j] - [f_0, f_1, \ldots, f_{j-1}]}{x_j - x_0} = \ldots .$$

With just two points, the divided difference is

$$[f_0, f_1] = \frac{f_1 - f_0}{x_1 - x_0}$$

which is equal to the slope of the linear interpolant. There are also other ways of expressing this polynomial, as seen in [17]. This form of the polynomial has one obvious advantage, that being that

the addition of another point x_{N+1} is not as costly as in the Lagrangian form. This is due to the fact that in the Lagrangian form one needs to recalculate the entire polynomial. For the Newton form, one simply needs to calculate one new term. Also, aside from some pre-calculation (the *Newton tableau* of divided differences), this evaluation form of the interpolating polynomial only requires $O(N)$ FLOPS (floating-point operations). However, the Newton form has some disadvantages, one of which is that the nodes have to be ordered in a specific way. Certain orderings and large values of N can lead to instabilities [12]. Also, the $O(N^2)$ pre-calculations that have to be performed depend on the $f(x_j)$ values. This means that for fixed collocation points but new function values $f^{new}(x_j)$, one has to reconstruct the entire polynomial. The Lagrangian form which was already discussed is better than the Newton form in this regard, but an even better representation exists.

Another, and in many ways superior, representation is the barycentric form. Although this form has existed for many years, only recently has it garnered much attention. To obtain the barycentric form one needs to rearrange the Lagrangian form as follows

$$\begin{aligned} p_N(x) &= \sum_{j=0}^{N} \prod_{\substack{k=0 \\ k\neq j}}^{N} \frac{x-x_k}{x_j-x_k} f(x_j) \\ &= \prod_{k=0}^{N}(x-x_k) \sum_{j=0}^{N} \frac{1}{x-x_j} \prod_{\substack{k=0 \\ k\neq j}}^{N} \frac{1}{x_j-x_k} f(x_j) \\ &= L(x) \sum_{j=0}^{N} \frac{w_j}{x-x_j} f_j \end{aligned} \tag{2.2}$$

where

$$w_j := \prod_{\substack{k=0 \\ k\neq j}}^{N} \frac{1}{x_j-x_k} \quad \text{and} \quad L(x) = \prod_{k=0}^{N}(x-x_k). \tag{2.3}$$

This is the *first barycentric form* of the interpolating polynomial. Commonly, and in this thesis, the w_j are referred to as *weights*. This form allows one to evaluate $p_N(x)$ in $O(N)$ FLOPS, as opposed to $O(N^2)$ FLOPS in the Lagrangian form (assuming that one pre-calculates the weights, which normally requires $O(N^2)$ operations). As mentioned in [12], the barycentric form also allows one to more easily include additional points and is more stable than the Lagrangian form for a given point distribution. To add a new data point x_{N+1} and its corresponding value f_{N+1}, one needs to update the current weights and calculate the new weight as follows

$$\begin{aligned} w_j^{(new)} &= \frac{w_j^{(old)}}{x_j - x_{N+1}}, \qquad \text{for } j = 0, 1, \ldots, N \\ w_{N+1} &= \prod_{\substack{k=0 \\ k\neq N+1}}^{N} \frac{1}{x_{N+1}-x_k}. \end{aligned}$$

As mentioned before, Equation (2.2) is only the *first barycentric form*. To further simplify this form, one can use the polynomial interpolation of the function f(x)=1 (all $f_j = 1$)

$$p_N^*(x) = L(x) \sum_{i=0}^{N} \frac{w_j}{x - x_j} = 1.$$

Then one can divide $p_N(x)$ by $p_N^*(x)$, cancelling the $L(x)$ function from the numerator and denominator. This yields the *true barycentric form*

$$p_N(x) = \sum_{j=0}^{N} \frac{w_j}{x - x_j} f_j \Big/ \sum_{j=0}^{N} \frac{w_j}{x - x_j}. \tag{2.4}$$

The previously mentioned advantages still hold. Furthermore one should note that, in this form, one can multiply the weights by a constant without changing the interpolating function.

The weights have been defined in Equation (2.3) for any distribution of interpolation points. For certain common distributions (such as equidistant and Chebyshev) the computation of the weights can be further simplified. For equidistant points the weights are

$$w_j = (-1)^j \binom{N}{j}.$$

For Chebyshev points of the second kind ($x_j = \cos \frac{j\pi}{N}, j = 0, 1, \ldots, N$), the weights are

$$w_j = \begin{cases} (-1)^j & j = 1, 2, \ldots N-1 \\ \\ \dfrac{(-1)^j}{2} & j = 0, N. \end{cases}$$

The term Chebyshev points will be used to refer to Chebyshev points of the second kind in this thesis.

2.2 Rational Interpolation

One of the main deficiencies of polynomial interpolation is the limited interval of convergence in some instances, such as in Runge's famous example, where one is interpolating

$$f(x) = \frac{1}{1 + x^2} \quad x \in [-5, 5]$$

with equidistant points. For this function, the polynomial interpolant will fail to converge near the ends of the interval. As seen in [35], using potential theory illustrates how one can visualize these regions of convergence in the complex plane. With equidistant points, these regions look like American footballs (prolate spheroids with pointed ends), with the largest region of convergence

not including any poles. Therefore, the convergence region for $f(x)$ as given above will not include the entire interval $[-5, 5]$. One way to remedy this is to choose a different point distribution, as is explained in [35], where Chebyshev points are used. However, another method of dealing with this problem is to use rational, instead of polynomial, interpolants.

Given a set of points $\{x_0, x_1, \ldots, x_N\}$ in an interval $x \in [a, b]$ the basic rational interpolant is the quotient of two polynomials

$$r(x) = \frac{p_{\tilde{M}}(x)}{q_{\tilde{N}}(x)}, \qquad r(x_j) = f_j$$

where $\tilde{N}, \tilde{M}$ represent the degrees of the polynomials. Classical rational interpolation stipulates that one chooses $\tilde{M}$ and $\tilde{N}$ such that $\tilde{M} + \tilde{N} = N$. One can, however, approach rational interpolation differently. As Schneider and Werner did in their paper [33], one can define the rational interpolant by its barycentric form. For the data $r_N(x_j) = f_j$, let

$$r_N(x) = \sum_{j=0}^{N} \frac{w_j}{x - x_j} f_j \Bigg/ \sum_{j=0}^{N} \frac{w_j}{x - x_j} \tag{2.5}$$

where

$$\lim_{x \to x_j} r_N(x) = f_j \quad \text{if} \;\; w_j \neq 0 \;\; \forall j.$$

The rational interpolant described in (2.5) is linear in f_j, since its denominator does not depend on the function values f_i. Conversely, a non-linear rational interplant would have some dependence on the function values in both the numerator and denominator. This dependence usually arises from the weights w_i depending on the function values (see [8] for more information). We only consider linear rational interpolants.

Benefits of this barycentric form were already mentioned in regards to polynomial interpolation. However, with rational interpolation this form has another benefit. For polynomial interpolation, the weights w_j are determined by the distribution of points; with rational interpolants, they no longer have this restriction. This allows one to not only choose how the points are distributed, but also how to specify the weights.

Specifying the weights is a non-trivial task, as in addition to optimizing the rational interpolant (perhaps in the sense of minimizing error in the max-norm) one also wants to avoid unattainable points and poles in the interval under consideration. In this barycentric form, one can easily avoid unattainable points by utilizing the simple theorem (Theorem 2) given by Schneider and Werner, [33].

Theorem 2. *A point (x_i, f_i) is unattainable if and only if $w_i = 0$.*

Aside from avoiding unattainable points, one can also control the existence of poles. For a given interval $[a, b]$, one needs to ensure, as mentioned by Berrut and Mittelmann [11], that not only do

the weights alternate in sign, but also that the weights are "similar enough" in magnitude (such that $|w_j/(x - x_j)|$ decreases on both sides of any x). The proof below is an extension of a proof by Berrut [7], where he considers the specific case of $w_k = (-1)^k$. Here, the proof is extended beyond the case where the weights are one in absolute value, but uses many of the ideas in Berrut's proof.

Theorem 3. *Let $x_0, x_1, \ldots, x_N$ be $N+1$ distinct points with $x_0 < x_1 < \ldots < x_N$. For clarity, let $I_0 := (-\infty, x_0)$, $I_k := (x_{k-1}, x_k)$ for $k = 1, 2, ...N$, and $I_{N+1} := (x_N, \infty)$. Let $w_0, w_1, \ldots, w_N$ be the $N+1$ corresponding weights, such that:*

- $w_k \neq 0$ *for* $k = 0, 1, \ldots, N$
- $sign(w_k) = -sign(w_{k-1})$ *for* $k = 1, 2, \ldots N$ *(the weights are alternating in sign)*
- *For an arbitrary interval I_a, ($a = 0, 1, \ldots, N$),*

$$\left| \sum_{\substack{0 \leq k \leq N-a \\ k \text{ is even}}} \frac{w_{a+k}}{x - x_{a+k}} \right| > \left| \sum_{\substack{1 \leq k \leq N-a \\ k \text{ is odd}}} \frac{w_{a+k}}{x - x_{a+k}} \right|$$

and

$$\left| \sum_{\substack{1 \leq k \leq a \\ k \text{ is odd}}} \frac{w_{a-k}}{x - x_{a-k}} \right| > \left| \sum_{\substack{0 \leq k \leq a \\ k \text{ is even}}} \frac{w_{a-k}}{x - x_{a-k}} \right|$$

Then the rational interpolant $r_N(x)$, where

$$r_N(x) = \sum_{j=0}^{N} \frac{w_j}{x - x_j} f_j \Big/ \sum_{j=0}^{N} \frac{w_j}{x - x_j}$$

has no poles in $\mathbb{R}$.

Proof. We shall prove the above statement by showing that

$$\sum_{k=0}^{N} \frac{w_k}{x - x_k} \neq 0 \qquad k = 0, 1, \ldots, N. \tag{2.6}$$

Looking at the terms in (2.6) we see the term corresponding to the interpolating point x_k is a hyperbola with a vertical asymptote at $x = x_k$. The intervals defined previously, I_k, denote the intervals between these asymptotes. Also, we define

$$s_k := \begin{cases} \dfrac{w_k}{x - x_k} & x < x_k \\ 0 & x > x_k \end{cases} \qquad\qquad r_k := \begin{cases} 0 & x < x_k \\ \dfrac{w_k}{x - x_k} & x > x_k \end{cases}$$

and consider s_k and r_k separately. We have

$$\sum_{k=0}^{n} \frac{w_k}{x - x_k} = s(x) + r(x)$$

where

$$s(x) := \sum_{k=0}^{N} s_k(x) \qquad\qquad r(x) := \sum_{k=0}^{N} r_k(x).$$

For $x \in I_a$,

$$s(x) = \sum_{k=a}^{N} s_k(x) \qquad\qquad r(x) = \sum_{k=0}^{a-1} r_k(x).$$

Therefore, due to the assumptions, the sign of the resulting sum of these series will be determined by the sign of the $s_a(x)$ and $r_{a-1}(x)$ terms respectively. Thus,

$$sign(s(x)) = sign(s_a) = -sign(w_a) = sign(w_{a-1}) = sign(r_{a-1}) = sign(r(x))$$

which implies that the sum $s(x) + r(x)$ will either be positive or negative, but not zero.

□

2.3 Floater-Hormann Family of Barycentric Rational Interpolants

2.3.1 Construction

Two of problems with general linear rational interpolation for arbitrary points (including equidistant points) are the slow rate of convergence and possibility of poles. However, Floater and Hormann [18] show that there exists a family of linear rational interpolants (which we will refer to as FH interpolants) which have arbitrarily high rates of convergence for most point distributions, including equidistant, which do not include any poles in $\mathbb{R}$ (which greatly reduces the issue of poles, even for general barycentric rational interpolants). They construct the interpolant as follows. Given an integer N, choose an integer d, $0 \leq d \leq N$. For each $i = 0, 1, 2, \ldots, N - d$ let p_i be the unique polynomial of degree at most d which interpolates the function f at nodes $x_i, x_{i+1}, \ldots, x_{i+d}$. Then let

$$r(x) = \frac{\sum_{i=0}^{N-d} \lambda_i(x) p_i(x)}{\sum_{i=0}^{N-d} \lambda_i(x)} \tag{2.7}$$

where

$$\lambda_i(x) = \frac{(-1)^i}{(x - x_i)(x - x_{i+1}) \ldots (x - x_{i+d})}.$$

As can be seen, with $d = N$, this simply represents polynomial interpolation. Floater and Hormann describe this rational interpolant r as "a blend of the polynomial interpolants $p_0, \ldots, p_{N-d}$ with the $\lambda_0, \ldots, \lambda_{N-d}$ acting as the blending functions." (Floater and Hormann, 2007, p. 317) Each rational interpolant described here, one for each d, has no poles in $\mathbb{R}$. Also, for a given $d \geq 1$ and $f \in C^{d+2}[a, b]$, $r(x)$ has approximation order $O(h^{d+1})$ as $h \to 0$, where h is the maximum spacing between any two adjacent nodes.

Floater and Hormann also show that this can be represented in barycentric form. One starts by writing the polynomial interpolants $p_i(x)$ in Equation (2.7) in the Lagrange form,

$$p_i(x) = \sum_{k=i}^{i+d} \prod_{\substack{j=i \\ j \neq k}}^{i+d} \frac{x - x_k}{x_i - x_k} f(x_k). \tag{2.8}$$

Then, substituting Equation (2.8) into Equation (2.7) and using the definition of $\lambda_i(x)$ we have

$$\begin{aligned} \sum_{i=0}^{N-d} \lambda_i(x) p_i(x) &= \sum_{i=0}^{N-d} (-1)^i \sum_{k=i}^{i+d} \frac{1}{x - x_k} \prod_{\substack{j=i \\ j \neq k}}^{i+d} \frac{1}{x_k - x_j} f(x_k) \\ &= \sum_{k=0}^{N} \frac{w_k}{x - x_k} f(x_k) \end{aligned}$$

with

$$w_k = \sum_{i \in J_k} (-1)^i \prod_{\substack{j=i \\ j \neq k}}^{i+d} \frac{1}{x_k - x_j} \tag{2.9}$$

$$J_k := \{i \in \{0, 1, \ldots, N - d\} : k - d \leq i \leq k\}.$$

Following this same process, and using the fact that

$$1 = \sum_{k=i}^{i+d} \prod_{\substack{j=i \\ j \neq k}}^{i+d} \frac{x - x_k}{x_i - x_k}$$

one can similarly rewrite the denominator of Equation (2.7) as

$$\sum_{k=0}^{N-d} \lambda_i(x) = \sum_{k=0}^{N} \frac{w_k}{x - x_k}.$$

Therefore, one now has the following rational interpolant in barycentric form

$$r_N(x) = \frac{\displaystyle\sum_{k=0}^{N} \frac{w_k}{x - x_k} f(x_k)}{\displaystyle\sum_{k=0}^{N} \frac{w_k}{x - x_k}} \tag{2.10}$$

with weights w_k defined in Equation (2.9).

Floater and Hormann also give another form of the weights, that being

$$w_k = (-1)^{k-d} \sum_{i \in J_k} \prod_{\substack{j=1 \\ j \neq k}}^{i+d} \frac{1}{|x_k - x_j|}. \tag{2.11}$$

One can see how this is obtained by looking at the factors of each term. First we see that for each term in the sum, there can be factors before and after the k value which is omitted. For example, if $i = 0$, $d = 6$, $k = 4$, there would be two factors after the k value, or

$$\frac{1}{x_4 - x_5} \frac{1}{x_4 - x_6}.$$

These would be negative. All factors before the k value are positive and therefore will not affect the sign of the term.

We now look at a few cases. If i is even, and k and d are both even or both odd, then there will be an even number of negative factors, and the product will be positive. If, however, k is even and d is odd (or k is odd and d is even) then there will be an odd number of negative factors, and the resulting product will be negative.

Alternatively, if i is odd and k and d are both even or both odd, then there will be an odd number of negative products, and the resulting product (with the $(-1)^i$ considered) will be be positive. If k is odd and d is even (or k is even and d is odd) then there will be an even number of negative factors, resulting in negative product. This can all be summarised as follows

$$\begin{aligned} w_k &= \sum_{i \in J_k} (-1)^i \prod_{\substack{j=i \\ j \neq k}}^{i+d} \frac{1}{x_k - x_j} \\ &= \sum_{i \in J_k} (-1)^i (-1)^{i+(k-d)} \prod_{\substack{j=i \\ j \neq k}}^{i+d} \frac{1}{|x_k - x_j|} \\ &= (-1)^{k-d} \sum_{i \in J_k} \prod_{\substack{j=1 \\ j \neq k}}^{i+d} \frac{1}{|x_k - x_j|}. \end{aligned}$$

This form is especially useful in easily seeing that the weights alternate in sign.

In [18] Floater and Hormann also give a simplified formula for the weights of equispaced points

$$w_k = (-1)^{k-d} \sum_{i \in J_k} \binom{d}{k-i}. \tag{2.12}$$

To re-emphasize, these weights have a new degree of freedom compared to the polynomial interpolant. Given a certain distribution of points, one can now choose the weights. In the case of the FH interpolants, aside from the interpolation points, the weights depend on the value of d. Floater

and Hormann discuss some of the properties of this parameter d, such as its optimal value (in minimizing the max-norm of an interpolation problem) for specific functions and various values of N.

Floater and Hormann gave a sampling of the absolute value of the weights for the first few values of d. They have divided all the weights by $d!h^d$, so that they are all integer values:

$$\begin{aligned} 1, 1, \ldots, 1, 1 \qquad & d = 0 \\ 1, 2, 2, \ldots, 2, 2, 1 \qquad & d = 1 \\ 1, 3, 4, 4, \ldots, 4, 4, 3, 1 \qquad & d = 2 \\ 1, 4, 7, 8, 8, \ldots, 8, 8, 7, 4, 1 \qquad & d = 3 \\ 1, 5, 11, 15, 16, 16, \ldots, 16, 16, 15, 11, 5, 1 \qquad & d = 4 \end{aligned} \tag{2.13}$$

It may be of interest to note the similarity of this pattern to Pascal's Triangle.

As stated in [13], one has an even simpler form of the weights, which is (where we are only looking at the absolute value of the integer form, and for $N \geq 2d$)

$$|w_k| = \begin{cases} \sum_{j=0}^{k} \binom{d}{k} & k \leq d \\ 2^d & d \leq k \leq N - d \\ |w_{N-k}| & k \geq N - d \end{cases} \tag{2.14}$$

As mentioned in [13], using the pattern in (2.13) or Equation (2.14) for $N < 2d$ results in incorrect weights. This is seen in the following example, where the absolute value of the weights for $0 \leq d \leq 4$ are given for $N = 4$:

$$\begin{aligned} 1, 1, 1, 1, 1 \qquad & d = 0 \\ 1, 2, 2, 2, 1 \qquad & d = 1 \\ 1, 3, 4, 3, 1 \qquad & d = 2 \\ 1, 4, 6, 4, 1 \qquad & d = 3 \\ 1, 4, 6, 4, 1 \qquad & d = 4 \end{aligned}$$

2.3.2 Properties

No Poles in $\mathbb{R}$, Floater-Hormann Method

In [18] Floater and Hormann show the absence of poles for the family of rational interpolants described by first rewriting the rational interpolant by multiplying both the numerator and denominator

of (2.7) by $(-1)^{N-d}(x-x_0)\dots(x-x_N)$, yielding

$$r_N(x) = \frac{\sum_{j=0}^{N-d} \mu_j(x) p_j(x)}{\sum_{j=0}^{N-d} \mu_j(x)} \tag{2.15}$$

with

$$\mu_j(x) = (-1)^{N-d}(x-x_0)\dots(x-x_N)\lambda_j(x) \tag{2.16}$$

$$= \prod_{i=0}^{j-1}(x-x_i) \prod_{k=j+d+1}^{N}(x_k-x). \tag{2.17}$$

They then proceed to show that, given a value of $d \in [0, N]$, the summation $\sum_{j=0}^{N-d} \mu_j$ is positive for all values of x. See [18] for details of proof.

Convergence Rate

The following theorems are found in [18]. $\|\cdot\|$ represents the maximum absolute value over $[a, b]$.

Theorem 4. *Suppose $d \geq 1$ and $f \in C^{d+2}[a, b]$, and let*

$$h := \max_{0 \leq i \leq N-1}(x_{i+1} - x_i).$$

Then

$$\|r_N - f\| \leq h^{d+1}(b-a)\frac{\|f^{d+2}\|}{d+2} \qquad \text{if } N-d \text{ is odd}$$

$$\|r_N - f\| \leq h^{d+1}\left((b-a)\frac{\|f^{d+2}\|}{d+2} + \frac{\|f^{d+1}\|}{d+1}\right) \qquad \text{if } N-d \text{ is even.}$$

Theorem 5. *Suppose $d = 0$ and $f \in C^2[a, b]$ and let*

$$\beta := \max_{1 \leq i \leq N-2} \min\left\{\frac{x_{i+1} - x_i}{x_i - x_{i-1}}, \frac{x_{i+1} - x_i}{x_{i+2} - x_{i+1}}\right\}$$

(and where h is defined as in Theorem 4). Then

$$\|r_N - f\| \leq h(1+\beta)(b-a)\frac{\|f''\|}{2} \qquad \text{if } N \text{ is odd}$$

$$\|r_N - f\| \leq h(1+\beta)\left((b-a)\frac{\|f''\|}{2} + \|f'\|\right) \qquad \text{if } N \text{ is even.}$$

The proofs of these theorems use, in part, the Newton divided difference form to derive the error bounds. See [18] for complete proof.

2.3.3 Numerical Results

The results of [18], wherein three functions were tested to see if the experimentally predicted convergence rate was achieved in practice, are reproduced here. Also investigated and reproduced here are various optimal d values for certain values of N (for the function $1/(1+x^2)$).

Table 2.1: Convergence Rates

	$f_1 = \frac{1}{(1+x^2)}, d = 3$		$f_2 = \sin(x), d = 4$		$f_3 = \lvert x \rvert, d = 3$	
N	Error	Order	Error	Order	Error	Order
10	6.88×10^{-2}		1.75×10^{-2}		1.79×10^{-1}	
20	2.82×10^{-3}	4.6	3.87×10^{-4}	5.5	9.01×10^{-2}	1.0
40	4.29×10^{-6}	9.4	7.10×10^{-6}	5.8	4.52×10^{-2}	1.0
80	5.10×10^{-8}	6.4	1.32×10^{-7}	5.8	2.27×10^{-2}	1.0
160	3.00×10^{-9}	4.1	2.65×10^{-9}	5.6	1.13×10^{-2}	1.0
320	1.82×10^{-10}	4.0	5.99×10^{-11}	5.5	5.68×10^{-3}	1.0
640	1.11×10^{-11}	4.0	1.51×10^{-12}	5.3	2.84×10^{-3}	1.0

Reproduced results of Floater and Hormann, [18]. For all functions we have $x \in [-5, 5]$ and the maximum absolute error was calculated using $3N$ equispaced points. The order of convergence is computed in the standard way.

Table 2.1 shows the estimated convergence rates for three functions. As predicted in [18], these greatly depend on the smoothness of the function. For the first two functions we have $f_1 \in C^{3+2}$ and $f_2 \in C^{4+2}$, and thus we can use the originally stated rate of convergence $O(h^{d+1})$, where the expected rates of 4 and 5 (respectively) are very close to the calculated rates. For the third function, $f_3 \in C^0$, and as is seen here, the convergence rate is 1, as opposed to 4 for a C^5 function.

Table 2.2: Optimal d Values

N	d	Error
10	1	3.60×10^{-2}
20	2	1.53×10^{-3}
40	4	4.27×10^{-6}
80	8	2.04×10^{-10}
160	10	1.78×10^{-15}

Reproduced results of Floater and Hormann, [18]. Optimal d values for the function $f_1 = 1/(1+x^2)$ with $x \in [-5, 5]$.

In Table 2.2 we see the optimal values of d for various values of N for Runge's example. More will be said about finding optimal d values, but here we simply look for the minimum of the maximum absolute error. This was done using $10N$ points (compared to the original $N+1$ interpolation nodes) to compare the interpolant and the original function. One can see that d increases with N; however, there is no simple expression to calculate the optimal d value.

2.4 Extended Floater-Hormann Method

Although the family of rational interpolants described by Floater and Hormann are much better than polynomial and general rational interpolants at interpolation, Klein and Berrut in [10] and Klein in [20] saw room for improvement. They noted that, when looking at the Lebesgue constant (see appendix) there was noticeable error in the interpolation function near the end points when using equidistant points. To remedy this, an extension of the Floater-Hormann family of interpolants was given (referred to as EFH). The idea is to extend the region in which the interpolant is constructed by d points on either end while maintaining the interval on which the interpolant is to be used. This eliminates the error at the ends.

For fixed N and d, one constructs the EFH interpolants by first choosing an $\tilde{N} \ll N$, and a $\tilde{d} \leq \tilde{N}$. Next, one needs to extend the interval $[a, b]$ by adding d points on each side, with the original spacing h. Let these points be $x_{-d}, x_{-d+1}, \ldots, x_{-1}$ and $x_{N+1}, x_{N+2}, \ldots, x_{N+d}$. Now, one needs to approximate the $2d$ new function values $f_{-d}, f_{-d+1}, \ldots, f_{-1}$ and $f_{N+1}, f_{N+2}, \ldots, f_{N+d}$, which correspond to the new points just created. This will be done in steps. First, one will calculate $r^{(k)}_{\tilde{N}}[f](x_0)$, $k = 1, 2, \ldots, \tilde{d}$, or the rational interpolant of the values $f_0, f_1, \ldots, f_{\tilde{N}}$ evaluated at x_0 with the parameter $\tilde{d}$. Also, one will calculate $r^{(k)}_{\tilde{N}}[f](x_N)$, $k = 1, 2, \ldots, \tilde{d}$ using $f_{N-\tilde{N}}, f_{N-\tilde{N}+1}, \ldots, f_N$.

Next, one can define the extended function values $\tilde{f}_j$ as

$$\tilde{f}_j := \begin{cases} f_0 + \sum_{k=1}^{\tilde{d}} r_{\tilde{N}}^{(k)}[f](x_0)\frac{(x_j - x_0)^k}{k!} & -d \leq j \leq -1 \\ f_j & 0 \leq j \leq N \\ f_N + \sum_{k=1}^{\tilde{d}} r_{\tilde{N}}^{(k)}[f](x_N)\frac{(x_j - x_N)^k}{k!} & (N+1) \leq j \leq (N+d). \end{cases}$$

Using these extended values $(\tilde{x_j}, \tilde{f}_j)$ one can now define the extended interpolant as

$$r_{\tilde{N}}[f](x) := \frac{\sum_{k=-d}^{N+d} \frac{w_k}{x - x_k}\tilde{f}_k}{\sum_{k=-d}^{N+d} \frac{w_k}{x - x_k}}$$

where this interpolant is only valid for $x \in [a, b]$. This construction relies upon equidistant points, and is not easily adapted to non-equidistant points (such as Chebyshev).

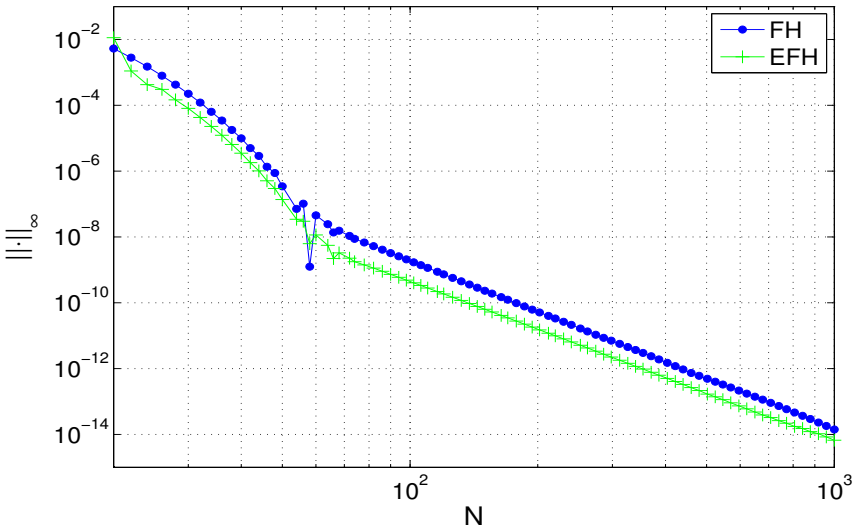

Figure 2.2: Maximum absolute differences between interpolants and the function $f(x) = \frac{1}{1+x^2}$ on the interval $x \in [-5, 5]$. For each value of N, the interpolants and function are evaluated at 2000 equispaced points and the error is then measured using these values. Parameters used are: $20 \leq N \leq 1000$ and N is even, $d = 4,\ \tilde{N} = 11,\ \tilde{d} = 7$.

As can be seen in Figure 2.2, results are reproduced almost exactly as in the literature. They

indicate that the EFH interpolants are only slightly better in terms of reducing maximum absolute error. As seen in Figure 2.3 though, the EFH interpolants are more stable, in the sense that if the function values are perturbed the choice of the parameter d does not yield large errors. The original FH interpolant can become very sensitive to perturbed data for larger values of d.

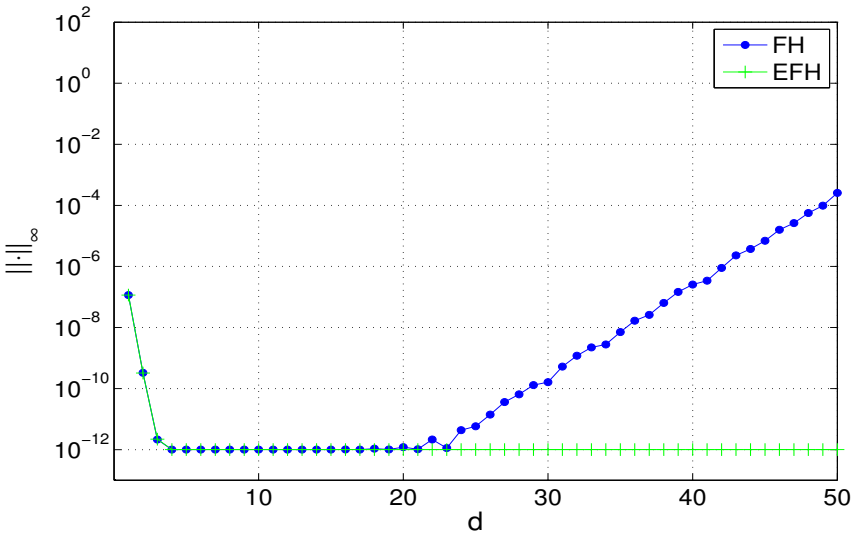

Figure 2.3: Maximum absolute differences between interpolants and the function $f(x) = \frac{1}{1+x^2}$ on the interval $x \in [-5, 5]$. For each value of d, the interpolants and function are evaulated at 2000 equispaced points and the error is then measured using these values. Parameters: $0 \leq d \leq 50$, $N = 1000$ $\tilde{N} = 11$, $\tilde{d} = 7$. Perturbations of $f + 10^{-12}$ and $f - 10^{-12}$ were used, alternating for each value of d.

Of note, the results obtained here are similar to the results obtained in [10], [20], except that in these sources, the error in the FH interpolant begins increasing linearly after $d \approx 4$. The difference could possibly be attributed to a more careful implementation of the FH algorithm in Matlab.

Chapter 3

Differentiation Matrices

3.1 Derivatives of Linear Rational Interpolants in Barycentric Form

3.1.1 Formulas and Matrices

The usefulness of the barycentric form, aside from the previously mentioned reasons, is mainly due to the work done by Schneider and Werner. In their 1986 paper (see [33]) they give simple formulas for differentiating rational interpolants in barycentric form, both at the nodes and at non-nodal points. The formula for the k^{th} derivative using (2.5) is, at non-nodal points and nodes respectively,

$$\frac{r_N^{(k)}(\xi)}{k!} = \sum_{i=0}^{N} \frac{w_i}{\xi - x_i} r_N\left[(\xi)^k, x_i\right] \Bigg/ \sum_{i=0}^{N} \frac{w_i}{\xi - x_i} \qquad \text{for } \xi \neq x_i, i = 0, 1, \ldots, N, k \geq 0$$

$$\frac{r_N^{(k)}(x_j)}{k!} = -\left(\sum_{\substack{i=0 \\ i \neq j}}^{N} w_i r_N\left[(x_j)^k, x_i\right] \right) \Big/ w_j \qquad \text{for } 0 \leq j \leq N, k \geq 1$$

where $r_N[(\alpha)^k, x_i]$ represents a divided difference with a k-fold argument of α. A number of sources ([7], [10], [21], as well as others) have defined differentiation matrices for the derivatives of $r_N(x)$ at

the nodes. For the 1^{st} and k^{th} derivatives, they are

$$D_{ij}^{(1)} := \begin{cases} \dfrac{w_j}{w_i}\dfrac{1}{x_i - x_j}, & i \neq j \\ \\ -\displaystyle\sum_{\substack{l=0 \\ l\neq i}}^{N} D_{il}^{(1)}, & i = j \end{cases} \tag{3.1}$$

(3.2)

$$D_{ij}^{(k)} := \begin{cases} \dfrac{k}{x_i - x_j}\left(\dfrac{w_j}{w_i}D_{ii}^{(k-1)} - D_{ij}^{(k-1)}\right), & i \neq j \\ \\ -\displaystyle\sum_{\substack{l=0 \\ l\neq i}}^{N} D_{il}^{(k)}, & i = j. \end{cases} \tag{3.3}$$

Klein and Berrut give an alternate expression for evaluating the non-diagonal elements of the k^{th} order differentiation matrix, as

$$D_{ij}^{(k)} = k\left(D_{ij}^{(1)}D_{ii}^{(k-1)} - \frac{D_{ij}^{(k-1)}}{x_i - x_j}\right) \qquad \text{for } i \neq j$$

This allows one to forgo redoing the w_j/w_i operation already performed in finding the first order differentiation matrix (for faster performance).

Due to the extended $\tilde{x_j}$ and $\tilde{f_j}$ values, where $-d \leq j \leq N + d$, more attention must be given to the EFH differentiation matrix. One computes the $(N + 2d + 1) \times (N + 2d + 1)$ differentiation matrix and then uses all columns ***only*** from the $(d + 1)^{th}$ row to the $(N + d + 1)^{th}$ row, resulting in the $(N + 1) \times (N + 2d + 1)$ matrix

$$\tilde{D}^{(k)} = \begin{bmatrix} D_{0,-d}^{(k)} & \cdots & \cdots & D_{0,N+d}^{(k)} \\ \vdots & & & \vdots \\ D_{N,-d}^{(k)} & \cdots & \cdots & D_{N,N+d}^{(k)} \end{bmatrix} \tag{3.4}$$

which can then be used to compute

$$\left.\frac{d^{(k)}f(x)}{dx^{(k)}}\right|_{x_i} \approx \frac{d^{(k)}}{dx^{(k)}}\tilde{r}_N[f](x_i) = \sum_{j=-d}^{N+d} D_{ij}^{(k)}\tilde{f}_j = \tilde{D}^{(k)}\tilde{f}$$

where the "weights" $D_{ij}^{(k)}$ are the entries of the non-square differentiation matrix in (3.4).

3.1.2 Convergence Rates of FH and EFH derivatives

In [9], Berrut, Floater, and Klein prove convergence rates of the first and second derivatives and speculate about convergence rates of higher order derivatives. In the following two theorems, C is a constant depending on the blending parameter d, derivatives of the function f, and the spacing between nodes. For the first and second derivatives, the error at the nodes is as follows:

Theorem 6. *For the FH interpolants, if $d \geq k-1$ and $f \in C^{d+1+k}[a,b]$, then*

$$|e^{(k)}(x_j)| \leq Ch^{d-k+1} \qquad 0 \leq j \leq N, \quad k = 1, 2$$

where

$$e(x) := f(x) - r_N(x).$$

Although they did not prove it (due to the complexity of the formulas for $e^{(k)}, k > 2$), the authors of [9] speculate that this error bound remains true for $k > 2$. Tables 3.1 and 3.2 below show computed convergence rates for both the FH and EFH derivatives. Despite the fact that Berrut *et al.* only proved Theorem 6 in [9] for the FH interpolants, one can see that the EFH method converges at about the same rate, but is slightly more accurate in the first derivative and much more accurate in the second derivative (especially for larger N). This can be seen more clearly in Figure 3.1.

Table 3.1: Error and Convergence Rates, FH derivatives

N	1^{st} Derivative		2^{nd} Derivative	
	Error	Order	Error	Order
20	5.2394e-03		4.4661e-02	
40	1.9325e-04	4.8	3.3298e-03	3.7
80	7.2223e-06	4.7	2.5130e-04	3.7
160	2.9347e-07	4.6	2.0619e-05	3.6
320	1.3345e-08	4.5	1.8922e-06	3.4
640	6.7722e-10	4.3	1.9344e-07	3.3

Max-norm errors and numerical convergence rates for $\sin(x), x \in [-5, 5]$ using the FH differentiation matrices with $d = 4$ and equidistant points.

Table 3.2: Error and Convergence Rates, EFH derivatives

N	1^{st} Derivative		2^{nd} Derivative	
	Error	Order	Error	Order
20	6.1165e-04		2.1454e-03	
40	2.3537e-05	4.7	6.8857e-05	5.0
80	4.7902e-07	5.6	1.6357e-06	5.4
160	4.9421e-08	3.3	3.4414e-08	5.6
320	3.7181e-09	3.7	2.1629e-09	4.0
640	2.5153e-10	3.9	1.4310e-10	3.9

Max-norm errors and numerical convergence rates for $\sin(x), x \in [-5, 5]$ using the EFH differentiation matrices with $d = 3, \tilde{N} = 11, \tilde{d} = 7$.

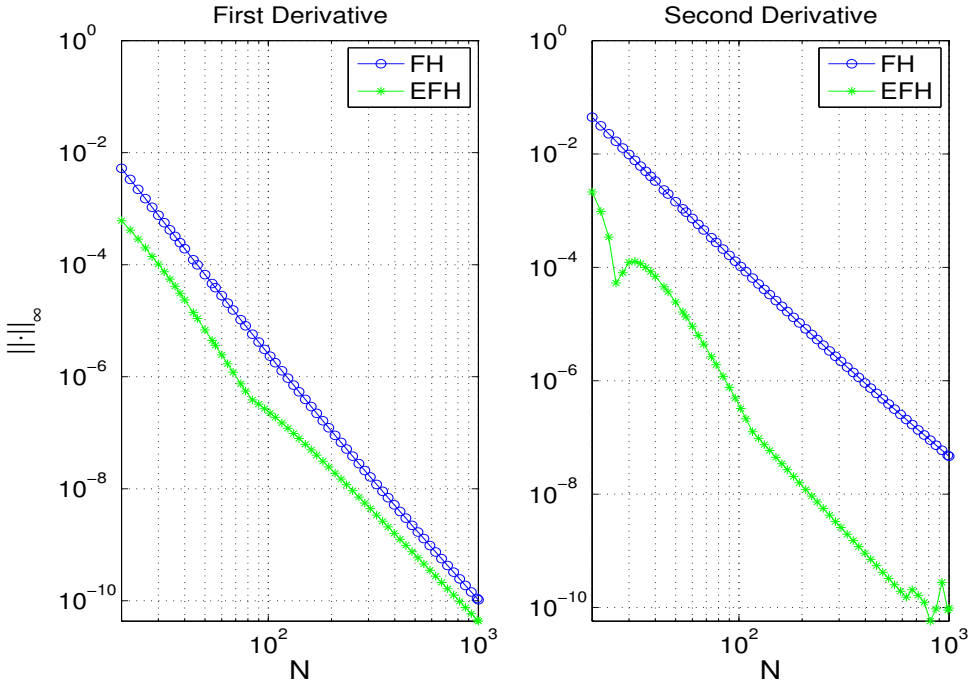

Figure 3.1: Log-log plot of the max-norm error in first and second derivatives of FH and EFH interpolants of the function $f(x) = \sin(x)$. Also $20 \leq N \leq 1000$, $d = 4$, $\tilde{N} = 11$, $\tilde{d} = 7$, $x \in [-5, 5]$.

As was seen in [20] and reproduced here in Figure 3.1, the EFH method gives a better approximation of the first and second derivatives of $\sin(x)$, the difference being especially noticeable in the second derivative. The results in [20] for the second derivative of the EFH are smoother than seen here. However, in [20], the second derivative of the EFH only achieved an error of about 10^{-9}, whereas here we achieve an error of approximately 10^{-10}. This difference may be attributable to better implementation in Matlab.

3.2 Chebyshev Differentiation Matrices

Although not immediately relevant, in subsequent chapters Chebyshev differentiation matrices, D_N, (or differentiation matrices based upon polynomial interpolation of Chebyshev points) will be used in solving boundary value problems. These will be referred to as the standard Chebyshev differentiation matrices. The following formulas are found in many references, including [35],

$$(D_N)_{00} = -(D_N)_{NN} = \frac{2N^2+1}{6}$$

$$(D_N)_{jj} = \frac{-x_j}{2(1-x_j^2)}, \quad j = 1, \ldots, N-1$$

$$(D_N)_{ij} = \frac{c_i}{c_j}\frac{(-1)^{i+j}}{x_i - x_j}, \quad i \neq j,\ i,j = 1, \ldots, N-1$$

where $c_i = 1$ for $i = 1, \ldots, N-1$ and $c_0 = c_N = 2$. One can also use the negative sum trick with sorting (see appendix) for the computation of the diagonal elements.

It is interesting here to compare the Chebyshev differentiation matrix just described with the differentiation matrix obtained from the FH interpolants with $d = N$. These two matrices should be the same, since using $d = N$ is just polynomial interpolation. Looking at the first and second differentiation matrices, we see the following maximum differences below:

Table 3.3: Maximum Absolute Relative Differences of entries between Chebyshev Differentiation Matrices

N	$D^{(1)}$	$D^{(2)}$
8	2.704e-16	1.035e-15
16	7.100e-15	2.390e-14
32	1.601e-14	5.288e-14
64	1.726e-14	7.047e-14
128	8.341e-14	2.138e-13
256	3.998e-13	9.967e-13
512	1.196e-12	3.025e-12
1024	5.057e-12	1.527e-11

Maximum absolute relative differences differences between first and second differentiation matrices from the FH interpolants with $d = N$ and from polynomial interpolation. Specifically, $\left(max_{i,j}|(D^{(k)}_{std})_{i.j} - (D^{(k)}_{FH})_{i,j}|\right) / max_{i,j}|(D^{(k)}_{std})_{i,j}|$,

These matrices should be identical, but numerical effects based on the different ways of computing each matrix lead to small differences. The maximum differences between the differentiation matrices occurred in the top left and bottom right corners. In each differentiation matrix, this is also where the largest values occur (usually within a 4×4 submatrix). For instance, in the case of $N = 1024$, the maximum difference between the first order and second order differentiation matrices occurs in the first row second column; this is also where the largest values in each matrix occur.

Chapter 4

Boundary Value Problems (BVPs)

4.1 Two-Point Linear BVPs

After observing the good interpolation and differentiation accuracy of the FH interpolants, one possible extension is to investigate their use in solving two-point boundary value problems (BVPs). Although the EFH interpolants share this high accuracy in interpolation and differentiation, they are not well suited for BVPs for a couple of reasons. Firstly, their differentiation matrices are rectangular, which can cause numerical issues. Secondly, they require an extension of the function values which could be especially troublesome for singularly perturbed BVPs (which will be investigated). Thus, testing is performed solely on the FH interpolants. We first look at their performance in solving second order linear BVPs of the form

$$u''(x) + p(x)u'(x) + q(x)u(x) = f(x)$$

with coefficients $p(x), q(x), f(x)$, and boundary conditions $u(-1) = \alpha$, $u(1) = \beta$.

For the discretization of the differential equation, the aforementioned differentiation matrices were used as follows

$$\begin{aligned} D^{(2)}u + diag(p)D^{(1)}u + diag(q)u &= f \\ \Big(D^{(2)} + diag(p)D^{(1)} + diag(q)\Big)\, u &= f \\ Au &= f \end{aligned}$$

One example is BVP_1

$$u'' + 2u' + u = 0, \;\; u(-1) = -1, \; u(1) = 1 \tag{4.1}$$

with exact solution

$$u(x) = \frac{1}{2}e^{-x-1}\Big(xe^2 + e^2 + x - 1\Big).$$

Using $N = 64$ and $d = 1$, the FH method (the spectral collocation method using FH interpolation) is tested to see how well it performs in regards to maximum absolute error. Figure 4.1 shows the numerical solution using the FH method and the exact solution.

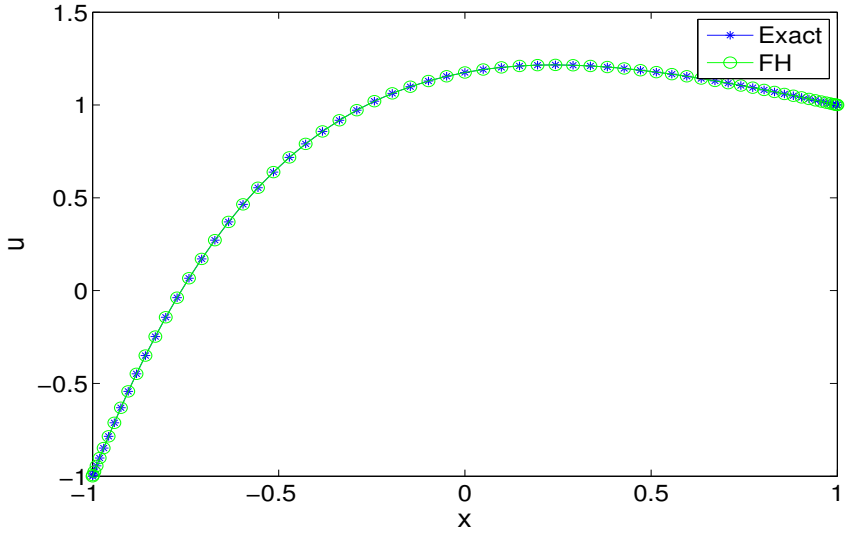

Figure 4.1: Solution profiles (FH approximation as well as exact) for the BVP $u'' + 2u' + u = 0$ with $N = 64$, $d = 5$, $x \in [-1, 1]$, and equidistant points.

As Figure 4.1 shows, the numerical solution using the FH interpolants is very good; the maximum absolute error using 64 collocation points is 8.10×10^{-10}.

4.2 Optimal d Value

Using the FH interpolants to solve BVPs, it was investigated what d value would be optimal in the sense of minimizing the maximum absolute error. The previous BVP was used, equation (4.1), with $N = 20$, along with two other simple examples, namely BVP_2

$$u'' = e^{4x}, \quad u(-1) = 0, \quad u(1) = 0$$

whose exact solution is

$$u(x) = \frac{e^{4x} - \sinh(4)x - \cosh(4)}{16}$$

(this example is from [35]) and BVP_3

$$u'' + u' = \frac{-2(x^3 - 3x^2 + x + 1)}{(x^2 + 1)^3}, \quad u(-1) = \frac{1}{2}, \quad u(1) = \frac{1}{2}$$

whose exact solution is

$$u(x) = \frac{1}{1 + x^2}.$$

Figure 4.2 is a plot of d values versus maximum absolute error, allowing us to determine the optimal d value (the minimum error).

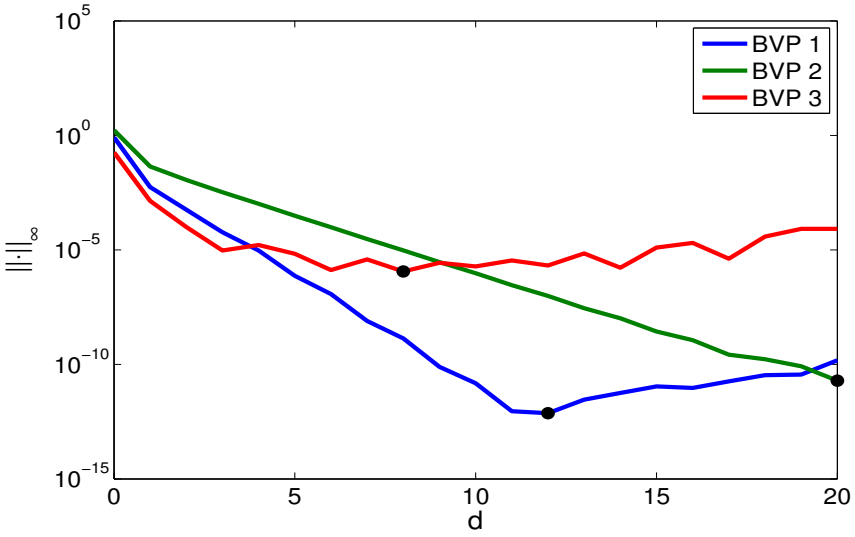

Figure 4.2: Maximum absolute differences between the exact and numerical solutions, with $N = 20$ and equidistant points, for $0 \leq d \leq N$. The black dot marks the minimum error.

One can see that the optimal d value is not the same for all examples. There is some discussion of optimal d values in the literature in relation to minimizing interpolation error, such as in [20] and [10]. However, these results do not necessarily hold true for the present work in solving BVPs and there are additional restrictions (which will be discussed) in choosing d values in this case.

Consideration is also given to whether using a different distribution of points would alter the results. Figure 4.3 shows the results:

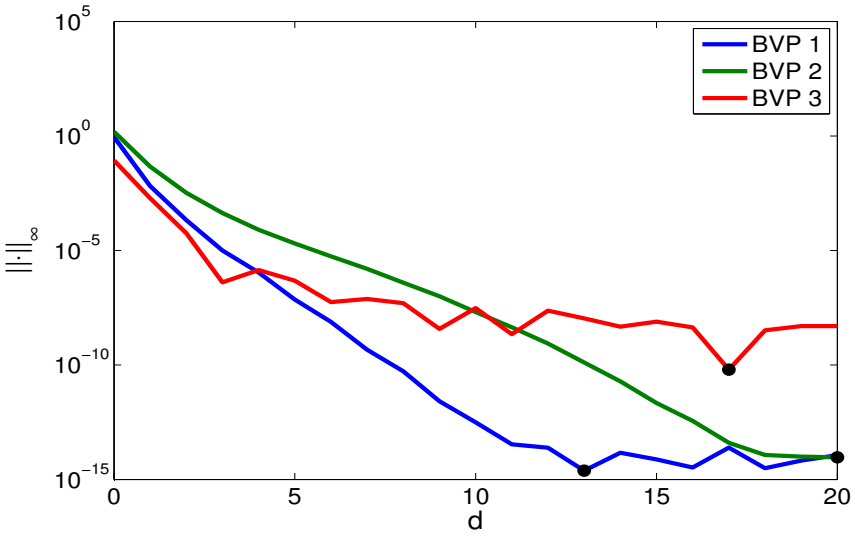

Figure 4.3: Maximum absolute differences between the exact and numerical solutions, with $N = 20$ and Chebyshev points, for $0 \leq d \leq N$. The black dot marks the minimum error.

Again there is no clear choice, though larger values of d seem to be better. However, as will be elaborated later, there are numerical issues with using larger d values when N becomes large, primarily due to the calculation of the weights, w_j. This can be seen by choosing $N = 128$. Figures 4.4 and 4.5 show the new optimal d values.

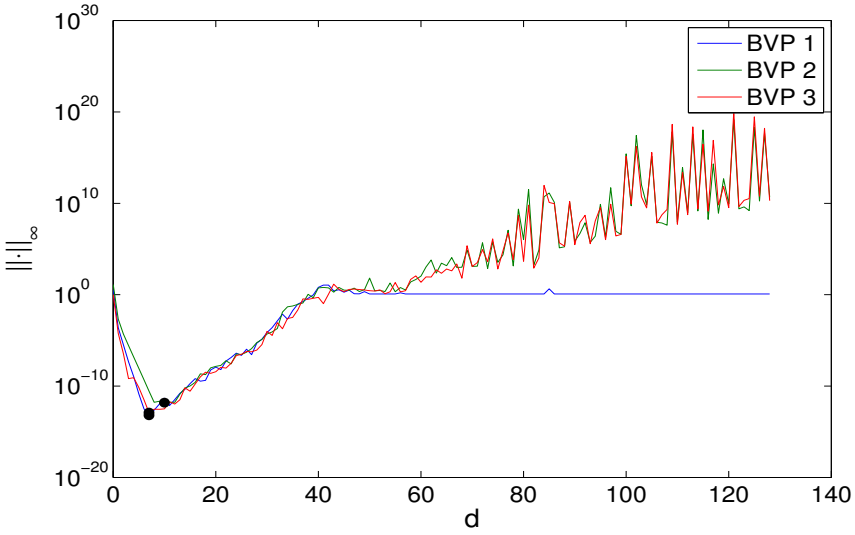

Figure 4.4: Maximum absolute differences between the exact and numerical solutions, with $N = 128$ and equidistant points, for $0 \leq d \leq N$. The black dot marks the minimum error.

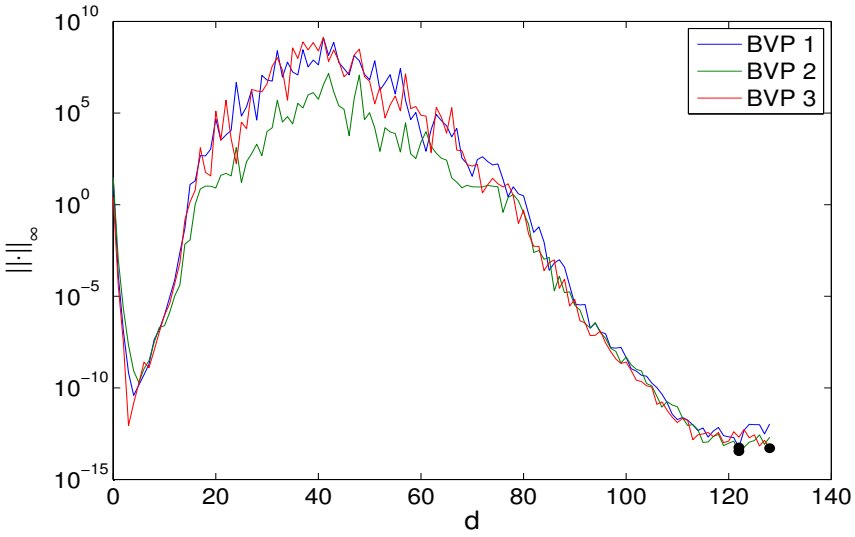

Figure 4.5: Maximum absolute differences between the exact and numerical solutions, with $N = 128$ and Chebyshev points, for $0 \leq d \leq N$. The black dot marks the minimum error.

As can be seen from Figures 4.4 and 4.5, numerical instability begins to play an important role, even for a relatively small value of N (compared to say $N = 1024$). This is largely due to numerical issues in the differentiation matrices and the FH weights. Figures 4.6 and 4.7 illustrate this by showing the largest weights (in absolute value) for each value of d, as well as the condition numbers of the first and second derivative matrices.

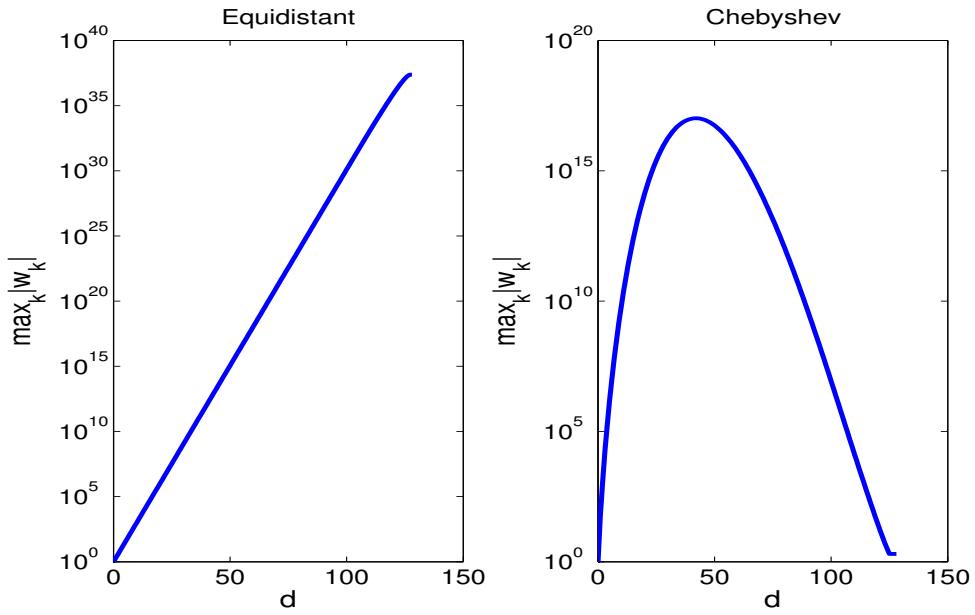

Figure 4.6: Maximum of the absolute value of scaled weights ($w_j / \min_i(w_i)$) for each value of d, with $N = 128$, for both equidistant and Chebyshev points.

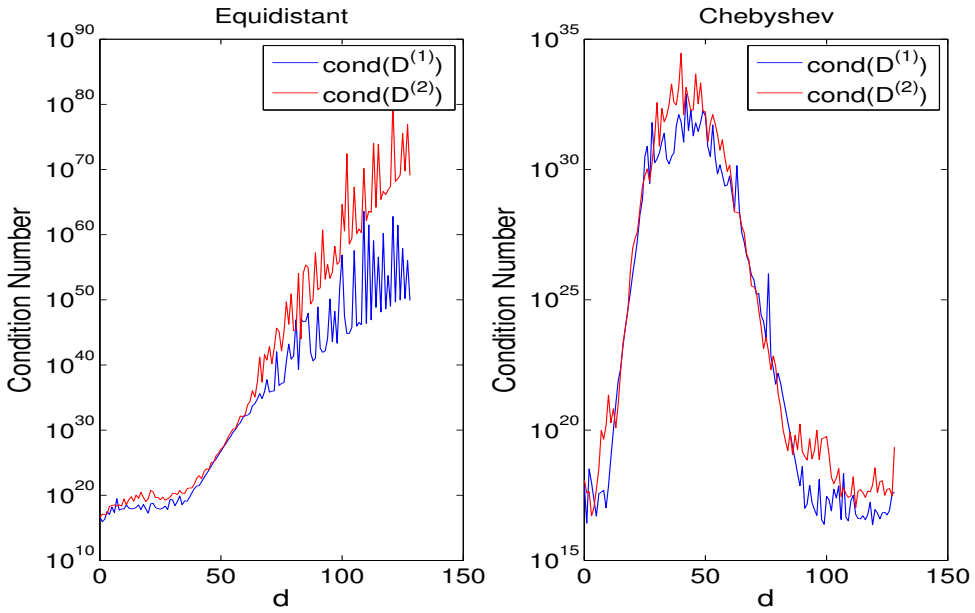

Figure 4.7: Condition numbers of the first and second derivative matrices for each value of d, with $N = 128$, for both equidistant and Chebyshev points.

Especially for the equidistant points, it is now obvious that a smaller value of d is optimal. This is due to the large value of the weights when calculating the differentiation matrices, as these matrices become very ill-conditioned. In the case of Chebyshev points, a very small or very large value of d seems to be acceptable, but with preference to values of d nearer to N. In the following work, two things will be investigated. Firstly, for a small value of d, $0 \leq d \leq 10$, these differentiation matrices are used in solving singularly perturbed BVPs; this will be discussed more. Although not optimal for the Chebyshev points, this is a near optimal value for the equidistant points. Secondly, in the case of Chebyshev points, it is investigated how the use of the FH interpolants with $d \approx N$ compares to the standard Chebyshev differentiation techniques using polynomial interpolation.

4.3 Boundary Layers

A further interesting application of the Floater-Hormann interpolants is to see their use in solving BVPs with boundary layers. For the purposes of this thesis, we will restrict our use of the term *boundary layer* to second order BVPs with a singular perturbation of the second derivative term, or equations of the form

$$\varepsilon u''(x) + p(x)u'(x) + q(x)u(x) = f(x).$$

BVPs of this form can have both interior and boundary layers. Unless otherwise stated, we will be considering only boundary layers, i.e. layers which occur at the endpoints of the interval in question.

4.3.1 FH Interpolants Approach

Firstly, looking simply at the techniques we have already developed, one can see that even with the Chebyshev points, certain singularly perturbed BVPs cannot be solved numerically (in the sense that the boundary layer cannot be resolved). This can be seen in Figure 4.8 below using the first example from [34], which is

$$\varepsilon u'' - xu' - u = \left(\frac{x+1}{\varepsilon} - 1\right) e^{(x+1)/\varepsilon} - 2\left(\frac{x-1}{\varepsilon} + 1\right) e^{(x-1)/\varepsilon}, \quad u(-1) = 1, \;\; u(1) = 2 \tag{4.2}$$

having the exact solution

$$u(x) = e^{-((x+1)/\varepsilon)} + 2e^{((x-1)/\varepsilon)}.$$

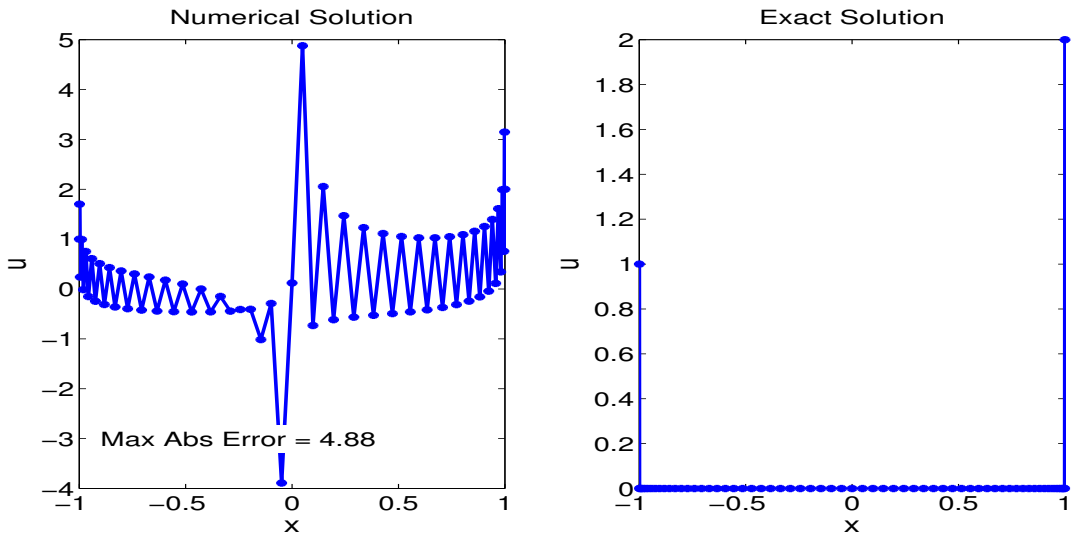

Figure 4.8: Solution to BVP in equation 4.2 with $N = 64$ and $\varepsilon = 10^{-4}$ using Chebyshev points and $d = 0$.

Here, Chebyshev points are used (which cluster around the endpoints and thus are better suited to solving BVPs with boundary layers) with FH interpolation. It was found that $d = 0$ gave the minimum max-norm error, and this best-case result is the one shown. With larger values of ε ($\varepsilon \approx 10^{-2}$), adequate results can be obtained simply using Chebyshev points; however, for smaller values of ε an alternative approach is required.

4.3.2 Combined FH Interpolants and BLRSC Approach

One solution to the inability of the FH interpolant approach to solving singularly perturbed BVPs lies in a paper by Tang and Trummer, [34]. In this paper, a method is developed which allows one to use any point distribution to resolve even a very small boundary layer. This method will be referred to as the *Boundary Layer Resolving Spectral Collocation*, or BLRSC, method. The basic idea is to use a coordinate transformation to cluster the points at the boundaries. This allows one to resolve boundary layers for very small value of ε. As stated in [34], good numerical results in solving the singularly perturbed BVP require that at least one collocation point lies in the boundary layer. The BLRSC method ensures that this will happen. First, to describe the method we start with the general second order BVP

$$\varepsilon u''(x) + p(x)u'(x) + q(x)u(x) = f(x), \quad x \in (-1, 1), \quad u(-1) = \alpha, \;\; u(1) = \beta \tag{4.3}$$

with $0 < \varepsilon \ll 1$, we then iteratively apply sine functions. These sine transformations, $x = g_m(y)$, $m = 0, 1, \ldots$ are defined as

$$g_0(y) := y, \qquad g_m(y) = \sin\left(\frac{\pi}{2} g_{m-1}(y)\right), \ \ m \geq 1. \tag{4.4}$$

With this transformation (and using the Chebyshev collocation points), [34] gives the spacing between the boundary point and first interior point as $O(N^{-4}), O(N^{-8})$, and $O(N^{-16})$ for one, two, and three sine transformations respectively. Thus, even for relatively small N, we can be sure that at least one point lies in any given boundary layer with sufficient sine transformations. Using equidistant points, the spacing between the boundary point and first interior point is $O(N^{-2m})$, which is not as close as the Chebyshev points. This implies that, typically, more transformations are needed for equidistant points given for given N and ε; this is also seen in results. However, as is also evident, for $m \geq 4$ using equidistant points, even relatively small N would ensure at least one collocation point lies in the boundary layer. For example, with $N = 32$ and $m = 4$, this method could resolve BVPs with boundary layers of width $\approx 10^{-12}$.

The transformed equation (via the variable transformation x=x(y)) is

$$\varepsilon v''(y) + \tilde{p}(y)v'(y) + \tilde{q}(y)v(y) = \tilde{f}(y)$$

where v is the transplant of u, $v(y) = u(x(y))$. The coefficients $\tilde{p}, \tilde{q}, \tilde{f}$ are given by

$$\tilde{p}(y) := \frac{p(x)}{y'(x)} + \varepsilon \frac{y''(x)}{y'(x)^2}, \qquad \tilde{q}(y) := \frac{q(x)}{y'(x)^2}, \qquad \tilde{f}(y) := \frac{f(x)}{y'(x)^2}.$$

In [34], computational forms for $\dfrac{1}{y'(x)^2}$ and $\dfrac{y''(x)}{y'(x)}$ are given.

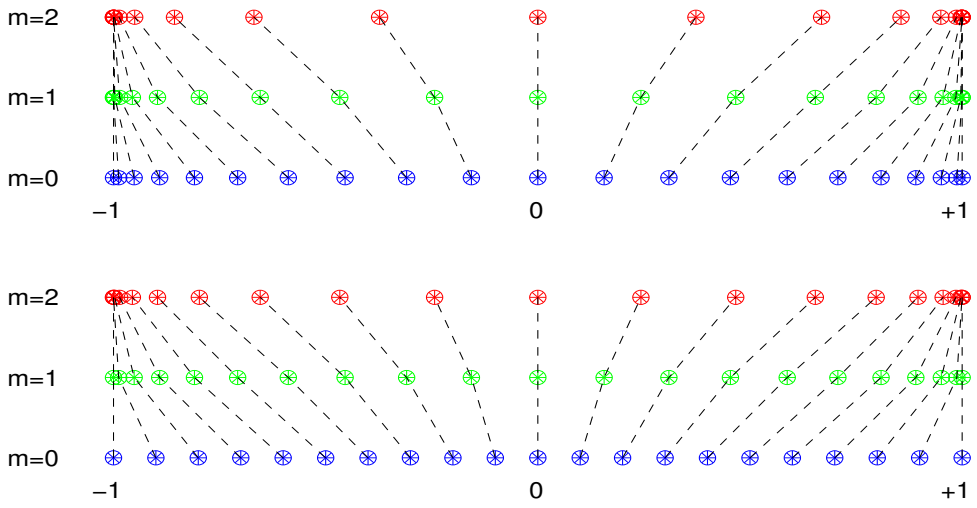

Figure 4.9: A useful representation of the effect of the transformation on Chebyshev and equidistant points, respectively.

4.3.3 Results of Combined FH Interpolants and BLRSC Approach

Returning to the previous example, Equation 4.2, we now use the BLRSC method. As already mentioned, we will also use the standard Chebyshev differentiation matrices, based on polynomial interpolation, as in [34]. This allows a comparison to the FH interpolants. Below are the solution profiles on the physical and computational domains when using the FH interpolants with $d = 4$ and Chebyshev points, as well as an error plot of three methods (FH interpolants using Chebyshev and equidistant points, and polynomial interpolation using Chebyshev points).

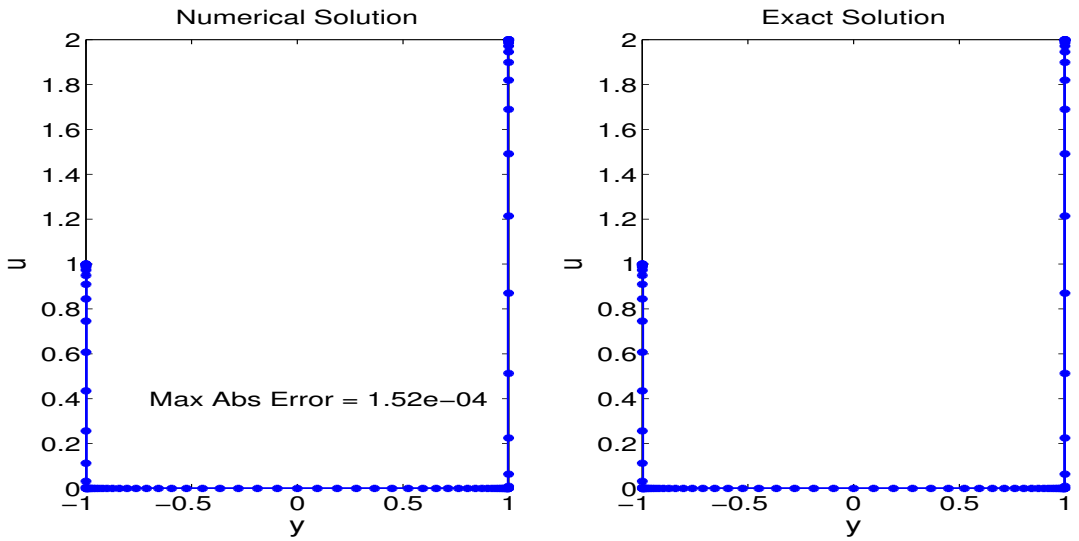

Figure 4.10: Solution to Equation (4.2) with $\varepsilon = 10^{-4}$ and three sine transformations in the physical domain using FH interpolants and Chebyshev points, with $d = 4$ and $N = 128$.

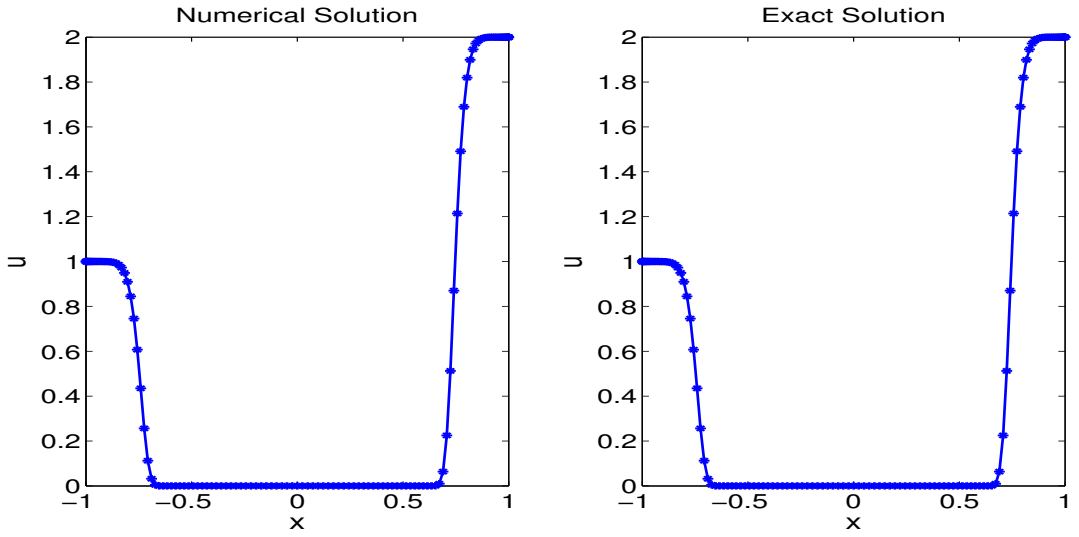

Figure 4.11: Solution to Equation (4.2) with $\varepsilon = 10^{-4}$ and three sine transformations in the computational domain using FH interpolants and Chebyshev points, with $d = 4$ and $N = 128$.

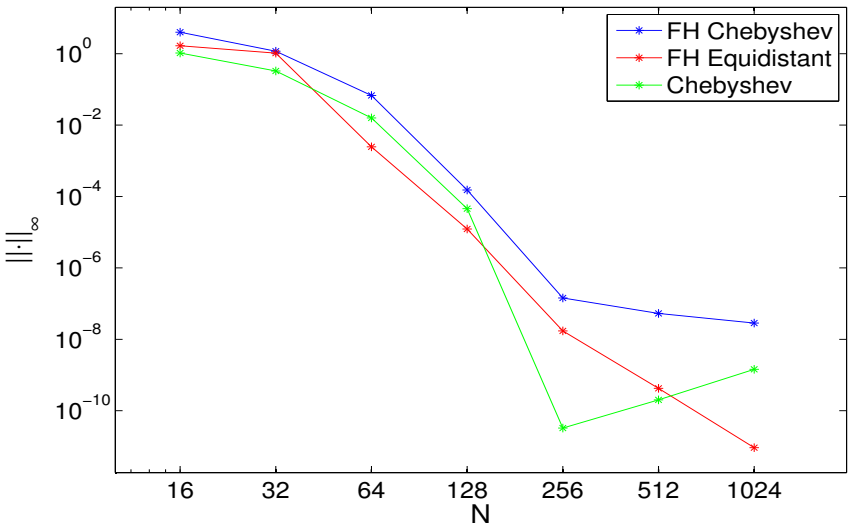

Figure 4.12: Maximum absolute error in solving Equation (4.2) with $\varepsilon = 10^{-4}$, three sine transformations, and $d = 4$ for the FH interpolants methods.

There are a few important things to note. Firstly, combining the FH interpolants and corresponding differentiation matrices allows one to successfully compute the numerical solution of the singularly perturbed BVP. Now, the solution closely approximates the exact solution and spectral accuracy can be seen in Figure 4.12. Also, in terms of the maximum absolute error, the different methods are fairly similar for small N, but quite different for larger N. The parameters here are chosen simply to illustrate the success in combining the FH interpolants with the BLRSC method.

Further testing was done to find optimal parameters for this particular BVP for each method. Four cases were looked at. First, equidistant points with FH interpolants and $d \in [0, 10]$. Second, Chebyshev points with FH interpolants and $d \in [0, 10]$. Third, Chebyshev points with FH interpolants, and $d \in [N - 9, N]$. Lastly, Chebyshev points with polynomial interpolants. Figure 4.13 shows the best case errors (minimum of the max-norm errors) and Table 4.1 shows which values of d and m produce these best case errors.

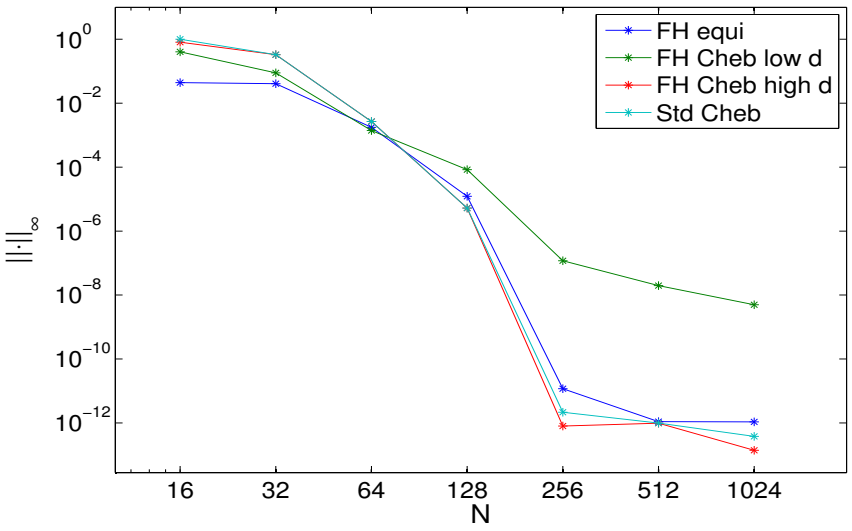

Figure 4.13: Minimum max-norm error in solving Equation (4.2) with $\varepsilon = 10^{-4}$. The appropriate parameters for each of the four methods are shown in the following table.

Table 4.1: Error in Solving BVP

N	FH Equi.		FH Cheb.		FH Cheb.*		Poly. Cheb.
	m	d	m	d	m	d	m
16	0	10	3	0	2	16	2
32	1	10	3	0	3	31	3
64	3	2	2	4	2	64	2
128	3	4	3	3	2	127	2
256	3	10	3	5	1	248	1
512	5	7	5	3	1	508	1
1024	6	5	3	3	0	1023	0

Optimal m and d values to obtain minimum error (absolute maximum differences). The third set of columns, titled FH Cheb*, is testing values of d near N as opposed to smaller values of d.

Figure 4.13 and Table 4.1 show a few important things. Firstly, using Chebyshev points with small values of d is far from optimal for larger values of N. The other two methods using Chebyshev points (using FH interpolants with values of d near N, and using polynomial interpolation) are very

similar. Although the FH method gives slightly better error for larger values of N, these results suggest that there is no major advantage between these methods in terms of max-norm error. However, since the polynomial interpolation method is more easily implemented and runs slightly quicker (both because the weights do not have to be calculated and because the implementation of differentiation matrices can take advantage of the known symmetry), the polynomial interpolation method is preferable.

Looking at the FH method with equidistant points, there is a slight advantage to using the FH method over the polynomial interpolation with Chebyshev points for small N. For $N = 16$ and $N = 32$ the error using the FH method with equidistant points was about one to two orders of magnitude smaller than the other methods. It was tested to see if the error would continue to decrease further if more values of d were tested. For $N = 16$ and $N = 32$, it decreased further, but only slightly (the error was still $O\left(10^{-2}\right)$). However, the error did not decrease further for $N = 64$, meaning that numerical issues began to dominate (the size of the weights and the condition number of the differentiation matrices). For larger values of N, this method was about the same as the standard Chebyshev method.

The other interesting thing to note is the optimal number of sine iterations. For the two viable methods using Chebyshev points (the FH interpolants with high d values and the method using polynomial interpolation) the number of sine iterations decreased as N increased. This was probably due to error in the interior. As can be seen in Figure 4.9, the effect of the sine transformations on the Chebyshev points causes a large gap between points in the interior. This was seen to be true when plotting the error for a given N against the domain to see where it occurred. In contrast, the optimal number of sine transformations decreased when using equidistant points. For large enough values of ε, this phenomenon seemed to allow equidistant points to outperform Chebyshev points (in terms of reducing maximum absolute error).

Chapter 5

Coupled System of Singularly Perturbed Linear BVPs

5.1 Background

Singularly perturbed boundary value problems have been extensively studied, whereas systems of coupled singularly perturbed boundary value problems have received less attention. Real world applications of these problems include the modelling of electrochemical reactions, various problems within optimal control theory, certain resistance-capacitor electrical circuits, and certain predator-prey models (see [22], [30], [28]).

In much of the mathematical literature dedicated to numerical methods for solving second order singularly perturbed linear BVPs problems for very small ε, the approach has been to use layer-adapted meshes, usually Shishkin or Bakhvaov meshes, and then to employ a finite difference or finite element scheme. This layer-adapted mesh divides the domain into two (or more) subdomains, one coarser and the other(s) finer, the finer mesh(es) capturing the boundary layer. Other methods include spectral collocation methods, a category to which our method belongs. There are other methods which are more general, but which can be applied to these specific problems. For a comprehensive review of singularly perturbed BVPs and a variety of numerical methods for solving them, see [2]. This thesis will focus on methods specifically designed for solving second order singularly perturbed linear BVPs problems with very small ε.

Here, we provide a ”new“ spectral collocation method for solving coupled systems of second order linear boundary value problems. We use the iterated sine transformations in conjunction with a spectral collocation method. Although this method was implemented in [16], their implementation was suboptimal, as discussed later. As seen in the results of this section, this method is spectrally accurate and can resolve systems with extremely small boundary layers. Also, this method is

very flexible, allowing one to solve a system of second order singularly perturbed BVPs with any type of strong and/or weak coupling. Often, methods in the literature investigated focus only on one form of a second order singularly perturbed system of boundary values problems, such as a reaction-diffusion equation. Additionally, as discussed later, this new method also allows for different perturbation parameters in each system of equations, whereas much of the literature reviewed required problems with equal perturbation parameters.

There are a few caveats. Firstly, the layers in these problems are assumed to be boundary layers, occurring at either or both ends of the domain; this is also true in the literature reviewed. Secondly, results in this thesis only include systems of two coupled boundary value problems; the method has been extended to systems of three coupled BVPs, and further extension to even larger systems seems straightforward, but these results are not included here.

As of yet, these results have only been computed using the standard Chebyshev method (using polynomial interpolation). A few tests have been conducted to see the benefit of using FH interpolation and equidistant points. For very small values of ε, the standard Chebyshev method seems to be better.

5.2 Method

The BLSRM for a single BVP was described in the previous chapter. Now, we consider a system of coupled boundary value problems

$$\begin{cases} \varepsilon_1 u_1''(x) + p_{11}(x)u_1'(x) + p_{12}(x)u_2'(x) + q_{11}(x)u_1(x) + q_{12}(x)u_2(x) &= f_1(x) \\ \\ \varepsilon_2 u_2''(x) + p_{21}(x)u_1'(x) + p_{22}(x)u_2'(x) + q_{21}(x)u_1(x) + q_{22}(x)u_2(x) &= f_2(x). \end{cases}$$

Making the transformation $x = x(y)$, we find that our equations become

$$\begin{cases} \varepsilon_1 v_1''(y)y'(x)^2 + \varepsilon_1 v_1'(y)y''(x) + p_{11}(x)v_1'(y)y'(x) + p_{12}(x)v_2'(y)y'(x) \\ \qquad\qquad +q_{11}(x)v_1(y) + q_{12}(x)v_2(y) &= f_1(x) \\ \\ \varepsilon_2 v_2''(y)y'(x)^2 + \varepsilon_2 v_2'(y)y''(x) + p_{21}(x)v_1'(x)y'(y) + p_{22}(x)u_2'(y)y'(x) \\ \qquad\qquad +q_{21}(x)v_1(y) + q_{22}(x)v_2(y) &= f_2(x) \end{cases}$$

where v_1 is the transplant of u_1, $v_1(y) = u_1(x(y))$ and v_2 is the transplant of u_2, $v_2(y) = u_2(x(y))$. After some rearranging, we have

$$\begin{cases} \varepsilon_1 v_1''(y) + \tilde{p}_{11}(y)v_1'(y) + \tilde{p}_{12}(y)v_2'(y) + \tilde{q}_{11}(y)v_1(y) + \tilde{q}_{12}(y)v_2(y) &= \tilde{f}_1(y) \\ \\ \varepsilon_2 v_2''(y) + \tilde{p}_{21}(y)v_1'(y) + \tilde{p}_{22}(y)v_2'(y) + \tilde{q}_{21}(y)v_1(y) + \tilde{q}_{22}(y)v_2(y) &= \tilde{f}_2(y) \end{cases}$$

where

$$\tilde{p}_{11}(y) = \varepsilon_1 \frac{y''(x)}{y'(x)^2} + \frac{p_{11}(x)}{y'(x)}, \qquad \tilde{p}_{12}(y) = \frac{p_{12}(x)}{y'(x)},$$
$$\tilde{p}_{21}(y) = \frac{p_{21}(x)}{y'(x)}, \qquad \tilde{p}_{22}(y) = \varepsilon_2 \frac{y''(x)}{y'(x)^2} + \frac{p_{22}(x)}{y'(x)},$$

$$\tilde{q}_{11}(y) = \frac{q_{11}(x)}{y'(x)^2}, \qquad \tilde{q}_{12}(y) = \frac{q_{12}(x)}{y'(x)^2},$$
$$\tilde{q}_{21}(y) = \frac{q_{21}(x)}{y'(x)^2}, \qquad \tilde{q}_{22}(y) = \frac{q_{22}(x)}{y'(x)^2},$$

$$\tilde{f}_1(y) = \frac{f_1(x)}{y'(x)^2} \qquad \tilde{f}_2(y) = \frac{f_2(x)}{y'(x)^2}.$$

We now compare the results of our method against some of the results found in the literature.

5.3 Numerical Results

All systems (two coupled linear second order BVPs) will be of the form

$$E\vec{u}'' + P\vec{u}' + Q\vec{u} = \vec{f}$$

where E, P, and Q are 2×2 matrices of the form

$$E = \begin{pmatrix} \epsilon_1 & 0 \\ \\ 0 & \epsilon_2 \end{pmatrix}, \quad P = \begin{pmatrix} p_{11}(x) & p_{12}(x) \\ \\ p_{21}(x) & p_{22}(x) \end{pmatrix}, \quad Q = \begin{pmatrix} q_{11}(x) & q_{12}(x) \\ \\ q_{21}(x) & q_{22}(x) \end{pmatrix}$$

with

$$\vec{u}'' = \begin{pmatrix} u_1''(x) \\ \\ u_2''(x) \end{pmatrix}, \quad \vec{u}' = \begin{pmatrix} u_1'(x) \\ \\ u_2'(x) \end{pmatrix}.$$

5.3.1 Comparison with Z. Cen

The system used here is

$$-E\vec{u}'' + P\vec{u}' + Q\vec{u} = \vec{f}$$

with

$$P = \begin{pmatrix} -1 & 0 \\ 0 & -2 \end{pmatrix}, \quad Q = \begin{pmatrix} 2 & -1 \\ -1 & 4 \end{pmatrix}$$

and $\vec{f}$ given as the result of the exact solution being

$$u_1(x) = \frac{1 - e^{-x/\epsilon_1}}{1 - e^{-1/\epsilon_1}} + \frac{1 - e^{-x/\epsilon_2}}{1 - e^{-1/\epsilon_2}} - 2\sin\left(\frac{\pi x}{2}\right)$$
$$u_2(x) = \frac{1 - e^{-x/\epsilon_2}}{1 - e^{-1/\epsilon_2}} - xe^{x-1}$$

along with boundary conditions

$$u_1(0) = u_1(1) = u_2(0) = u_2(1) = 0.$$

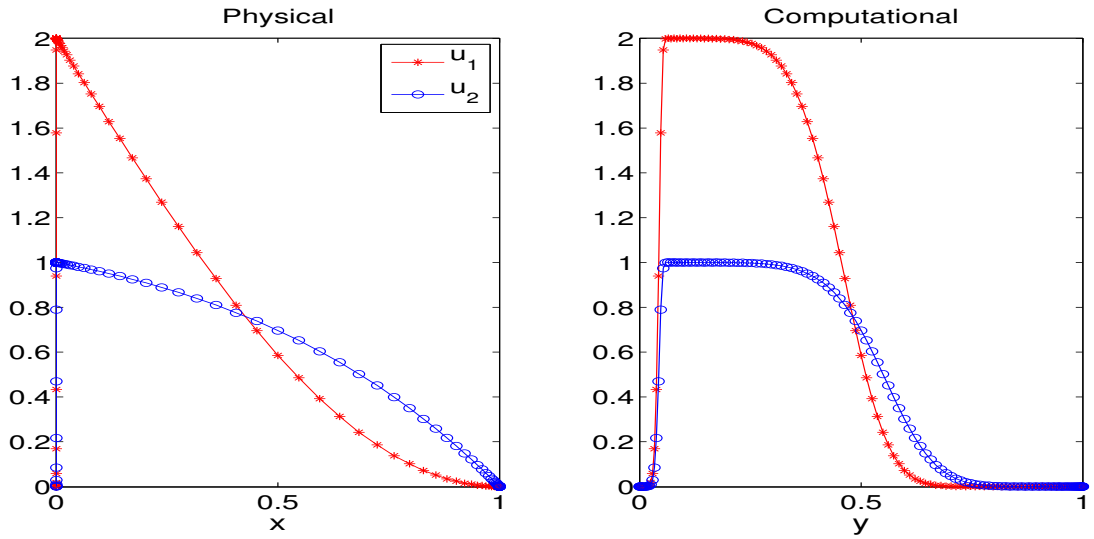

Figure 5.1: Numerical solution profile on physical and computational domain, with $\varepsilon_1 = \varepsilon_2 = 10^{-8}$, three sine iterations, and $N = 128$.

Table 5.1: Error Using BLRSC Method, Example from Z. Cen

	$\epsilon_1 = 10^{-8}$		
N	$\epsilon_2 = 10^{-1}$	$\epsilon_2 = 10^{-4}$	$\epsilon_2 = 10^{-8}$
16	1.17	8.78e-01	2.54
32	6.31e-01	6.63e-01	1.31
64	2.59e-02	2.67e-02	5.19e-02
128	9.20e-04	9.21e-04	1.85e-03
256	1.20e-06	1.20e-06	2.40e-06
512	6.64e-09	2.96e-09	5.47e-09
1024	2.71e-09	2.71e-09	5.41e-09

BLRSC method results in solving example from Z. Cen [15]. Three sine transformations are used for each trial. Errors shown are the maximum of u_1 and u_2 errors.

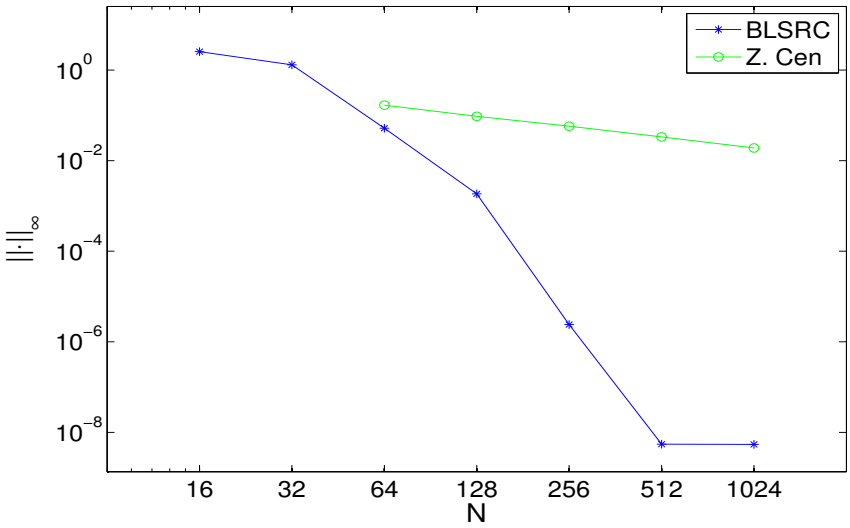

Figure 5.2: Log-log plot of maximum absolute error, with $\varepsilon_1 = \varepsilon_2 = 10^{-8}$ and 3 sine tranformations.

In comparison to the last row of Table 5.1, results obtained by [15] were (for $N = 1024$ and $\epsilon_1 = 10^{-8}$), $1.230e-02$, $2.999e-02$, and $1.899e-02$ for $\epsilon_2 = 10^{-1}, 10^{-4}$, and 10^{-8} respectively. Referring to [15] for more error results, it can be seen that results obtained in this thesis for $N \geq 128$

are far more accurate than those in [15], even for very small $\varepsilon_1, \varepsilon_2$.

5.3.2 Comparison with Matthews, Miller, O'Riordan, and Shishkin

The system being considered here has

$$P = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad Q = \begin{pmatrix} 3 & -1 \\ -1 & 3 \end{pmatrix}, \quad \vec{f} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

with boundary conditions

$$u_1(0) = u_1(1) = u_2(0) = u_2(1) = 0$$

In this example, no exact solution was given. Instead, [27] compares their computed solutions using $N = 8, 16, 32, \ldots, 1024$ to a finest mesh solution of $N = 4096$. The same was done here. Our results are seen in Figures 5.3 and 5.4 and Table 5.2 below:

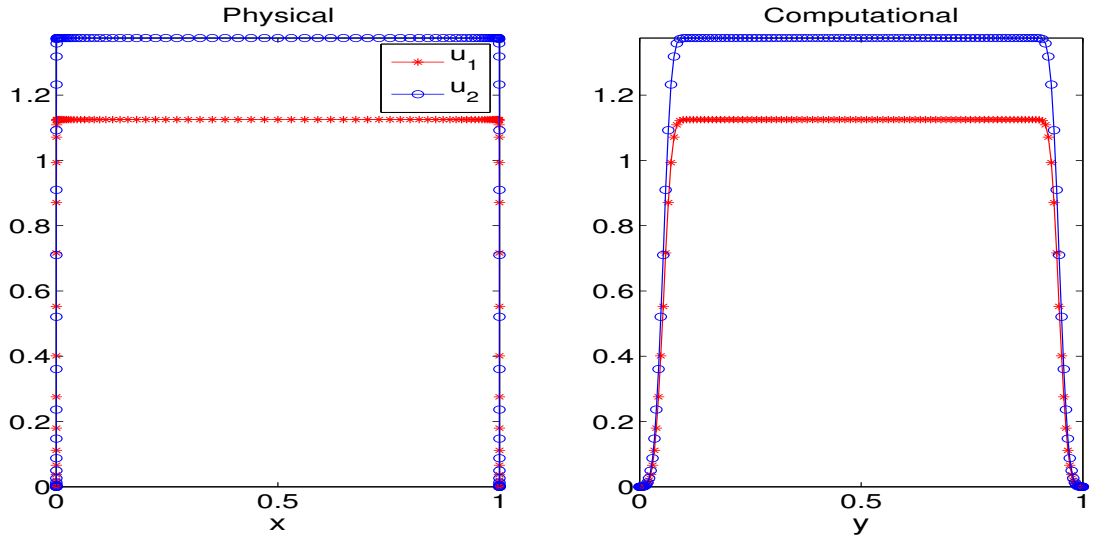

Figure 5.3: Solution profile on computational and exact domains, with $\varepsilon_1 = \varepsilon_2 \approx 2^{-24}$ and two sine iterations.

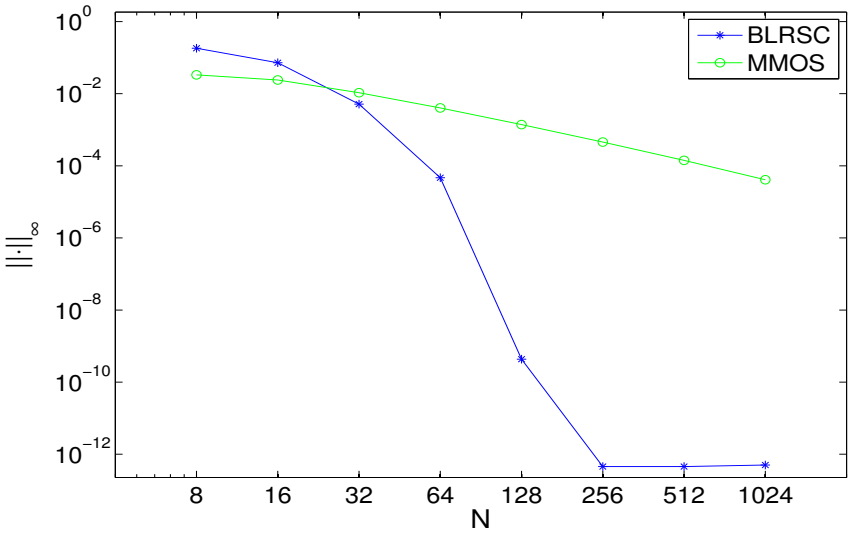

Figure 5.4: Log-log plot of maximum absolute error, with $\varepsilon_1 = \varepsilon_2 = 2^{-24}$ and two sine iterations.

Table 5.2: Error using BLRSC Method, Example from Matthews *et al.*

ε	$N=8$	$N=16$	$N=32$	$N=64$	$N=128$	$N=256$	$N=512$	$N=1024$
2^{0}	5.137e-11	1.464e-12	1.478e-12	1.521e-12	1.524e-12	1.544e-12	1.403e-12	1.178e-12
2^{-2}	1.237e-08	7.091e-12	7.094e-12	7.142e-12	7.176e-12	7.216e-12	7.233e-12	6.974e-12
2^{-4}	3.048e-07	1.118e-11	1.097e-11	1.098e-11	1.096e-11	1.090e-11	1.102e-11	1.129e-11
2^{-6}	1.460e-04	2.041e-09	6.704e-12	6.704e-12	6.709e-12	6.672e-12	6.652e-12	6.762e-12
2^{-8}	1.512e-02	5.253e-05	2.462e-10	1.119e-12	1.159e-12	1.145e-12	1.118e-12	1.072e-12
2^{-10}	3.215e-02	2.790e-04	1.721e-08	1.273e-12	1.268e-12	1.281e-12	1.264e-12	1.334e-12
2^{-12}	1.392e-02	1.792e-03	1.010e-06	1.182e-12	1.299e-12	1.293e-12	1.294e-12	1.310e-12
2^{-14}	1.161e-01	4.289e-03	1.709e-05	9.649e-12	5.019e-13	5.097e-13	5.163e-13	5.499e-13
2^{-16}	1.384e-01	1.756e-02	1.241e-04	1.673e-09	2.955e-13	2.938e-13	3.022e-13	3.157e-13
2^{-18}	3.286e-01	8.764e-03	1.450e-03	8.323e-07	1.131e-12	1.099e-12	1.130e-12	1.111e-12
2^{-20}	3.409e-01	6.085e-02	1.390e-03	2.708e-06	3.732e-12	1.134e-12	1.143e-12	1.044e-12
2^{-22}	3.195e-01	7.860e-02	5.432e-03	1.382e-05	9.590e-11	8.829e-13	8.847e-13	9.378e-13
2^{-24}	1.826e-01	7.231e-02	5.151e-03	4.661e-05	4.313e-10	4.550e-13	4.545e-13	5.040e-13
2^{-26}	1.208e-01	3.013e-02	8.954e-03	1.114e-04	5.981e-09	5.888e-13	6.379e-13	6.630e-13
2^{-28}	3.026e-01	1.059e-01	5.923e-03	2.648e-04	3.381e-08	2.260e-13	2.688e-13	2.288e-13

BLRSC method results in solving example from Matthews, Miller, O'Riordan, and Shishkin. Number of sine transformations used were 0 for $\varepsilon = 2^{0}, 2^{-2}, 2^{-4}$, 1 for $\varepsilon = 2^{-6}, \ldots, 2^{-16}$, and 2 for $\varepsilon = 2^{-18}, \ldots, 2^{-28}$. For this example, $\varepsilon_1 = \varepsilon_2$. The errors shown are in u_1, as was done in [27].

For $N \geq 32$, results are as accurate or more accurate. For $N \geq 128$ and for all values of ε, results are at least six orders of magnitude more accurate. Table 5.2 was reproduced for comparison purposes, as this table appears in [27].

5.3.3 Example Illustrating Non-Constant Advection Coupling

The system being considered here has

$$P = \begin{pmatrix} 2 + \left(\frac{x+1}{4}\right)e^{(x+1)/2} & -1 - \frac{x}{2} \\ -\left(1 + \frac{x+1}{2}\right)/2 & 1 + \frac{(x+1)^2}{8} \end{pmatrix}, \quad Q = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

and $\vec{f}$ given as the result of the exact solution being

$$u_1(x) = \frac{3}{4}e^{-(x+1)/\varepsilon_1} + \frac{x^2 + 6x + 25}{16}$$
$$u_2(x) = \frac{-5}{8}e^{-(x+1)/\varepsilon_2} + \frac{9x^2 + 6x + 49}{32}$$

with boundary conditions

$$u_1(-1) = u_1(1) = u_2(1) = 2, \;\; u_2(-1) = 1.$$

Figures 5.5, 5.6, and 5.7 show results for various values of ε_1, ε_2, and m. In Figure 5.6 we have $\varepsilon_1 = 10^{-2}$ and $\varepsilon_2 = 10^{-4}$ with $m = 1, 2, 3$, and 4. In Figure 5.7 we have $\varepsilon_1 = 10^{-6}$ and $\varepsilon_2 = 10^{-8}$ with $m = 2, 3, 4$, and 5.

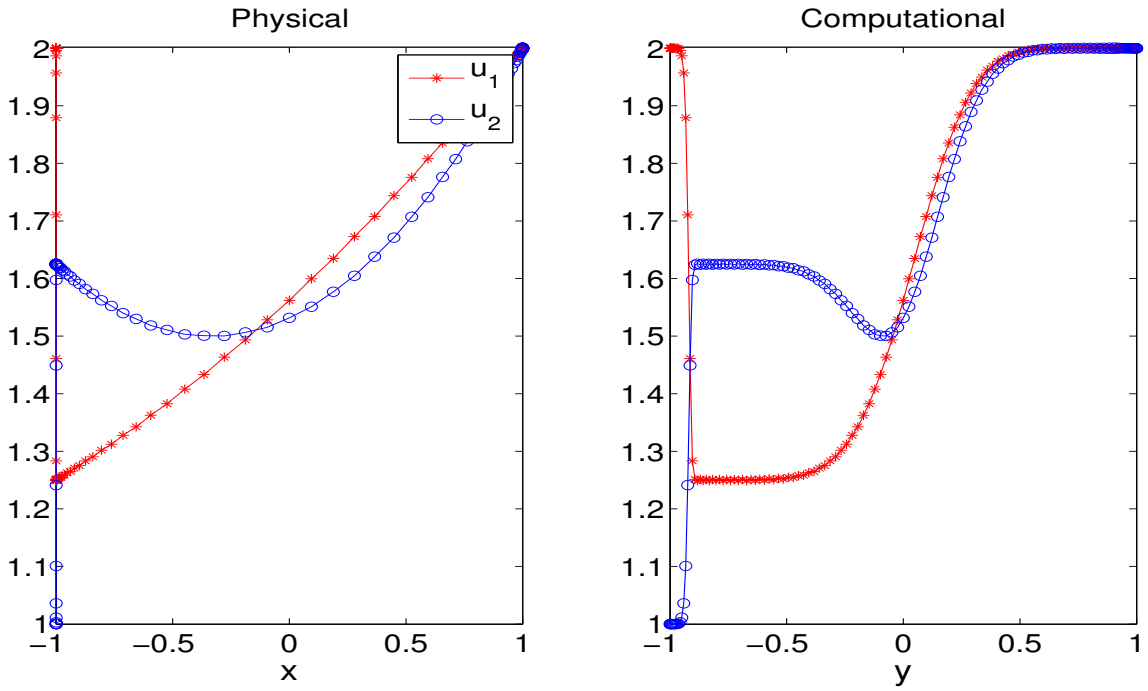

Figure 5.5: Solution, numerical and exact, on computational domain, with $\varepsilon_1 = \varepsilon_2 = 10^{-8}$ and three sine iterations.

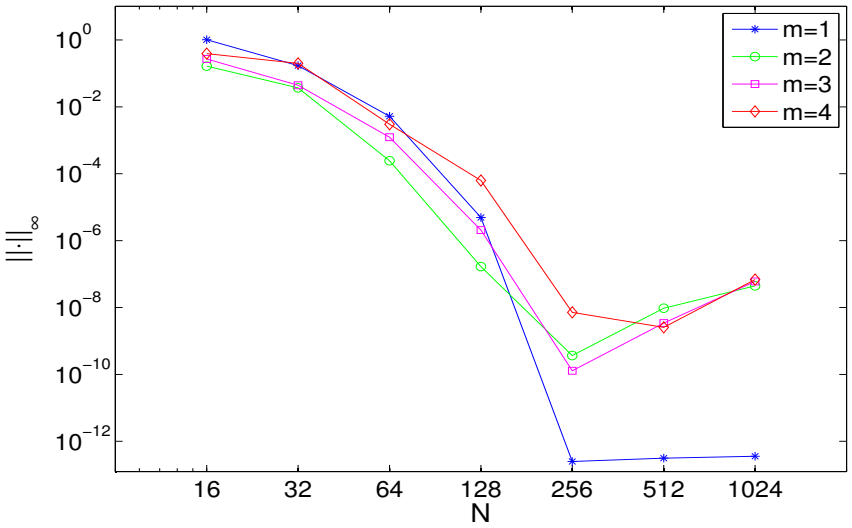

Figure 5.6: Log-log plot of maximum absolute error, with $\varepsilon_1 = 10^{-2}, \varepsilon_2 = 10^{-4}$ and various sine transformations.

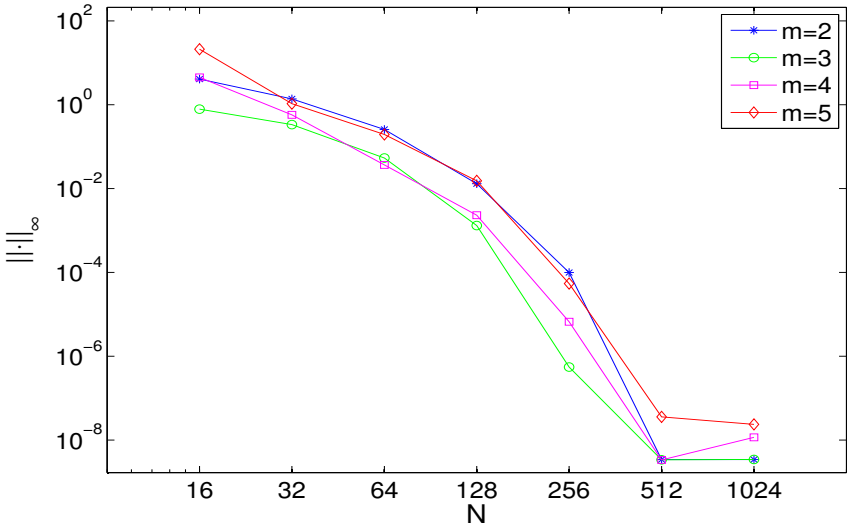

Figure 5.7: Log-log plot of maximum absolute error, with $\varepsilon_1 = 10^{-6}, \varepsilon_2 = 10^{-8}$ and various sine transformations

A number of things are evident from these results. Most obviously, we see spectral accuracy for both sets of perturbation parameters. In Figure 5.6 we see that additional sine iterations (beyond $m = 1$) actually result in higher error. The reason for this, as discussed earlier, is most likely due to larger errors in the interior of the computational domain. For the second case, as seen in Figure 5.7, we see that for most values of N the error is not too sensitive to additional sine transformations. However, if one were to use only one sine transformation in the second case, the error would be much larger for all values of N (not shown). This implies that a general rule of thumb in deciding the number of sine transformations is sufficient in producing results that are reasonably close to optimal results.

5.4 Method/Implementation Comparison to Chen *et al.*

In a recent paper by Chen, Wang, and Wu [16], a spectral collocation method for solving coupled systems of singularly perturbed boundary value problems is developed. This method, the RSC-sinh method, involves mapping the Chebyshev points to a set of transformed points which are determined by the coupling present in the system and the value of ε. The method then employs rational interpolation and the differentiation matrices from rational interpolants in barycentric form (the same as those mentioned earlier in this thesis).

In their results, the authors also implement the BLRSC method, although without much explanation and in a way which is not optimal. After reproducing results from their first numerical example,

the maximum absolute error in the BLRSC method was lower than reported in their paper, especially for smaller values of ε. This was partially due to their use of an inappropriate number of sine transformations for certain perturbation parameters (for example, with $\varepsilon = 10^{-8}$ they only used one sine transformation). Another reason for the smaller error in our examples is perhaps due to better implementation in Matlab. Specifically, this may be the result of how the differentiation matrices are constructed, since the difference was most noticeable for small values of ε and large values of N.

Figures 5.8, 5.9, and 5.10 show the first three cases of example one in [16], where $\varepsilon = 10^{-6}, 10^{-8}$, and 10^{-12}. The error for their RSC-sinh method has been read off the graphs in their paper (as this was the only form it is given in), and thus is only approximate. Even if only approximate, it is useful in seeing the true relative error between methods these two methods.

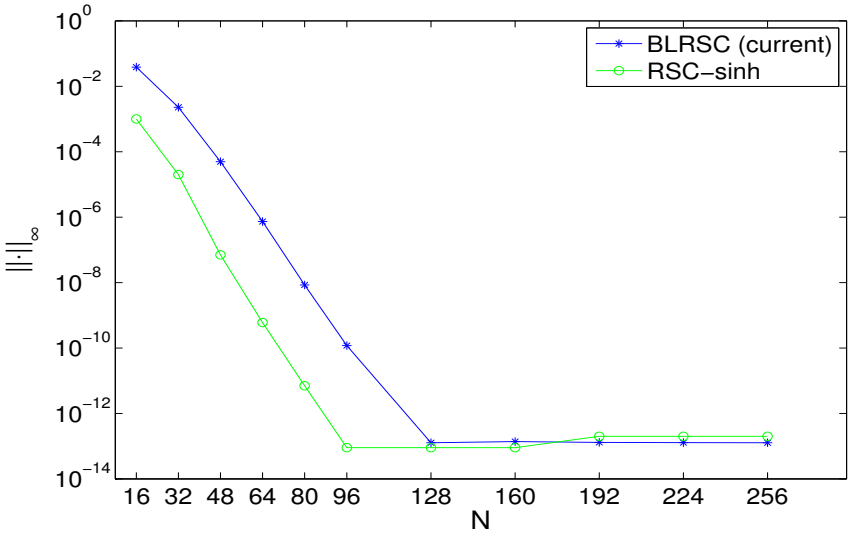

Figure 5.8: Semi-log plot (in the vertical axis) of maximum absolute error, with $\varepsilon_1 = \varepsilon_2 = 10^{-6}$ and one sine transformation.

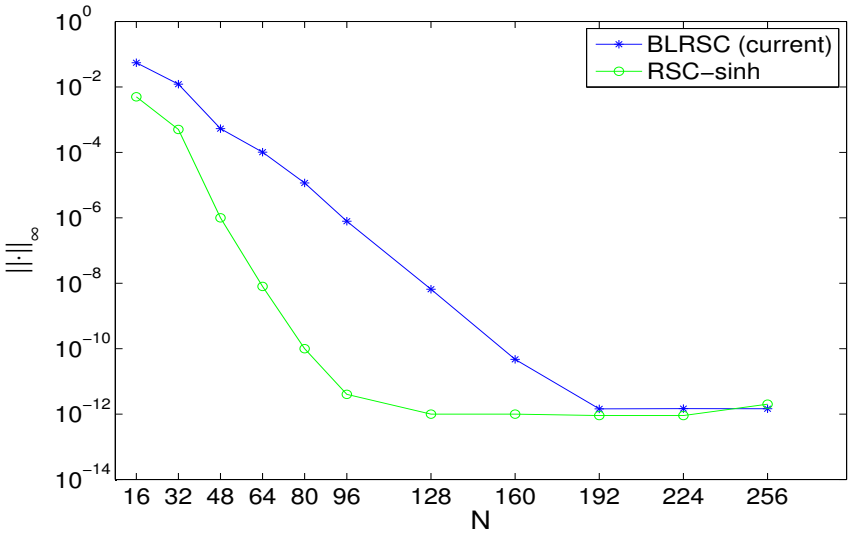

Figure 5.9: Semi-log plot (in the vertical axis) of maximum absolute error, with $\varepsilon_1 = \varepsilon_2 = 10^{-8}$ and two sine transformations.

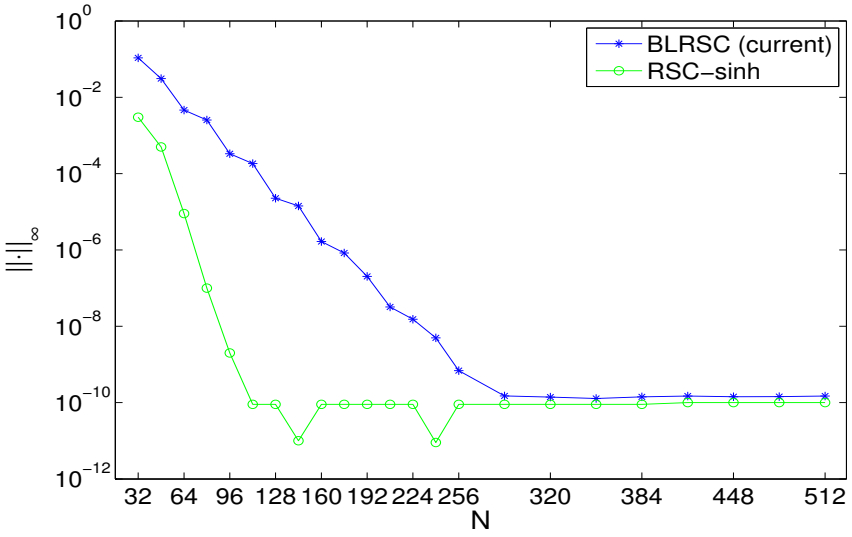

Figure 5.10: Semi-log plot (in the vertical axis) of maximum absolute error, with $\varepsilon_1 = \varepsilon_2 = 10^{-12}$ and two sine transformations.

Additionally, [16] failed to mention the major upsides of the BLRSC method, most notably its ease of implementation and flexibility. For their method, special attention needs to be given to the type of coupling in the system of BVPs. For the BLRSC method, it can handle any type of coupled system of BVPs, with convective and/or reactive coupling without any changes required. Also,

there is no need to know anything about the boundary layer or location. As well, it is extremely computationally efficient. Also, the RSC-sinh method presented in the paper by Chen *et al.* does not allow for different values of ε_j, the perturbation parameter, which (according to certain papers in the literature) seems to be an important feature of these coupled systems and their relation to real world examples. The BLRSC method does allow for different values of ε_j.

Chapter 6

Coupled System of Singularly Perturbed Nonlinear BVPs

In this chapter we briefly investigate nonlinear systems of singularly perturbed BVPs. One difference from the previous chapter is the greater variety of nonlinear second order BVPs. This means that the calculations for the coordinate stretching transformation will need to be done on a case by case scenario. Additionally, these results are simply to prove that the BLRCS method is applicable to nonlinear systems; they do not represent an optimal implementation of the method.

6.1 Method

Here, we simply extend the method described in the previous chapter and apply it to nonlinear systems. The coordinate transformation will largely remain the same, but with the details of the coordinate transformation being dependent upon the specific system being considered. We will employ Newton's method to resolve the nonlinearity. Specifically, once the original problem has been transformed to the appropriate domain of $x \in [-1, 1]$ (if necessary) and the appropriate sine transformations applied to the equation (as described at the beginning of the previous chapter), we will compute the Jacobian. We define our system (which is now in $v(y)$ after the sine transformations) as follows, temporarily writing the two functions v_1 and v_2 as v and w, where $\vec{V} = [v\ w]^T$:

$$\begin{aligned} G &= \varepsilon_1 v'' + f_1\left(v', w', v, w, y, \varepsilon_1, \varepsilon_2\right) \\ H &= \varepsilon_2 w'' + f_2\left(v', w', v, w, y, \varepsilon_1, \varepsilon_2\right). \end{aligned} \tag{6.1}$$

Replacing the derivatives with differentiation matrices we have

$$G(v,w) = \varepsilon_1 D^{(2)} v + \tilde{f}_1\left(Dv, Dw, v, w, y, \varepsilon_1, \varepsilon_2\right)$$
$$H(v,w) = \varepsilon_2 D^{(2)} w + \tilde{f}_2\left(Dv, Dw, v, w, y, \varepsilon_1, \varepsilon_2\right).$$

We then define our Jacobian as

$$J = \begin{pmatrix} \dfrac{\partial G_i}{\partial v_j} & \dfrac{\partial G_i}{\partial w_j} \\ \\ \dfrac{\partial H_i}{\partial v_j} & \dfrac{\partial H_i}{\partial w_j} \end{pmatrix}$$

where $v_j = v(y_j)$ and $w_j = w(y_j)$. We can thus state Newton's method as

$$\vec{V}^{new} = \vec{V}^{old} - J^{-1}[G\ H]^T. \tag{6.2}$$

We define our stopping criterion as

$$\|\vec{V}^{new} - \vec{V}^{old}\|_2 \leq tol$$

where tol is a user defined tolerance.

It was found that this method did not always converge for certain problems. Therefore, we use a slightly modified version of Newton's method as follows

$$\vec{V}^{new} = \vec{V}^{old} - \alpha J^{-1}[G\ H]^T \tag{6.3}$$

where $0 < \alpha \leq 1$. As discussed in [29] and [32], this is simply taking a smaller step length. Although more sophisticated methods exist for calculating an optimal step length, here we use a simple backtracking method when needed (see [29], [32]). Thus, convergence rates are not always optimal (in the sense of speed of convergence).

6.2 Numerical Results

6.2.1 Example 1, Modified Burgers' Equation

We slightly modify the Burgers' equation example from [34] to obtain

$$\begin{cases} \varepsilon_1 u_1''(x) + u_1(x)u_1'(x) + u_2(x) - \tanh\left(\dfrac{x}{2\varepsilon_2}\right) = 0 \\ \\ \varepsilon_2 u_2''(x) + u_2(x)u_2'(x) + u_1(x) - \tanh\left(\dfrac{x}{2\varepsilon_1}\right) = 0 \end{cases} \tag{6.4}$$

where

$$x \in (0,1), \quad u_1(0) = u_2(0) = 0, \quad u_1(1) = u_2(1) = \tanh(1/(2\varepsilon)).$$

For this example, we have an exact solution which is

$$u_1(x) = \tanh\left(\frac{x}{2\varepsilon_1}\right), \quad u_2(x) = \tanh\left(\frac{x}{2\varepsilon_2}\right).$$

In this example, we take $\vec{u}_0 = 1$ as our initial guess. For this system, $\alpha = 1$ is acceptable and a tolerance of 10^{-10} is used. In Table 6.1 and Figures 6.1 and 6.2, we see the spectral accuracy of our results.

Table 6.1: Error in Numerical Results, Example 1

	$\varepsilon_1 = 10^{-4}, \varepsilon_2 = 10^{-6}$		$\varepsilon_1 = 10^{-6}, \varepsilon_2 = 10^{-8}$	
N	u_1	u_2	u_1	u_2
16	2.74	3.04	1.00	1.00
32	2.43	4.60	2.42	5.26
64	1.19e-02	5.90e-02	3.14e-02	4.02e-02
128	8.36e-05	1.21e-03	9.15e-04	5.60e-03
256	7.93e-09	2.00e-07	1.66e-07	5.34e-06
512	1.44e-13	1.38e-11	1.35e-11	1.35e-09
1024	1.89e-13	1.38e-11	1.35e-11	1.35e-09

Absolute maximum error of simple coupled nonlinear system based on Burgers' equation example from [34]. For the first case, where $\varepsilon_1 = 10^{-4}$ and $\varepsilon_2 = 10^{-6}$, two sine transformations are used. For the second case, three sine transformations are used.

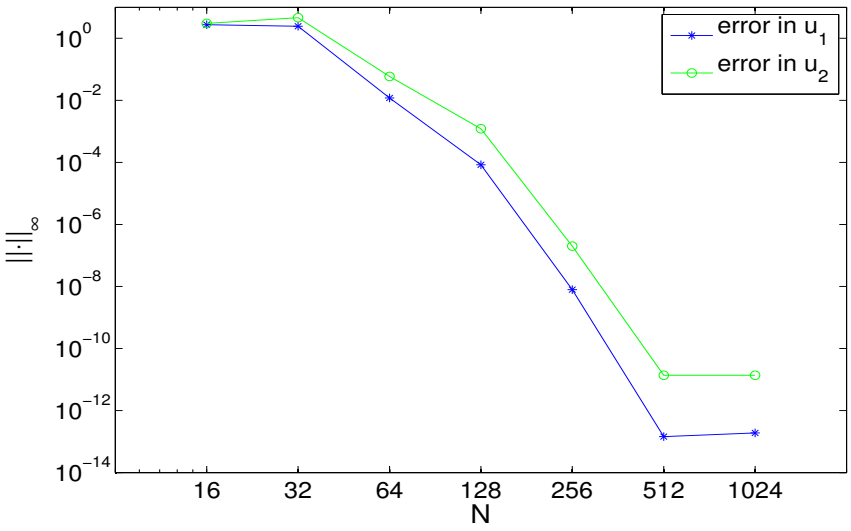

Figure 6.1: Log-log plot of maximum absolute error, with $\varepsilon_1 = 10^{-4}$, $\varepsilon_2 = 10^{-6}$, and two sine iterations.

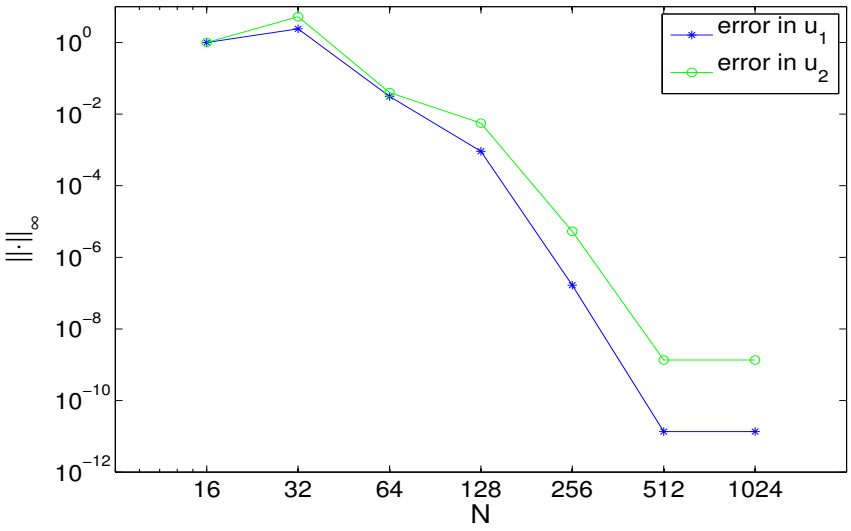

Figure 6.2: Log-log plot of maximum absolute error, with $\varepsilon_1 = 10^{-6}$, $\varepsilon_2 = 10^{-8}$ and three sine iterations.

6.2.2 Example 2, Comparison with Gracia *et al.*

We now look at an example from [19]

$$\begin{cases} \varepsilon_1 u_1''(x) - u_1(x) + 1 + (1 - u_1(x))^3 - e^{u_1(x)-u_2(x)} = 0 \\ \varepsilon_2 u_2''(x) - u_2(x) + 0.5 + (0.5 - u_2(x))^5 - e^{u_2(x)-u_1(x)} = 0 \end{cases} \tag{6.5}$$

where

$$x \in (0,1), \quad \vec{u}(0) = \vec{u}(1) = 0.$$

For this example, we do not have an exact solution. Instead, we look at the differences between solutions at successive values of N, as was done in [19]. This is a different way of dealing with no exact solution than was done previously; it was done intentionally to more closely mimic the procedure in [19]. For $N = 16, 32, 64, \ldots, 1024$, we interpolate the solution at N and then compare this to the previously computed solution at $N/2$ at the nodes used in computing the solution at $N/2$. We will refer to these differences as ΔE_N, such that ΔE_N is the maximum absolute error between solutions computed at N and $N/2$.

For this example, we need to employ values of $\alpha < 1$ in order for the Newton iterations to converge. This makes the convergence much slower and as such we increase our tolerance level to 10^{-4}. Additionally, we use a continuation approach for our initial guess. More specifically, for given values of ε_1 and ε_2, we use a solution to the system with $10\varepsilon_1$ and $10\varepsilon_2$. For $\varepsilon_1 = \varepsilon_2 = 10^0$, we use $\vec{u}_0 = 0$ as our initial guess. Although results were obtained using $\vec{u}_0 = 0$ for all values of $\varepsilon_1, \varepsilon_2$, these took a very long time to converge, especially for $\varepsilon_{1,2} \ll 1$

Comparing our results against those of [19] was somewhat difficult in that they do not report a specific perturbation parameter for their error. Since they are looking more at convergence rates, they give their error as a maximum over a set of perturbation parameters which are, approximately, $10^0 \leq \varepsilon_1 \leq 10^{-9}$ and $10^0 \leq \varepsilon_2 \leq 10^{-5}$. As such, two cases were looked at here for comparison. In the first case, we use $\varepsilon_1 = 10^{-4}, \varepsilon_2 = 10^{-2}$. In the second case, we use $\varepsilon_1 = 10^{-9}, \varepsilon_2 = 10^{-5}$. The second case is roughly a worst case scenario, and thus most comparable to results in [19]. In Figure 6.3 we see the solution profile for the first perturbation parameters. In Figures 6.4 and 6.5 we see the error for the first and second case respectively.

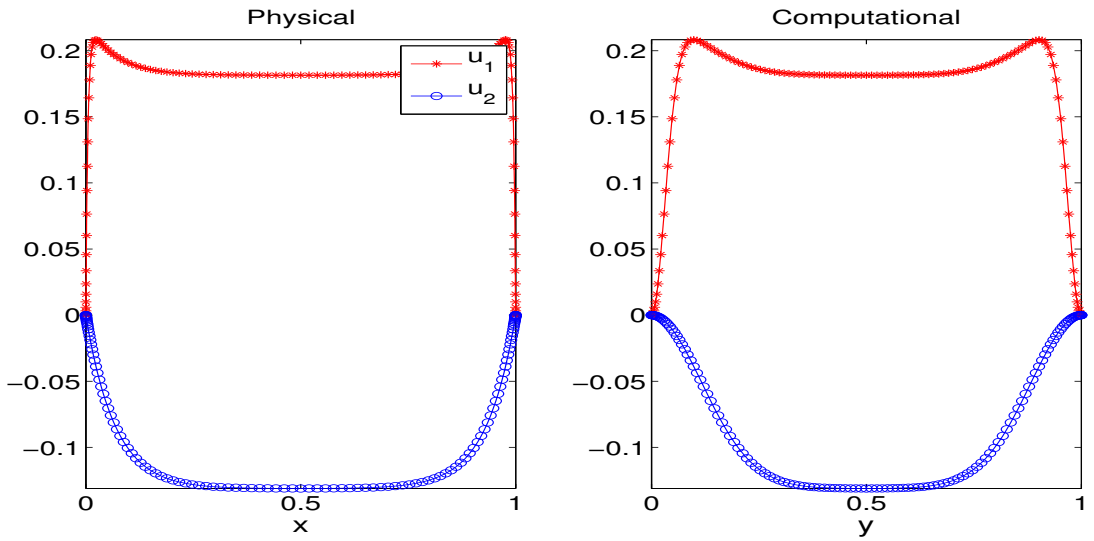

Figure 6.3: Solution profile on physical and computational domains with $\varepsilon_1 = 10^{-2}$ and $\varepsilon_2 = 10^{-4}$, one sine iteration and $N = 128$.

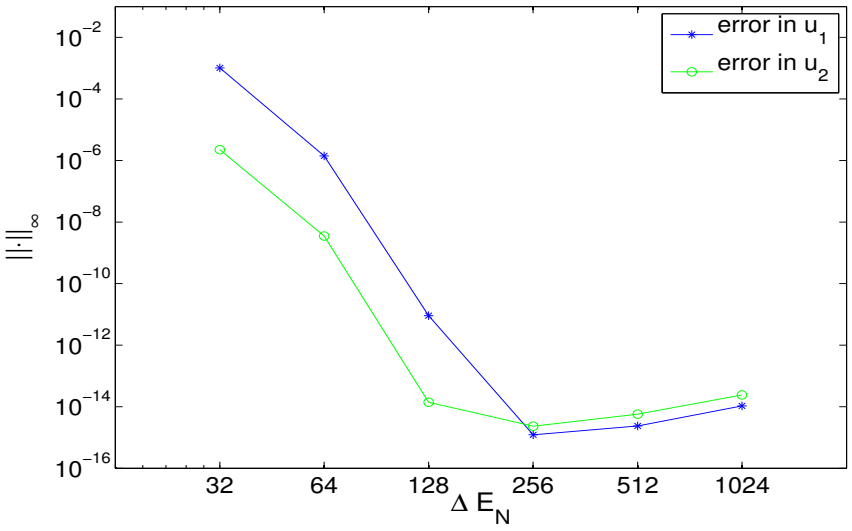

Figure 6.4: Log-log plot of maximum absolute error between successive solutions, with $\varepsilon_1 = 10^{-2}$ and $\varepsilon_2 = 10^{-4}$ and one sine iteration.

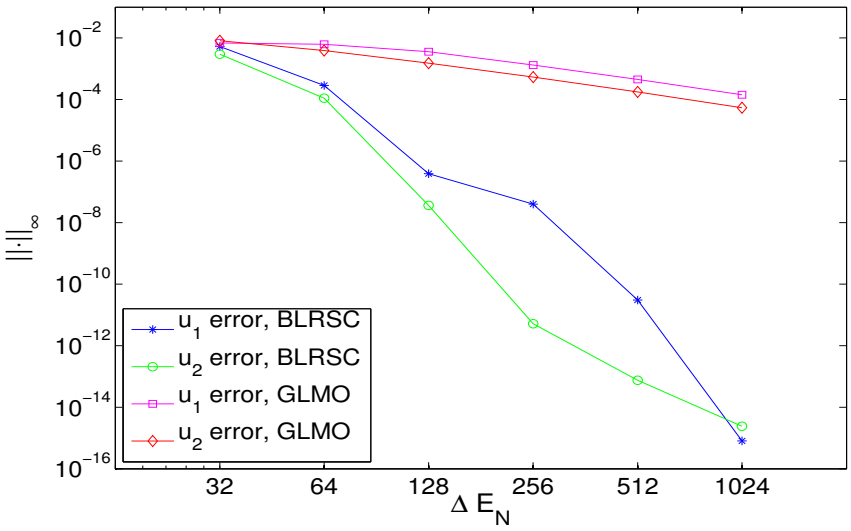

Figure 6.5: Log-log plot of maximum absolute error between successive solutions, with $\varepsilon_1 = 10^{-9}$ and $\varepsilon_2 = 10^{-5}$ and one sine iteration.

As can be seen in Figure 6.5 and 6.4, the BLRSC method is able to achieve much more accurate results for larger N. However, there are a few things to note. Firstly, results in [19] were obtained using far fewer iterations of the Newton's method. Our method requires step sizes smaller than one for many iterations and thus the convergence of the Newton iteration is slow. A more sophisticated method of choosing the step size would help improve this; also, a different method of solving the nonlinear equations may be more appropriate. Additionally, it appears that more sine transformations exacerbated the issues with the convergence of Newton's method. Thus, for the second case in example two, even with relatively small values of ε_1 and ε_2, only one sine transformation is used. Despite these shortcomings, the results for both examples one and two do show that the BLRSC method is applicable to singularly perturbed systems of nonlinear BVPs.

Chapter 7

Conclusion

This investigation has produced several interesting findings. First, the FH interpolants have been demonstrated to be excellent in interpolation and differentiation and, in conjunction with the BLRSC method, can be useful in solving singularly perturbed BVPs. Perhaps the numerical issues encountered with this FH method reduces its utility (in the sense of minimizing max-norm error) in solving singularly perturbed BVPs; but, it does provide greater flexibility in choosing collocation points. Also, especially for larger perturbation parameters, the FH method seems to be viable and competitive (compared to the standard Chebyshev method).

Second, in looking at the coupled systems of singularly perturbed BVPs, the BLRSC approach seems to be an excellent method, given its ease of implementation, efficiency in computation speeds, and ability to deal with a wide variety of systems with solutions having boundary layers, including different perturbation parameters. Not only does it outperform the methods based on Shishkin meshes and finite difference schemes in terms of absolute maximum error, but it is also is very competitive compared to other specialized spectral methods, such as the RSC-sinh method.

A number of avenues are available for future work. The first is to further test the FH interpolation method in solving coupled systems of BVPs. Along this vein, one can also improve the BLRSC method by incorporating greater adaptivity– by automatically choosing the number of sine transformations based on the value of ε and N. Perhaps, one could even automate which type of interpolation to use, depending on the blending parameter, the value of N, and the perturbation parameter. Finally, more work needs to be done with the nonlinear systems, particularly with how the nonlinear equation is solved.

Bibliography

[1] L. R. Abrahamsson, H. B. Keller, and H. O. Kreiss. Difference approximations for singularly perturbations of systems of ordinary differential equations. *Numer. Math.*, 22:367–391, 1974.

[2] Uri M. Ascher, Robert M. M. Mattheij, and Robert D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. Society for Industrial and Applied Mathematics: Philadelphia, 1995. 37

[3] R. Baltensperger. Improving the accuracy of the matrix differentiation method for arbitrary collocation points. *Appl. Numer. Math.*, 33:143–149, 2000. 64

[4] R. Baltensperger, J.-P. Berrut, and B. Noël. Exponential convergence of a linear rational interpolant between transformed Chebyshev points. *Math. Comp.*, 68:1109–1120, 1999.

[5] R. Baltensperger and M. Trummer. Spectral differencing with a twist. *SIAM J. Sci. Comput.*, 24:1465–1487, 2002.

[6] S. Bellew and E. O'Riordan. A parameter robust numerical method for a system of two singularly perturbed convection-diffusion equations. *Appl. Numer. Math.*, 51:171–186, 2009.

[7] J.-P. Berrut. Rational functions for guaranteed and experimentally well-conditioned global interpolation. *Comput. Math. Applic.*, 15:1–16, 1988. 7, 17

[8] J.-P. Berrut, R. Baltensperger, and H. D. Mittelmann. Recent developments in barycentric rational interpolation. *International Series of Numerical Mathematics*, 151:27–51, 2005. 6

[9] J.-P. Berrut, M. Floater, and G. Klein. Convergence rates of derivatives of a family of barycentric rational interpolant. *Appl. Numer. Math.*, 61:989–1000, 2011. 19

[10] J.-P. Berrut and G. Klein. Recent advances in linear barycentric rational interpolation. *J. Comput. Appl. Math.*, 259:95–107, 2014. 14, 16, 17, 25

[11] J.-P. Berrut and H. D. Mittelmann. Lebesgue constant minimizing linear rational interpolation of continuous functions over the interval. *Comput. Math. Appl.*, 33:77–86, 1997. 6

[12] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM Rev.*, 46:501–517, 2004. 4, 65

[13] L. Bos, S. D. Marchi, K. Hormann, and G. Klein. On the Lebesgue constant of barycentric rational interpolation at equidistant nodes. *Numer. Math.*, 121:462–471, 2012. 11, 62

[14] John P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications, Inc.: New York, 2000. 1

[15] Z. Cen. Parameter-uniform finite difference scheme for a system of coupled singularly perturbed convection-diffusion equations. *J. Sys. Sci. Complex.*, 18:498–510, 2005. 41, 42

[16] S. Chen, Y. Wang, and X. Wu. Rational spectral collocation method for a coupled system of singularly perturbed boundary value problems. *J. Comput. Math*, 29:458–473, 2011. 37, 47, 48, 49

[17] P. J. Davis. *Interpolation and Approximation*. Blaisell Pub. Co.: New York, 1963. 3

[18] M. Floater and K. Hormann. Barycentric rational interpolation with no poles and high rates of approximation. *Numer. Math.*, 107:315–331, 2007. 8, 10, 11, 12, 13, 14

[19] J. L. Gracia, F. J. Lisbona, M. Madaune-Tort, and E. O'Riordan. A system of singularly perturbed semilinear equations. *Lecture Notes in Computational Science and Engineering*, 69:163–172, 2009. 55, 57

[20] G. Klein. An extension of the Floater-Hormann family of barycentric rational interpolants. *Math. Comp.*, 82:2273–2292, 2013. 14, 16, 21, 25

[21] G. Klein and J.-P. Berrut. Linear rational finite differences from derivatives of barycentric rational interpolants. *SIAM J. Numer. Anal.*, 50:643–656, 2012. 17

[22] P. V. Kokotovic. Applications of singular perturbation techniques to control problems. *SIAM Review*, 26:501–550, 1984. 37

[23] N. Kopteva and E. O'Riordan. Shishkin meshes in the numerical solution of singularly perturbed differential equations. *International J. Numer. Anal. and Model.*, 7:393–415, 2009.

[24] T. Linß. On a set of singularly perturbed convection-diffusion equations. *J. Comput. and Appl. Math.*, 180:173–179, 2005.

[25] T. Linß. Analysis of an upwind finite difference scheme for a system of coupled singularly perturbed convection-diffusion equations. *SIAM J. Sci. Comput.*, 79:23–32, 2007.

[26] T. Linß and N. Madden. A finite element analysis of a coupled system of singularly perturbed reaction-diffusion equations. *App. Math. and Comput.*, 148:869–880, 2004.

[27] S. Matthews, J.J. H. Miller, E. O'Riordan, and G. I. Shishkin. *A parameter robust numerical method for a system of singularly perturbed ordinary differential equations, in: J.J.H. Miller, G.I. Shishkin, L. Vulkov (Eds.), Analytical and Numerical Methods for Convection-Dominated and Singularly Perturbed Problems*. Nova Science Publishers: New York, 2000. 42, 44, 45

[28] S. Matthews, E. O'Riordan, and G. I. Shishkin. A numerical method for a system of singularly perturbed reaction-diffusion equations. *J. Comput. and App. Math.*, 145:151–166, 2002. 37

[29] Jorge Nocedal and Stephan J. Wright. *Numerical Optimization*. Springer: New York, 2006. 52

[30] E. O'Riordan, J. Stynes, and M. Stynes. A parameter-uniform finite difference method for a coupled system of convection-diffusion two-point boundary value problems. *Numer. Math.: Theor. Meth. Appl.*, 1:176–197, 2008. 37

[31] E. O'Riordan and M Stynes. Numerical analysis of a strongly coupled system of two singularly perturbed convection-diffusion problems. *Adv. Comput. Math.*, 30:101–121, 2009.

[32] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press: New York, 1992. 52

[33] C. Schneider and W. Werner. Some new aspects of rational interpolation. *Math. Comp.*, 47:285–289, 1986. 6, 17

[34] T. Tang and M. Trummer. Boundary layer resolving pseudospectral methods for singular perturbation problems. *SIAM J. Sci. Comput.*, 17:430–438, 1996. 1, 29, 30, 31, 32, 52, 53, 65

[35] L. N. Trefethen. *Spectral Methods in Matlab*. SIAM: Philadelphia, 2000. 5, 6, 21, 24

Appendix A

Supplemental Mathematical Material

A.1 Lebesgue Constant

In a basic sense, the Lebesgue constant provides an error bound or condition number for linear interpolants (polynomial, rational, etc.) as will be shown. Given the set of $n+1$ distinct nodes $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ in the interval $x \in [a, b]$ and their corresponding values $f(x_0), f(x_1), \ldots, f(x_n)$, we define the interpolant $g(x)$ in terms of the Lagrangian polynomials

$$l_j(x) = \prod_{\substack{i=0 \\ j \neq i}}^{n} \frac{x - x_i}{x_j - x_i}$$

as

$$g(x) = \sum_{j=0}^{n} l_j(x) f(x_j).$$

As such, we can define the Lebesgue function associated with the function $g(x)$ as

$$\Lambda_n(x) = \sum_{j=0}^{n} |l_j(x)|$$

and the Lebesgue constant as

$$\Lambda_n = \max_{a \leq x \leq b} \Lambda_n(x).$$

As mentioned in Bos *et al.* [13], if every $f(x_j)$ value is given an absolute error or perturbation of at most ε, then the maximum distance between the perturbed interpolated data $h(x)$ and unperturbed interpolated data $g(x)$ in the interval $[a, b]$ is bounded as

$$\max_{a \leq x \leq b} |h(x) - g(x)| \leq \varepsilon \Lambda_n(x).$$

Since $g(x)$ is linear, and since Λ_n is an upper bound on the amplification error, the Lebesgue constant is also the condition number for the interpolation.

A.2 Condition Number

The condition number of a matrix gives an indication of how accurate or inaccurate a solution will be due to loss of precision or rounding error in matrices. The condition number of a matrix $A \in \mathbb{C}^{m \times n}$, $\kappa(A)$ is given as:

$$\kappa(A) = \|A\|\|A^{-1}\|$$

and can alternatively be given in terms of the singular values (as the ratio of the largest to the smallest singular values) if one uses the 2-norm. In this thesis, the 2-norm is used.

Computations with a matrix that is ill-conditioned ($\kappa(A)$ is "large") can be very sensitive to perturbations, i.e. it can cause large errors in solving the system $Ax = b$. However, the condition number is an upper bound for amplifications of such errors; as such, actual computations may yield smaller errors.

Appendix B

Numerical Stability Considerations

B.1 Considerations for Differentiation Matrices

B.1.1 General Differentiation Matrices

One important numerical implementation consideration of these formulas, as noted by Baltensperger in [3], is (for the diagonal elements) to sum the terms from least to greatest, in absolute value; this is sometimes referred to as the *negative sum trick*, or NST, with sorting. This avoids "smearing", or cancellation, errors.

B.1.2 Chebyshev Differentiation Matrices

The formulas for the Chebyshev differentiation matrix are:

$$D_{ij} = \frac{c_i}{c_j}\frac{(-1)^{i+j}}{x_i - x_j}, \qquad i \neq j \tag{B.1a}$$

$$D_{ii} = -\frac{x_i}{2(1-x_i^2)}, \qquad i \neq 0, N \tag{B.1b}$$

$$D_{00} = -D_{NN} = \frac{2N^2+1}{6}. \tag{B.1c}$$

where

$$x_j = \cos\left(\frac{\pi j}{n}\right), \qquad j = 0, 1, \ldots, N$$

One way to improve the stability, and to slightly reduce the computations needed in implementation, is to use the symmetry of the matrix

$$D_{N-i,N-j} = D_{i,j}.$$

Also, as mentioned before, one can also use the identity

$$D_{ii} = \sum_{\substack{j=0 \\ j \neq i}}^{N} D_{ij}.$$

Finally, one can use the definition of the Chebyshev points and trigonometric identities to transform the above equations to

$$D_{ij} = \frac{c_i}{c_j} \frac{(-1)^{i+j}}{\sin\left((i+j)\pi/(2N)\right) \, \sin\left((i-j)\pi/(2N)\right)}.$$

As mentioned in [34], computing the upper left hand section of the differentiation matrix, then using the symmetry property to compute the other part is more efficient and more accurate. However, the benefit, in terms of greater accuracy, only appears for relatively large values of N. In [12] it is shown that it is best to use the original formulas, Equation B.1b and the negative sum trick.

B.2 Other Numerical Considerations

B.2.1 Summing Large Numbers

As mentioned before, one of the main tricks is to sum numbers from smallest to largest (in absolute value). This avoids smaller numbers being ignored if very large numbers appear first.

When calculating the FH weights for large N, especially when using the Chebyshev points, the weights become extremely large. For certain values of d, they produced values of $\pm$Inf in Matlab. To avoid numerical errors, a number of tricks were employed to make the calculations more accurate. In some cases this resulted in much smaller values, values which Matlab could calculate. One technique, recommended by Piers Lawrence, is to use logarithms to avoid multiplication of large numbers. Instead of multiplication, one can sum (using ordering again) the log of each number and then exponentiate this sum, as seen below

$$\prod_{i=0}^{N} \alpha_i = e^{\sum_{j=0}^{N} \log(\alpha_j)}.$$