University of California
Santa Barbara

Applications of the Minimum Sobolev Norm and Associated Fast Algorithms

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy
in
Mathematics
by
Christopher Henry Gorman

Committee in charge:

Professor Shivkumar Chandrasekaran, Co-Chair
Professor Xu Yang, Co-Chair
Professor Hector Ceniceros
Dr. Xiaoye Sherry Li

June 2019

The dissertation of Christopher Henry Gorman is approved.

Professor Hector Ceniceros

Dr. Xiaoye Sherry Li

Professor Shivkumar Chandrasekaran, Co-Chair

Professor Xu Yang, Co-Chair

May 2019

Applications of the Minimum Sobolev Norm and Associated Fast Algorithms

by

Christopher Henry Gorman

Dedicated to the Blessed Holy Trinity: Father, Son, and Holy Spirit;
and my future wife: may I find you soon.

When I look at your heavens, the work of your fingers,
    the moon and the stars, which you have set in place,
what is man that you are mindful of him,
    and the son of man that you care for him?
— Psalm 8:3–4

For by him all things were created, in heaven and on earth, visible and invisible, whether thrones or dominions or rulers or authorities – all things were created through him and for him. And he is before all things, and in him all things hold together.
— Colossians 1:16–17

Acknowledgements

Graduate school has been a long and difficult road, and there are many people I would like to thank, not all of whom can be listed here. Without their help and support, I would not have graduated.

First and foremost, I would like to thank my doctoral committee: Shiv Chandrasekaran, Xu Yang, Hector Ceniceros, and Sherry Li. Shiv, you have been a great help through the entire process as I studied numerical linear algebra and fast algorithms, always reminding me about numerical stability and floating point arithmetic. Xu, thank you for teaching me about partial differential equations and asymptotic analysis. I am so glad to have both Shiv and Xu as doctoral advisors. Hector, thank you for teaching me about finite difference methods and pushing me to clearly articulate my results. Sherry, it was great to spend two summers working on STRUMPACK with you, Pieter, Gustavo, and Yang at Berkeley Lab. I learned a lot about computing, randomization, and software and algorithm development.

During my six years at UCSB, I have gained many friends. Those who have been with me from the beginning: Nancy, Steve, Jay, and Kyle. I have had a great time with you all. Those in Shiv's group: Kristen, Abhejit, Nithin, and Ethan. It has been great to know you while learning from Shiv. Medina, thank you for all the work you do in the Mathematics Department.

While living in Santa Barbara, I have gained many friends and mentors at Good Shepherd Lutheran Church. These include Pastor and Kitty; Ted and Andrea; and Chuck and Lois. Thank you for teaching me about life outside of school and work.

Thank you for the many friends I made in Reformed University Fellowship (RUF) at UCSB. Thank you especially to Johnathan (and Jaimeson) for leading RUF. You and Grace show God's love in Jesus to people on campus. Evan: it has been great to have

another graduate student who is going through the same process. Nathan and Stanley: it has been great to get to know you both these past few years.

Thank you to my family and especially my parents: Dad and Lin, and Mom and Paul. In particular, I must thank Vince and Megan: it has been great to get to know you and spend time with your family these years, especially Joe and Alice. Joe, you have been a great mentor and helped me early on in my doctoral program by encouraging me to recognize my dissertation is not the end, but rather the beginning, of my work.

My twin brother James: It was great going to Wabash College with you. Working on mathematics and physics homework late at night helped us succeed as undergraduates and continue on to earn our doctorates. Thank you for being my best friend.

Christopher Henry Gorman

University of California
Santa Barbara, CA 93106-3080

805-893-5306
gorman@math.ucsb.edu

Education

Ph.D. Mathematics June 2019
Emphasis: Computational Science and Engineering
Dissertation: Applications of the Minimum Sobolev Norm and Associated Fast Algorithms
Advisors: Shivkumar Chandrasekaran and Xu Yang
University of California, Santa Barbara, CA

M.A. Mathematics December 2014
University of California, Santa Barbara, CA

A.B. *Summa Cum Laude* Mathematics and Physics May 2013
Wabash College, Crawfordsville, IN

Research Experience

Graduate Intern – Systems Engineering Summer 2018
Mark Nussmeier, FLIR Systems, Inc., Goleta, CA
Performed design tests for thermal camera development

Graduate Student – Non GSRA Summer 2016/2017
Dr. Xiaoye Sherry Li, Lawrence Berkeley National Laboratory, Berkeley, CA
Assisted in the development of fast algorithms for Hierarchically Semi-Separable matrices

Research Graduate Student III/IV Summer 2014/2015
Dr. Nan Yu, Jet Propulsion Laboratory, Caltech, Pasadena, CA
2014: Performed error progation calculations and simulations for gravity gradiometer experiments
2015: Simulated atom interferometery to help development of equivalence principle test

Physics Research Assistant Summer 2012
Dr. K. Vollmayr-Lee, Bucknell University, Lewisburg, PA
Investigated structural glasses and found scaling predictions from Spin Glass theory apply to Silica

Physics Research Assistant Summer 2011
Dr. V.V. Kresin, University of Southern California, Los Angeles, CA
Studied nanoclusters and their formation while enhancing laboratory practices

Physics Research Assistant Summer 2010
Dr. M.J. Madsen, Wabash College, Crawfordsville, IN
Used Finite Element Analysis software interfaced with *Mathematica* to design compact torodial ion trap

Publications

Madsen, M.J. and Gorman, C.H., **"Compact toroidal ion-trap design and optimization,"** *Phys. Rev. A*, **82**, 043423 (2010)

K. Vollmayr-Lee, C.H. Gorman, and H.E. Castillo, **"Universal Scaling in the Strong Glass Former $\mathbf{SiO_2}$,"** J. Chem. Phys. **144**, 234510 (2016) (**JCP Editors' Pick**)

P. Ghysels, X. S. Li, C. Gorman, and F. H. Rouet, **"A robust parallel preconditioner for indefinite systems using hierarchical matrices and randomized sampling,"** *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Orlando, FL, 2017, pp. 897-906.

S. Chandrasekaran, C.H. Gorman, and H.N. Mhaskar, **"Minimum Sobolev norm interpolation of scattered derivative data,"** Journal of Computational Physics **365**, pp. 149–172 (2018)

C. Gorman, G. Chávez, P. Ghysels, T. Mary, F. H. Rouet, and X. S. Li, **"Robust and Accurate Stopping Criteria for Adaptive Randomized Sampling in Matrix-free HSS Construction,"** SIAM J. Sci. Comput., 2019. To appear.

Abstract

Applications of the Minimum Sobolev Norm and Associated Fast Algorithms

by

Christopher Henry Gorman

This dissertation focuses on the development, implementation, and analysis of fast algorithms for the Minimum Sobolev norm (MSN). The MSN method obtains a unique solution from an underdetermined linear system by minimizing a derivative norm in the appropriate Hilbert space. We obtain fast algorithms by exploiting the inherent structure of the underlying system. After performing an Inverse Discrete Cosine Transform, a small number of additional operations are required. Results show the method performs as well as Chebyshev interpolation when approximating smooth functions and better than a wide variety of smooth Chebyshev filters when attempting to approximate rough functions.

One chapter is devoted to analyzing a stochastic norm estimate which is useful when computing low-rank approximations of matrices. This estimate allows us to compute approximations with relative error close to machine precision, which previously was not possible.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The major goal of this line of research is to develop high order, numerically stable fast algorithms for solving elliptic partial differential equations using the Minimum Sobolev norm (MSN). This will require much more work than can be completed in one dissertation. The present work focuses on developing fast algorithms to solve interpolation and ordinary differential equation problems using the MSN method, which will be a stepping stone to understand the structure of the matrices arising in 2D and 3D PDEs. We introduce these ideas by discussing some of the problems present in interpolation methods and how the MSN method attempts to solve them.

1.1 Lagrange Interpolation and Known Difficulties

The well-known Weierstrass Approximation Theorem, which we reproduce for completeness, says continuous functions on compact, connected intervals can be approximated arbitrarily well by polynomials:

Theorem 1.1 (Weierstrass Approximation Theorem; Theorem 7.26 in [57])

If $f \in C[a,b]$, then there exists a sequence of polynomials $\{P_n\}_{n=1}^{\infty}$ such that

$$\lim_{n\to\infty} ||f - P_n||_{\infty,[a,b]} = 0. \tag{1.1}$$

Here,

$$||g||_{\infty,[a,b]} \equiv \sup_{x\in[a,b]} |g(x)| \tag{1.2}$$

is the supremum norm. When the interval $[a,b]$ is understood, we may write $||\cdot||_\infty$ in place of $||\cdot||_{\infty,[a,b]}$. This theorem shows that the set of polynomials $\mathcal{P}$ is dense in $C[a,b]$, the space of continuous functions on $[a,b]$. We let $\mathcal{P}_n$ denote all polynomials of degree at most n. This gives rise to an important concept: degree of approximation. We also have the following theorem:

Theorem 1.2 (Best Uniform Approximation of Continuous Functions; Section 1.1 in [55]) If $f \in C[a,b]$ and $n \in \mathbb{N}_0$, then there exists a unique $q_n \in \mathcal{P}_n$ so that

$$||f - q_n||_\infty = \inf_{p\in\mathcal{P}_n} ||f - p||_\infty . \tag{1.3}$$

We set

$$E_n(f) \equiv \inf_{p\in\mathcal{P}_n} ||f - p||_\infty \tag{1.4}$$

and have

$$E_0(f) \geq E_1(f) \geq E_2(f) \geq \cdots \to 0. \tag{1.5}$$

There are many ways to prove the Weierstrass Approximation theorem. In [57, Chapter 7], Rudin convolves against a polynomial kernel. This is useful theoretically but in practice, one may only have function and derivative information at particular points. In order to reconstruct the underlying function, we want to use these function and derivative values to build an approximation, frequently chosen to be a polynomial. This is interpolation.

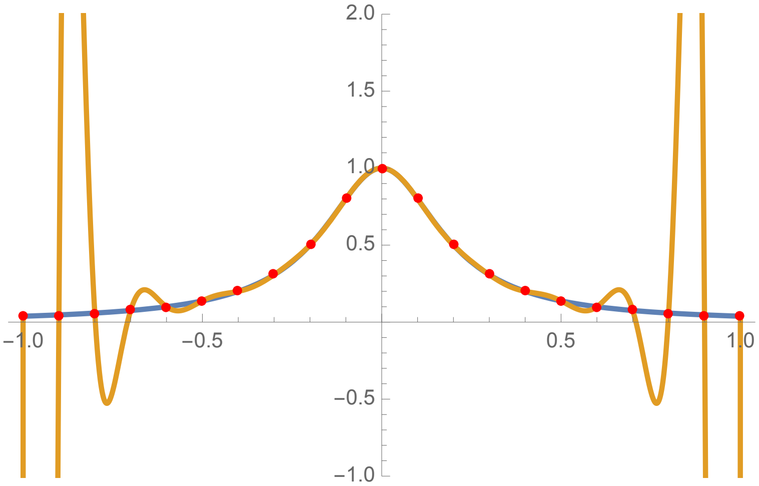


Figure 1.1: Here is an example of Lagrange interpolation of the Runge function $[1+25x^2]^{-1}$ using Lagrange interpolation with 21 equally-spaced nodes.

Interpolation Problem: Let $f \in C[-1,1]$ be continuous and specify a sequence of grid points

$$-1 \leq x_{1;n} < x_{2;n} < \cdots < x_{n;n} \leq 1. \tag{1.6}$$

for $n \in \mathbb{N}$. Determine a polynomial p_n with $\deg p_n = m(n)$ which satisfies the conditions

$$f(x_{k;n}) = p_n(x_{k;n}), \tag{1.7}$$

and ascertain under what restrictions on f and $\{x_{k;n}\}_{k=1}^{n}$ ensure

$$||f - p_n||_\infty \to 0, \quad n \to \infty. \tag{1.8}$$

A popular choice is to set $m(n) = n-1$, resulting in Lagrange interpolation. In Fig. 1.1, we see an example of Lagrange interpolation on equally-spaced nodes of the Runge function $[1+25x^2]^{-1}$ on $[-1,1]$. This function is analytic; even so, in [58] Runge proved that Lagrange interpolation diverges in this case. In particular, there are large oscillations near the boundary points.

In fact, much more is known about Lagrange interpolation, and we introduce notation which will make this discussion easier. Let

$$X = \{x_{k;n} \mid k = 1, \cdots, n; n \in \mathbb{N}\} \tag{1.9}$$

be an interpolatory matrix. Then, given $f \in C[-1,1]$, we have the following standard definitions [65, Chapter 1]:

$$\begin{aligned}
L_n(f, X, x) &= \sum_{k=1}^{n} \ell_{k,n}(X, x) f(x_{k;n}) \\
\Omega_n(X, x) &= \prod_{i=1}^{n} (x - x_{i;n}) \\
\ell_{k,n}(X, x) &= \frac{\Omega_n(X, x)}{\Omega_n'(X, x_{k;n})(x - x_{k;n})} \\
\lambda_n(X, x) &= \sum_{k=1}^{n} |\ell_{k,n}(X, x)| \\
\Lambda_n(X) &= ||\lambda_n(X, x)||_{\infty,[-1,1]} \, .
\end{aligned} \tag{1.10}$$

Naturally, $L_n(f, X, x)$ is the Lagrange interpolating polynomial of degree at most $n-1$ for the interpolation matrix X. The Lebesgue constants $\Lambda_n(X)$ are of critical importance, as we see

$$\begin{aligned}
|L_n(f, X, x) - f(x)| &\leq |L_n(f, X, x) - q_{n-1}(x)| + |f(x) - q_{n-1}(x)| \\
&\leq |L_n(f - q_{n-1}, X, x)| + E_{n-1}(f) \\
&\leq [\Lambda_n(X) + 1] \, E_{n-1}(f).
\end{aligned} \tag{1.11}$$

Here, $q_{n-1} \in \mathcal{P}_{n-1}$ is the minimizer in the supremum norm, and we note $L_n : \mathcal{P}_{n-1} \to \mathcal{P}_{n-1}$ is the identity map. Because $E_n(f) \to 0$, Lagrange interpolation converges when $\Lambda_n(X)E_{n-1}(f) \to 0$.

If we could find an interpolatory matrix Y so that $\Lambda_n(Y)$ is bounded, then $L_n(f, Y) \to f$ uniformly. Unfortunately, this is not the case. In [71], Vértesi references Faber (1914) as proving

$$\Lambda_n(X) > \frac{1}{8\sqrt{\pi}} \log n \tag{1.12}$$

for every X, showing $\Lambda_n(X)$ is unbounded. One popular set of interpolation nodes is the zeros of the Chebyshev polynomials:

$$\begin{aligned} T &= \{z_k^n \mid k = 1, \cdots, n; n \in \mathbb{N}\} \\ z_k^n &= \cos\left[\frac{\pi}{n}\left(n - k + \frac{1}{2}\right)\right]. \end{aligned} \tag{1.13}$$

The Chebyshev polynomials $T_n(x)$ are a set of orthogonal polynomials which will be discussed in detail in Sec. 2.1.2. In [13], it was shown

$$\Lambda_n(T) < 8 + \frac{2}{\pi}\log n, \tag{1.14}$$

For this reason, we see that the interpolatory matrix T is close to optimal and, coupled with fast interpolation methods, gives reason for its popularity. Better bounds for Lebesgue constants can be found in [62].

We also give bounds for equally-spaced points, setting

$$E = \left\{ -1 + 2\frac{k-1}{n-1} \,\middle|\, k = 1, \cdots, n; n \in \mathbb{N} \right\}. \tag{1.15}$$

In [68], Trefethen and Weideman give the bounds

$$\frac{2^{n-2}}{n^2} < \Lambda_n(E) < \frac{2^{n+3}}{n} \tag{1.16}$$

as well as referencing the asymptotic result and some of the history of equally-spaced interpolation. Clearly, the exponential growth of $\Lambda_n(E)$ helps quantify how much worse E is when compared with T.

This divergence is not restricted to equally-spaced point distributions, though. In fact, we have the following result:

Theorem 1.3 (Theorem 4.3 in [65])

For an interpolatory matrix $X \subset [-1, 1]$, there exists $h \in C[-1, 1]$ so that

$$\limsup_{n\to\infty} |L_n(h, X, x)| = \infty \tag{1.17}$$

for almost every $x \in [-1, 1]$.

So, there is no interpolatory matrix Y so that $||L_n(f, Y) - f||_\infty \to 0$ for all continuous functions f and the approximation error can be arbitrarily bad.

1.2 Possible Solutions to Divergence of Lagrange Interpolation

Although the previous result paints a bleak picture of Lagrange interpolation, this is true only in extreme situations. From [55, Chapter 1], we have the following theorem discussing how the degree of approximation is related to smoothness:

Theorem 1.4 (Jackson Inequality)
If $g \in C^k[-1, 1]$ and $g^{(k)}$ is α-Hölder with Hölder constant L, then for $n > k$, we have

$$E_n(g) \le \frac{c}{n^k}\left(\frac{1}{n-k}\right)^\alpha \tag{1.18}$$

with $c = 6^{k+1}e^k(1 + k)^{-1}L$.

This theorem shows that if g is merely α-Hölder continuous, then $||L_n(g, T) - g||_\infty \to 0$ by Eq. (1.11). As noted above, we can have $||L_n(f, E) - f||_\infty \not\to 0$ even when f is analytic.

We previously noted $L_n(f, E)$ has large oscillations in the Runge example. Because of this, there has been interest in Hermite-Fejér interpolation. Given an interpolatory matrix X, we let $H_n(f, X, x) \in \mathcal{P}_{2n-1}$ so that

$$\begin{aligned} H_n(f, X, x_{k;n}) &= f(x_{k;n}) \\ H_n'(f, X, x_{k;n}) &= 0. \end{aligned} \tag{1.19}$$

In this case, it can be shown $||H_n(f,T) - f||_\infty \to 0$ as $n \to \infty$ for all $f \in C[-1,1]$ [65, Chapter 5]. Unfortunately, this does not hold in general; in fact, for equally-spaced nodes we have the particularly bad result

$$\begin{aligned} f(x) &= x \\ \limsup_{n\to\infty} |H_n(f,E,x)| &= 0, \quad 0 < |x| \leq 1, \end{aligned} \tag{1.20}$$

which is discussed in [65, Chapter 6]. Controlling the derivative of the interpolation polynomial at the Chebyshev nodes appears to give sufficient control of the polynomial in order to obtain convergence for all continuous functions. Even so, while this gives convergence in the limit, it is not useful in practice because we purposefully limit the accuracy of interpolation near, but not at, interpolation nodes.

In another direction, Bernstein polynomials give up interpolation to get overall approximation. In fact, [22, 55] use Bernstein polynomials to prove the Weierstrass Approximation Theorem. The downside is that convergence to the solution is slow:

Theorem 1.5 (Error Estimate for Berstein polynomials; Theorem 1.2 in [55])
Suppose $g \in C[0,1]$ is α-Hölder with Hölder constant L and $B_n g$ is the Berstein polynomial of degree n for g; then

$$||g - B_n g||_{\infty,[0,1]} \leq \frac{3L}{2}\frac{1}{n^{\alpha/2}}, \tag{1.21}$$

and this bound in n cannot be improved.

This precludes it from being of much use in practice, especially when f is smooth.

By relaxing the condition $\deg L_n(f,X) \leq n-1$, Erdős was able to prove in [28] that, under some conditions on X, one could prove convergence for all continuous functions by choosing p_n so that $\deg p_n = c(X)n$, with c a constant depending only on X. The extension to all matrices X is shown in [65, Theorem 2.7]. This is important in practice, because we can not always choose the interpolation nodes. It is beneficial for a method to work well independent of node location, especially if, because of instrument specifications, data collection location cannot be modified. Unfortunately, these results require function values at arbitrary points, and this is not possible in practice.

1.3 Interpolation in Higher Dimensions

Up to this point, we have only talked about methods for approximating functions on $[a, b]$; even so, many problems in science and engineering are inherently two- and three-dimensional. A review of recent methods for multivariable polynomial interpolation can be found in [29, 30]. One challenge of interpolation in higher dimensions is choosing the correct polynomial space and point distribution. Now, the fact $\dim \mathcal{P}_{n-1} = n$ makes this easy in 1D but in higher dimensions there does not appear to be a simple way to choose a multivariable polynomial space of arbitrary dimension. Naturally, this is a topic of great interest. In [30], some standard methods discussed include tensor products of univariate polynomials, Gröbner bases, and ideal interpolation schemes.

1.4 Hermite and Birkhoff Interpolation

Hermite or Birkhoff interpolation problems involve interpolating function and derivative values. Hermite interpolation consists of interpolating function and derivative values up to a certain degree at interpolation nodes. Birkhoff interpolation is more general, allowing any combination of specified function and derivative values at nodes. Hermite interpolation is well-posed and can easily be solved in 1D. This is not the case for Birkhoff interpolation, where only certain combinations ensure a unique solution [43, 48]. The problem is even more complicated in dimension 2 and larger; see [49, 50] for a review of these topics. An additional challenge in multidimensional interpolation comes from proving error bounds and determining sufficient conditions for convergence.

1.5 Characteristics of Good Algorithms

This dissertation focuses on the development, implementation, and analysis of fast MSN methods. The ideas behind the MSN method will be discussed in the Sec. 1.6, but here we discuss good qualities that numerical algorithms should have, especially algorithms for approximation. These are high-order convergence, low computational complexity, and numerical stability.

Given a low-order method and a high-order method of similar computational cost, a faster-converging method is more effective and useful. In practice, there is always a limit to the amount of computational resources (memory, processor speed, or bandwidth), so a high-order method would be preferred as it would lead to less work overall. As mentioned before, Bernstein polynomials converge to all continuous function but do so at a slow rate. This alone does not necessarily disqualify the algorithm, but from Thm. 1.4, we know smoother functions can have better polynomial approximations. This incentivizes developing accurate approximations and algorithms to compute them.

While some methods may be of theoretical importance, algorithms will only be of practical value if there are efficient methods to compute them. The total cost should be of reasonable size, so that both the asymptotic growth ($O(\log n)$ or $O(n^3)$) and the explicit cost ($10^6 \log n$ and $\frac{2}{3}n^3$) are important. Because computational resources are always limited, asymptotics may not be as important as the prefactor hidden by Big O notation.

Finally, numerical stability is of critical importance. Almost all algorithms are implemented on computers using floating-point arithmetic, inevitably leading to small errors. It is necessary for practical algorithms to be immune to these changes; namely, small changes in inputs should lead to small changes in outputs. The condition number quantifies how much changes in outputs come from changes in inputs; a standard reference for numerical stability is [41].

1.6 The Minimum Sobolev Norm Method

We previously showed Lagrange interpolation does not work, for there can be large oscillations in the interpolating polynomial as seen in the Runge phenomenon, while using a polynomial of higher degree allows continuous functions to be approximated arbitrarily well. By combining these observations, the Minimum Sobolev norm (MSN) method was developed: a general method for computing approximate solutions to problems with linear constraints.

The MSN method has been used to solve problems in interpolation [16], Birkhoff interpolation [18], and partial differential equations [20], For simplicity, we assume we are performing approximations using algebraic polynomials, even though theoretical work often uses trigonometric polynomials. The main idea is this: given N linear constraints and polynomials of up to degree $M(N)$ contained in V, unknown coefficients a, correct values f, and a diagonal matrix D_s with condition number $O(M^s)$, the MSN solution solves the equation

$$\min_{Va=f} \left|\left|D_s a\right|\right|_2 . \tag{1.22}$$

We choose D_s so that $\left|\left|D_s a\right|\right|_2$ is a Sobolev norm. This implies that we seek an approximation which satisfies the linear constraints as well as having the smallest derivative norm. Here we focus on computing the minimum 2-norm solution because this dissertation investigates efficient numerical algorithms for MSN equations and we explicitly compute LQ factorizations; methods for p-norm minimization are discussed in [16, 18]. Additionally, this description is independent of dimension and node location. The parameter s determines which derivative of the polynomial approximation we wish to control. Larger s gives more derivative control on the approximation but leads Eq. (1.22) to have higher condition numbers. Great care is required to limit the effects of these condition numbers in order to ensure convergence to the underlying solution [18].

The technical challenge of this method is to determine the explicit form of $M(N)$ to ensure convergence to the solution. The methods in [16, 18] involve the close approximation of integral kernels by polynomials. The end result is that it is sufficient to choose $M(N) = C\eta^{-1}$, where η is the minimum separation between between interpolation nodes. Although this is a theoretically optimal result, knowing from [65] that this result cannot be improved except in the constant, it is not useful in practice because the constants from [16, 18] are difficult to explicitly compute. In practice, we have found that choosing the $M(N) = 2\pi\eta^{-d}$ is sufficient, where d is the dimension of the space. These details, along with implementation issues, will be discussed more in the next section.

One advantage of the MSN method is that we do not insist on forming a square linear system. In fact, it is necessary to take enough columns (more than twice the number of rows) in order to ensure a good approximation. Choosing the proper polynomial space was a challenge mentioned in Secs. 1.3 and 1.4.

1.7 MSN Interpolation Examples

We present some results of MSN interpolation on equally-spaced nodes in single and double precision for 1D and 2D. We do this to show that the difficulty of approximating functions on equally-spaced points arises from using suboptimal methods of interpolation rather than node location. These and similar results were published in [16, 18].

We can rewrite Eq. (1.22) as

$$\begin{aligned} &\min_{VD_s^{-1}x=f} ||x||_2 \\ &a = D_s^{-1}x. \end{aligned} \tag{1.23}$$

In order to compute the MSN solution, we must compute the minimum norm solution from Eq. (1.23). To do this, we must compute an LQ factorization of VD_s^{-1}, where L is a lower triangular matrix and Q is orthogonal. As previously mentioned, large s leads to greater derivative control but also gives VD_s^{-1} high condition number. Because

Algorithm 1 Slow, stable algorithm for solving MSN systems

```
 1: function SLOW_MSN_SOLVE($f$,$V$,$D_s$)        ▷ Solve $\min_{Va=f} ||D_s a||_2$.
 2:     Compute $P_1 L_1 Q_1 = V$ using an LQ factorization based on QRCP.
 3:     Determine permutation $\Pi$ such that $Q_1 D_s^{-1} \Pi$ has decreasing column norms.
 4:     Compute the SVD: $U \Sigma V^* = Q_1 D_s^{-1} \Pi$; only $U$ is stored.
 5:     Compute $P_2 L_2 Q_2 = U^* Q_1 D_s^{-1} \Pi$.
 6:     Solve $L_1 z = P_1^* f$.
 7:     Solve $L_2 y = P_2^* U^* z$.
 8:     Set $a = D_s^{-1} \Pi Q_2^* y$
 9:     return $a$
10: end function
```

of this, the standard pivoted LQ factorization based on QR with Column Pivoting is insufficient. A Rank-Revealing QR factorization based on [37] would be better, but an implementation is not readily available so we use another method presented here and described in [18]; see Alg. 1.

The unique feature of the algorithm may be Lines 4 and 5. Clearly, VD_s^{-1} is badly column-scaled. LQ factorizations can deal with poor row-scaling but not poor column-scaling. We compute the singular value decomposition $U\Sigma V^* = Q_1 D_s^{-1}\Pi$ in Line 4 and see $U^* Q_1 D_s^{-1}\Pi \approx \Sigma V^*$ to machine precision. This ensures we can accurately compute the pivoted LQ factorization $P_2 L_2 Q_2 = U^* Q_1 D_s^{-1}\Pi$ in Line 5. Thus, U is a preconditioner for numerical stability, showing that we can safely convert poor column-scaling to poor row-scaling. Using Alg. 1, the effective condition number of this problem appears to be that of V and not VD_s^{-1}.

Looking at the algorithm, we see two pivoted LQ factorizations and one SVD are required. Because we have N interpolation requirements and cN columns, this gives us $O(N^3)$ floating-point operations and $O(N^2)$ units of memory. At first glance, this does not seem too bad. If we are in dimension d with n^d tensor grid points, then $N = n^d$ and we require $O(n^{3d})$ flops and $O(n^{2d})$ units of memory. While these costs may be acceptable for $d = 1$ and bearable for $d = 2$, when $d = 3$ this is too great. Parallel computation would not be of much use here because the communication required for pivoted LQ and the SVD would cause the entire process to be extremely slow, although there has been recent work in reducing the communication cost in pivoted QR factorizations [23]. In order for these algorithms to be used when solving large, difficult problems, we need to investigate other methods. Similar costs arise when solving differential equations and this necessitates fast, structured algorithms. When developing fast algorithms, it is critical that we are able to convert the poor columns scaling to poor row scaling. The inherent structure of the linear system allows us to do this using careful factorizations.

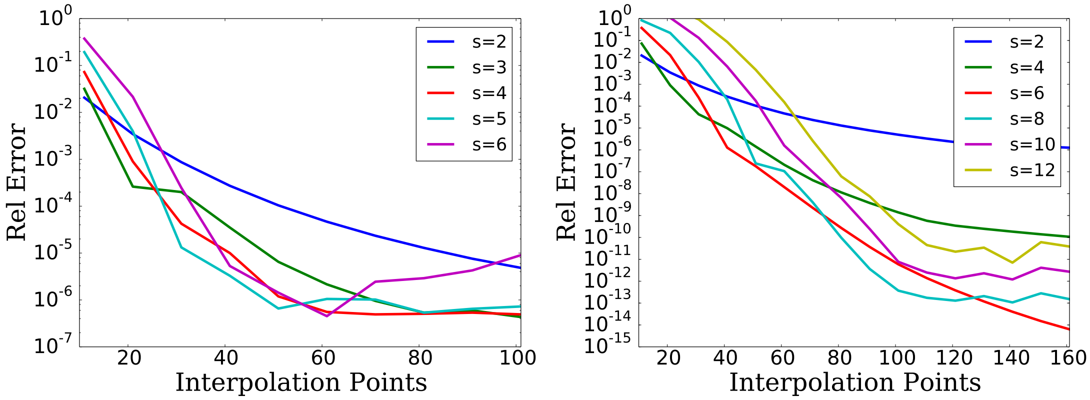
We present some examples of MSN interpolation and Birkhoff interpolation. The functions we approximate are

$$\begin{aligned}
f(x) &= \frac{1}{1+25x^2}\\
g(x,y) &= \frac{1}{1+25(x^2+y-0.3)^2} + \frac{1}{1+25(x+y-0.4)^2}\\
&\quad + \frac{1}{1+25(x+y^2-0.5)^2} + \frac{1}{1+25(x^2+y^2-0.25)^2}\\
h(x) &= g(x,-0.96).
\end{aligned} \tag{1.24}$$

Naturally, f is the usual Runge function. Here, g is a 2D function with Runge functions on one line, one circle, and two parabolas.

We remember machine precision is $2^{-23} \approx 1.2 \times 10^{-7}$ in single precision and $2^{-52} \approx 2.2 \times 10^{-16}$ in double precision. This is the smallest relative error that we could expect for any nonzero result. All of the plots show results for $||f-p||_\infty \,/\, ||f||_\infty$ for true function f and approximation p. The sup-norm is approximated by sampling the function at a large number of locations and taking the maximum.

The results for interpolating f are shown in Fig. 1.2. For single precision, all error curves decay toward 10^{-7} as we increase the number of points. The main exception is for $s = 6$, which starts to increase around 60 points. We believe this occurs because of rounding error. This would also make sense given for $s \in \{3,4,5\}$, the error curves hover close to 10^{-6}. We see a similar results for double precision. In this case, the beginnings



(a) Single Precision

(b) Double Precision

Figure 1.2: Relative error results for MSN interpolation on equally-spaced points on the function $f(x)$ from Eq. (1.24) for various s values using single and double precision.

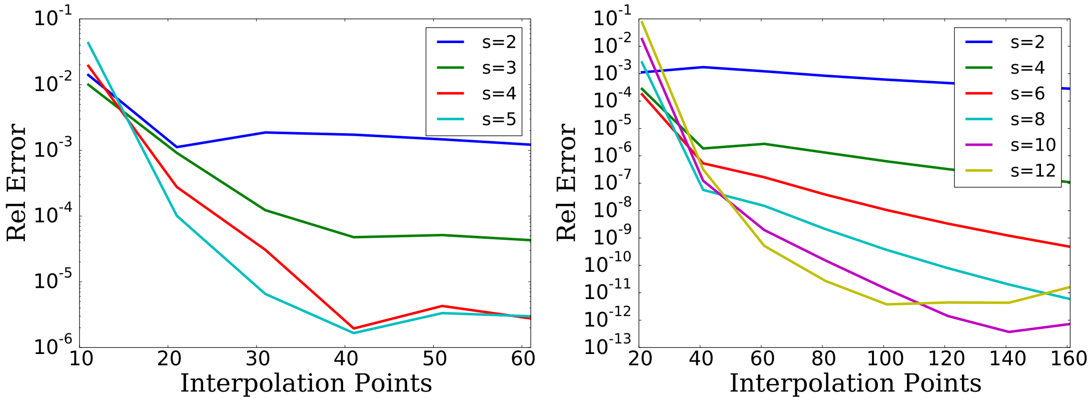
of the U-shaped error curve seem present for $s \in \{8, 10, 12\}$.

In Fig. 1.3, we have the results of MSN Birkhoff interpolation in 1D for h. Although the error curve for $s = 2$ in single and double precision hovers around 10^{-3}, the other error curves decay to machine precision. The beginning of a U-shaped error curve may be seen for $s \in \{4, 5\}$ in single precision and $s \in \{10, 12\}$ in double precision. The errors are low, although they may be slightly larger than those we see in regular MSN interpolation. This could stem from the fact the condition number is inherently larger for Birkhoff interpolation than for interpolation.

In Fig. 1.4, we have results for MSN Birkhoff interpolation in 2D for g from Eq. (1.24). In every case the error decreases with increasing data except for $s = 5$ with single precision. In this case, we may start to see the beginning of the effects of roundoff error. The challenge for 2D problems is the long time required to run Alg. 1.

1.8 Dissertation Outline

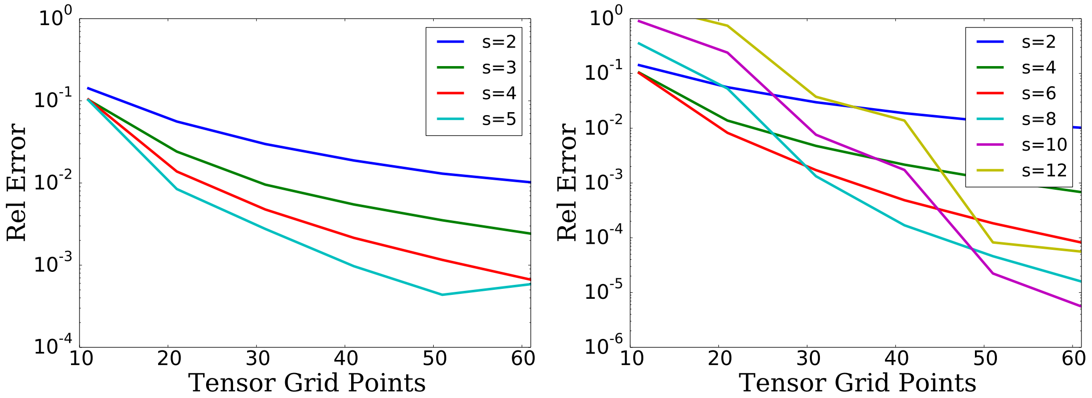
As we noted above, the slow methods for solving problems using MSN become difficult in 2D and practically impossible in 3D due to memory requirements and flop count. With the eventual desire to use MSN to solve 3D PDEs, we will need to take advantage of



(a) Single Precision

(b) Double Precision

Figure 1.3: Relative error results for MSN birkhoff interpolation on equally-spaced points using both function and derivative values on the function $h(x)$ from Eq. (1.24) for various s values using single and double precision.



(a) Single Precision

(b) Double Precision

Figure 1.4: Relative error results for MSN birkhoff interpolation on equally-spaced tensor-grid points using both function and derivative values on the function $g(x)$ from Eq. (1.24) for various s values using single and double precision.

everything we can. Keeping this in mind, the focus of this dissertation will be developing fast algorithms for solving interpolation and differential equations using the MSN method on Chebyshev nodes and express our solution in a Chebyshev polynomial basis. We review notation conventions and structured matrices in Chapter 2. In Chapter 3, we review some of the properties of Chebyshev-Vandermonde matrices which arise when developing these fast algorithms. Matrix factorizations important for interpolation problems are discussed in Chapters 4 and 5. Using these factorizations, we present examples of MSN approximation in Chapter 6. Next, we present new proofs showing that our fast methods will converge to the solution under minimal smoothness assumptions of the underlying function in Chapter 7. We investigate fast algorithms for Boundary Value Problems for ODEs in Chapter 8.

In the Chapter 9, we discuss results related to randomized low-rank approximations, unrelated to the previous work. Some of this was discussed in [33] but more details and examples will be shown here.

1.9 Algorithms Similar to the MSN Method

The ideas pursued in this dissertation are similar to those used by Chebfun [26], a software package in MATLAB [38] which attempts to have the "feel" of symbolic software with the speed of numerics. The book Approximation Theory and Approximation Practice [66] uses Chebfun to introduce the field of Approximation Theory. Here, we focus on investigating fast algorithms based on values computed on Chebyshev polynomial roots. This is similar to the fast algorithms present in Chebfun, which computes values on the Chebyshev polynomial extrema. The book Exploring ODEs [67] also uses Chebfun to introduce advanced differential equation topics. The methods in [67] are built on the work from [5, 25, 74] and are incorporated into Chebfun. Although spectral methods are well-known [10], [5] *adds* additional rows to the square linear system to impose boundary or other requirements instead of replacing rows. Naturally, this requires increasing the degree of the approximation. The work presented here does not force a square system, which allows us to add a finite number of additional requirements which do not affect the asymptotic complexity of the overall algorithm. From [67, Appendix A], it appears that Chebfun uses standard dense linear algebra algorithms to solve its ODEs. This is unfortunate, because the linear systems arising from ODEs are highly structured when approximated on Chebyshev nodes. This dissertation will show this structure and construct associated fast algorithms. The work here could be used to speedup the Chebfun ODE solver.

Chapter 2

Notation Convention, Structured Rotations, and Kronecker Products

We begin with some notation conventions before reviewing standard orthogonal matrices and looking at the Kronecker product and how it affects fast matrix-vector multiplication. One standard reference for matrix-related topics is [32]. These topics will then be used to look at matrix factorizations of Chebyshev-Vandermonde (C-V) matrices in Chapters 4 and 5.

2.1 Notation and Conventions

In this section, we present most of the notation conventions we will use throughout this dissertation. Probability theory is only used in Chapter 9, so we will not discuss the specifics until then.

2.1.1 Order Notation and Constant Convention

Throughout this dissertation, we will be using constants such as A, B, C that are absolute. They may be used in the same set of equations even though their value changes between inequalities. If the constant is not absolute but depends on a variable (say s), then we will write C_s or $C(s)$.

We review Big O notation; one reference is [35, Chapter 9]. We assume $f, g : [0, \infty) \to [0, \infty)$ and write

$$f(x) = O(g(x)) \tag{2.1}$$

when

$$f(x) \leq Cg(x), \quad C > 0, \quad x \geq N. \tag{2.2}$$

Thus, f is asymptotically bounded above by g. Similarly,

$$f(x) = \Omega(g(x)) \tag{2.3}$$

when

$$f(x) \geq Cg(x), \quad C > 0, \quad x \geq N. \tag{2.4}$$

Naturally, f is asymptotically bounded below by g. Finally,

$$f(x) = \Theta(g(x)) \tag{2.5}$$

when

$$f(x) = O(g(x)) \quad \text{and} \quad f(x) = \Omega(g(x)), \tag{2.6}$$

so that f is asymptotically bounded above and below by g.

2.1.2 Chebyshev Polynomials

The nth Chebyshev polynomial T_n is defined as

$$T_n(x) \equiv \cos\left[n \arccos x\right], \quad x \in [-1, 1]\,. \tag{2.7}$$

We will show the following recurrence relation holds, which is useful because of the restrictions in the previous definition:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x). \end{aligned} \tag{2.8}$$

From the definition, it is clear that

$$T_n(\cos\theta) = \cos n\theta \tag{2.9}$$

and

$$\begin{aligned} \max_{x\in[-1,1]} |T_n(x)| &= T_n(1) \\ &= 1. \end{aligned} \tag{2.10}$$

We recall the roots $\{z_k^n\}$ of T_n:

$$z_k^n = \cos\left[\frac{\pi}{n}\left(n - k + \frac{1}{2}\right)\right], \quad k \in \{1, \cdots, n\}. \tag{2.11}$$

With this convention, we are numbering the Chebyshev roots from negative to positive.

We now prove the recurrence relation of Eq. (2.8). First, we have equality for T_0 and T_1. We have the following property for multiplying cosine functions:

$$\cos\theta\cos\varphi = \frac{1}{2}\left[\cos(\theta+\varphi) + \cos(\theta-\varphi)\right], \tag{2.12}$$

from which it follows

$$\cos n\theta\cos\theta = \frac{1}{2}\left[\cos(n+1)\theta + \cos(n-1)\theta\right]. \tag{2.13}$$

This equation along with Eq. (2.9) gives us

$$T_n(x)T_1(x) = \frac{1}{2}\left[T_{n+1}(x) + T_{n-1}(x)\right] \tag{2.14}$$

when $n \geq 1$; rearranging and noting $T_1(x) = x$ gives us the desired recurrence relation, making it clear that T_n is a polynomial.

2.1.3 Function Spaces

In this work we will primarily deal with the Sobolev spaces. If $g : [-\pi, \pi] \to \mathbb{C}$ is periodic and integrable, then we can define Fourier coefficients

$$a_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} g(\theta) e^{-ik\theta} d\theta. \tag{2.15}$$

Under mild smoothness assumptions, it is well known that we have equality of the function and its Fourier series:

$$g(\theta) = \sum_{k \in \mathbb{Z}} a_k e^{ik\theta}. \tag{2.16}$$

From here, we define the Sobolev s-norm of g to be

$$||g||_s^2 \equiv \sum_{k \in \mathbb{Z}} (1 + |k|)^{2s} |a_k|^2 , \tag{2.17}$$

and the Sobolev space

$$H_s \equiv \{g \mid ||g||_s < \infty\} . \tag{2.18}$$

Even so, the focus here will be on $[-1, 1]$ because we wish to approximate non-periodic functions. To do so, we look at $f : [-1, 1] \to \mathbb{R}$ and have the Chebyshev series expansion

$$f(x) = \sum_{k=0}^{\infty} b_k T_k(x), \tag{2.19}$$

where

$$\begin{aligned} b_k &= \frac{2}{\pi} \int_{-1}^{1} \frac{f(x) T_k(x)}{\sqrt{1 - x^2}} dx \\ &= \frac{2}{\pi} \int_{0}^{\pi} f(\cos\theta) \cos k\theta d\theta. \end{aligned} \tag{2.20}$$

After a change of coordinates, we obtain the Fourier series:

$$f(\cos\theta) = \sum_{k=0}^{\infty} b_k \cos k\theta. \tag{2.21}$$

Thus, we similarly define the Sobolev s-norm in this case:

$$||f||_s^2 \equiv \sum_{k=0}^{\infty} (1+k)^{2s} |b_k|^2 . \tag{2.22}$$

We will sometimes write $||a||_s$ or $||b||_s$ in place of $||f||_s$ or $||g||_s$. The importance of H_s comes from the fact our algorithms minimize $||\cdot||_s$ in the appropriate space of polynomials.

We will focus on the case when $s > \frac{1}{2}$, because then $H_s \subseteq C$. Similarly, for $s > m + \frac{1}{2}$, we have $H_s \subseteq C^m$. Finally, we also have $C^{m,\alpha} \subseteq H_s$ for $s < m + \alpha + \frac{1}{2}$, where $C^{m,\alpha}$ is the space of m continuously differentiable functions whose mth derivative is α-Hölder continuous. These results are not difficult to show but we postpone their proof until Sec. 7.4.

2.1.4 Matrix Notation and Norm Definitions

Throughout this work, we use notation similar to that in [32]. Given a matrix $A \in \mathbb{R}^{m\times n}$, we let $A_{i,j}$ (or A_{ij}, $a_{i,j}$, and a_{ij}) denote the entry of A at the ith row and jth column. For index sets $I = \begin{bmatrix} i_1 & i_2 & \cdots & i_r \end{bmatrix}$ and $J = \begin{bmatrix} j_1 & j_2 & \cdots & j_s \end{bmatrix}$, we have the matrix subblock

$$A(I,J) = \begin{bmatrix} A_{i_1,j_1} & A_{i_1,j_2} & \cdots & A_{i_1,j_s} \\ A_{i_2,j_1} & A_{i_2,j_2} & \cdots & A_{i_2,j_s} \\ \vdots & \vdots & \ddots & \vdots \\ A_{i_r,j_1} & A_{i_r,j_2} & \cdots & A_{i_r,j_s} \end{bmatrix}. \tag{2.23}$$

At times we may write $A_{I,J}$ instead of $A(I,J)$. Similarly, $A(I,:)$ denotes the submatrix of A with rows in I while $A(:,J)$ denotes the submatrix of A with columns in J. Even though we will focus on real-valued matrices in this work, our notation also carries over to complex-valued matrices. We let A^T and A^* denote the transpose and conjugate transpose of A, respectively. Additionally, we define

$$|A|_{ij} \equiv |A_{ij}| ; \tag{2.24}$$

that is, $|A|$ is matrix we obtain when taking the absolute value of each element in A.

Given a vector $x \in \mathbb{R}^n$, we let E be the permutation which inverts the entries on x; that is,

$$Ex = \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_2 \\ x_1 \end{bmatrix}. \tag{2.25}$$

Clearly, this implies E has 1's on the antidiagonal and zeros everywhere else:

$$E = \begin{bmatrix} & & & & 1 \\ & & & 1 & \\ & & \iddots & & \\ & 1 & & & \\ 1 & & & & \end{bmatrix}. \tag{2.26}$$

We will also let Π denote the circular downshift permutation when right multiplication is performed:

$$\Pi x = \begin{bmatrix} x_n \\ x_1 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{bmatrix}. \tag{2.27}$$

Thus, we have

$$\Pi = \begin{bmatrix} 0 & & & & & 1 \\ 1 & 0 & & & & \\ & 1 & 0 & & & \\ & & & \ddots & & \\ & & & 1 & 0 & \\ & & & & 1 & 0 \end{bmatrix}. \tag{2.28}$$

We also have the following relation for left mulitplication:

$$\begin{bmatrix} v_1 & v_2 & \cdots & v_{n-1} & v_n \end{bmatrix} \Pi = \begin{bmatrix} v_2 & v_3 & \cdots & v_n & v_1 \end{bmatrix}. \tag{2.29}$$

Furthermore, we will make frequent use of D_s, the diagonal scaling matrix that arise in MSN interpolation. In particular, we let

$$D_s = \operatorname{diag}\left(1^s, 2^s, \cdots, M^s\right). \tag{2.30}$$

The specific size of D_s will be context dependent but will always have this form. There is nothing which requires s to be a (positive) integer, but we will usually assume this to be the case.

A general vector or matrix norm will be denoted by $||\cdot||$. Given $x \in \mathbb{R}^n$, the vector p-norms are defined as

$$||x||_p \equiv \begin{cases} \left(\sum_{k=1}^{n} |x_k|^p\right)^{1/p}, & p \in [1, \infty) \\ \max_{k=1,\cdots,n} |x_k|, & p = \infty \end{cases}. \tag{2.31}$$

For $A \in \mathbb{R}^{m\times n}$, the corresponding induced matrix p-norms are

$$||A||_p \equiv \sup_{||x||_p=1} ||Ax||_p. \tag{2.32}$$

Throughout this dissertation we will primarily be using the matrix 2-norm. Even so, there may be times when we use the Frobenius norm, which we sometimes call the F-norm:

$$||A||_F \equiv \sqrt{\sum_{k=1}^{m}\sum_{j=1}^{n} |A_{ij}|^2}. \tag{2.33}$$

In Chapter 9 only, we will use the Schatten p-norms [7, Chapter 4]:

$$||A||_{s,p} \equiv \begin{cases} \left(\sum_{k=1}^{\min(m,n)} \sigma_k^p\right)^{1/p}, & p \in [1, \infty) \\ \sigma_1, & p = \infty \end{cases}. \tag{2.34}$$

Here, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$ are the singular values of A. From this definition, it is clear $||A||_{s,\infty} = ||A||_2$ and $||A||_{s,2} = ||A||_F$.

Although there are many different norms used throughout this dissertation, the particular norm should be clear from context.

2.1.5 DCT Convention

All of the code for this dissertation was written in Julia [6], Version 0.5. If C_{II} is Julia's $n \times n$ DCT (the unitary version of DCT-II), then we require

$$C^{-1} = DC_{II}E, \tag{2.35}$$

where

$$D = \frac{1}{\sqrt{n}} \operatorname{diag}\left(1, \sqrt{2}, \cdots, \sqrt{2}\right). \tag{2.36}$$

This will also be discussed later in Chapter 3.

2.2 Rotations and Structured Factorizations

Householder reflectors and Givens rotations are standard orthogonal matrices which allow us to selectively zero entries of a matrix. Both will be important in our matrix factorizations, for our fast algorithms require us to compute the LQ factorization of Chebyshev-Vandermonde matrices.

2.2.1 Householder Reflectors

One special type of orthogonal matrix is a Householder reflector. Given distinct nonzero x and y with the same length (that is, $x \neq y$ and $||x||_2 = ||y||_2 > 0$), we wish to find an orthogonal matrix P such that $Px = y$. There is a reflection which does this: we set $v = x - y$ and choose

$$P = I - \frac{2}{v^*v}vv^*. \tag{2.37}$$

With this choice, we see

$$Px = x - \frac{2v^*x}{v^*v}v$$

Algorithm 2 Householder Reflector

```
function HOUSE($x$)    ▷ Numerically stable way to compute Householder Reflector
    $m = \text{length}(x)$
    $v = \text{zeros}(m)$
    $\sigma = ||x(2:m)||_2^2$
    $\mu = \sqrt{x_1^2 + \sigma}$
    if $x_1 \leq 0$ then
        $v_1 = x_1 - \mu$
    else
        $v_1 = -\frac{\sigma}{x_1+\mu}$
    end if
    $v = v/\, ||v||_2$
    return $v$
end function
```

$$\begin{aligned}
&= x - \frac{2\left(x^*x - x^*y\right)}{x^*x - 2x^*y + y^*y}\left(x - y\right) \\
&= x - \frac{x^*x - 2x^*y + y^*y}{x^*x - 2x^*y + y^*y}\left(x - y\right) \\
&= y
\end{aligned} \tag{2.38}$$

From the line 2 to line 3, we are using the fact that $||x||_2 = ||y||_2$. The previous work also holds when $x, y \in \mathbb{C}^n \setminus \{0\}$.

We frequently want to choose a vector parallel to e_1, allowing us to zero most of the entries of x. In this case, we choose $v = x - ||x||_2\, e_1$. One difficulty when v is computed numerically is the catastrophic cancellation that can occur in v_1. Care must be taken in order to ensure this does not happen; see Alg. 2, which has been modified from the version in [32, Alg. 5.1.1].

Matrix-vector products involving Householder reflector P can be computed quickly by taking advantage of its structure; it can be computed and applied in $O(n)$ flops. This implies that a small, constant number of reflectors can be applied with total cost $O(n)$. Also, in practice we never need to explicitly store P, only v. In this work, we will be assuming that our reflection vector v has unit length.

2.2.2 Givens Rotations

We make use of Givens rotations, so we review them and their ability to selectively zero entries of a matrix. This is important because our matrices are structured. A Givens rotation G is the identity matrix with a rank-2 correction:

$$G\left(\left[i,j\right],\left[i,j\right]\right)=\begin{bmatrix} c & s \\ -s & c \end{bmatrix}. \tag{2.39}$$

Here, $c=\cos\theta$ and $s=\sin\theta$ for some θ. We do not need to determine θ explicitly in practice, for we require

$$\begin{bmatrix} \alpha & \beta \end{bmatrix}\begin{bmatrix} c & s \\ -s & c \end{bmatrix}=\begin{bmatrix} r & 0 \end{bmatrix}. \tag{2.40}$$

Thus, we are finished if we can write c and s in terms of α and β. This is easy, though, and a numerically stable way to compute c and s is given in Alg. 3. Our convention here differs from others (such as [32, Alg. 5.1.3] or [8]) by the fact that we ensure $r=\sqrt{\alpha^2+\beta^2}$ in Eq. (2.40). The total cost for computing a Givens rotation and applying it to a vector is $O(1)$. This allows us to compute and apply $O(n)$ Givens rotations for $O(n)$ total cost.

2.3 Kronecker Products and Fast Matrix-Vector Multiplication

If A and B are structured matrices which allow for fast matrix-vector multiplication, then intuitively it should be possible to compute matrix-vector products involving $A\otimes B$ quickly as well. In this section we demonstrate particular instances that are important for the present research. If $A\in\mathbb{R}^{m\times n}$, $B\in\mathbb{R}^{k\times\ell}$, and $g_r=g\left(\left[r-1\right]\ell+1\!:\!r\ell\right)$ with $r\in\{1,\cdots,n\}$, then, in block form, we have

Algorithm 3 Givens Rotation

```
function GIVENS($\alpha$,$\beta$)    ▷ Numerically stable way to compute Givens rotations
    if $|\alpha| \geq |\beta|$ then
        if $\alpha = 0$ then
            $c = 1$
            $s = 0$
        else if $\beta = 0$ then
            $c = \operatorname{sign}(\alpha)$
            $s = 0$
        else
            $\tau = -\frac{\beta}{\alpha}$
            $c = \operatorname{sign}(\alpha)/\sqrt{1+\tau^2}$
            $s = c\tau$
        end if
    else
        if $\alpha = 0$ then
            $s = -\operatorname{sign}(\beta)$
            $c = 0$
        else
            $\tau = -\frac{\alpha}{\beta}$
            $s = -\operatorname{sign}(\beta)/\sqrt{1+\tau^2}$
            $c = s\tau$
        end if
    end if
    return $c$, $s$
end function
```

$$(A \otimes B) g = \begin{bmatrix} a_{11}Bg_1 + a_{12}Bg_2 + \cdots + a_{1n}Bg_n \\ a_{21}Bg_1 + a_{22}Bg_2 + \cdots + a_{2n}Bg_n \\ \vdots \\ a_{m1}Bg_1 + a_{m2}Bg_2 + \cdots + a_{mn}Bg_n \end{bmatrix}. \tag{2.41}$$

If we set

$$G = \begin{bmatrix} g_1 & g_2 & \cdots & g_n \end{bmatrix}, \tag{2.42}$$

then we can compute the above matrix product like

$$\begin{aligned} H &= BGA^T \\ &= \begin{bmatrix} h_1 & h_2 & \cdots & h_m \end{bmatrix}, \end{aligned} \tag{2.43}$$

where h_r has k rows, and

$$\begin{aligned} (A \otimes B) g &= h \\ &= \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix}. \end{aligned} \tag{2.44}$$

It is this insight which allows us to compute matrix-vector multiplication quickly when working with tensor products of matrices, especially if A or B allow for fast matrix-vector products. We merely note that the above product (if A and B are square matrices of size n) has cost $O(n^4)$ flops if we explicitly form $A \otimes B$. By taking advantage of the tensor structure, we can reduce the cost of multiplication to $O(n^3)$ flops. Naturally, this does not count memory transfer.

We will need to do something slightly different if we are performing matrix-matrix multiplication. We see

$$(A \otimes B) \begin{bmatrix} g_1 & \cdots & g_s \end{bmatrix} = \begin{bmatrix} (A \otimes B) g_1 & \cdots & (A \otimes B) g_s \end{bmatrix}$$

$$\sim \begin{bmatrix} BG_1A^T & \cdots & BG_sA^T \end{bmatrix}. \tag{2.45}$$

This could be computed quickly by

$$\begin{aligned} B\begin{bmatrix} G_1 & \cdots & G_s \end{bmatrix} &= \begin{bmatrix} H_1 & \cdots & H_s \end{bmatrix} \\ \begin{bmatrix} H_1 \\ \vdots \\ H_s \end{bmatrix} A^T &= K, \end{aligned} \tag{2.46}$$

before reading off the solution from the components of K.

Throughout this section, we assume that F is a general matrix which allows for fast matrix-vector products and we look at computing $(A \otimes F)\, g$ for a vector g and particular matrix A.

For completeness, we recall some properties of the Kronecker product; one reference is [69]. These include bilinearity and associativity. Furthermore,

$$(A \otimes B)\,(C \otimes D) = AC \otimes BD. \tag{2.47}$$

Here, we are assuming that the matrix products AC and BD are well-defined. Additionally, there are permutation matrices (called perfect shuffle matrices) P and Q so that

$$P(A \otimes B)Q = B \otimes A. \tag{2.48}$$

2.3.1 Low-rank Matrices

We look at computing the product $(wv^* \otimes F)\, g$, where $w, v \in \mathbb{R}^m$ and $F \in \mathbb{R}^{n\times n}$. First, we see

$$\begin{aligned}
(wv^* \otimes F)\, g &= \begin{bmatrix} w_1v_1F & w_1v_2F & \cdots & w_1v_mF \\ w_2v_1F & w_2v_2F & \cdots & w_2v_mF \\ \vdots & \vdots & \cdots & \vdots \\ w_mv_1F & w_mv_2F & \cdots & w_mv_mF \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{bmatrix} \\
&= \begin{bmatrix} w_1\left(v_1Fg_1 + v_2Fg_2 + \cdots v_mFg_m\right) \\ w_2\left(v_1Fg_1 + v_2Fg_2 + \cdots v_mFg_m\right) \\ \vdots \\ w_m\left(v_1Fg_1 + v_2Fg_2 + \cdots v_mFg_m\right) \end{bmatrix} \\
&= w \otimes FGv, \qquad (2.49)
\end{aligned}$$

where, as before,

$$\begin{aligned}
g_k &= g\left([k-1]\,n+1\!:\!kn\right) \quad k \in \{1, \cdots, m\} \\
G &= \begin{bmatrix} g_1 & g_2 & \cdots & g_m \end{bmatrix}. \qquad (2.50)
\end{aligned}$$

Now, from the above computation, we see

$$\begin{aligned}
(WV^* \otimes F)\, g &= \left(\begin{bmatrix} w_1 & \cdots & w_r \end{bmatrix} \begin{bmatrix} v_1 & \cdots & v_r \end{bmatrix}^* \otimes F\right) g \\
&= (w_1v_1^* \otimes F)\, g + \cdots + (w_rv_r^* \otimes F)\, g \\
&= w_1 \otimes FGv_1 + \cdots + w_r \otimes FGv_r \qquad (2.51)
\end{aligned}$$

To efficiently compute this, we can use

$$\begin{aligned}
H &= FGV \\
&= \begin{bmatrix} FGv_1 & \cdots & FGv_r \end{bmatrix} \\
&= \begin{bmatrix} h_1 & \cdots & h_r \end{bmatrix} \qquad (2.52)
\end{aligned}$$

so that

$$(WV^* \otimes F)\, g = w_1 \otimes h_1 + \cdots + w_r \otimes h_r. \qquad (2.53)$$

Thus, we only need to compute fast multiplication with F once.

2.3.2 Householder Reflectors

From Sec. 2.3.1, we can easily compute $(H \otimes F)\, g$, where $H = I - 2uu^*$ and u is a unit vector. From properties of the Kronecker product, we see

$$\begin{aligned}(H \otimes F)\, g &= (I \otimes F)\, g - 2\,(vv^* \otimes F)\, g \\ &= (I \otimes F)\, g - 2\,(v \otimes FGv)\end{aligned} \tag{2.54}$$

We can quickly compute $(I \otimes F)\, g$ from looking at the columns of FG.

2.3.3 Givens Rotations

We assume our Givens rotation P has the form

$$P\left([i,j]\,,[i,j]\right) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}. \tag{2.55}$$

If $g_k = g\left([k-1]\,n + 1\!:\!kn\right)$ for $k \in \{1, \cdots, m\}$, then

$$(P \otimes F)\, g = \begin{bmatrix} Fg_1 \\ \vdots \\ cFg_i + sFg_j \\ \vdots \\ -sFg_i + cFg_j \\ \vdots \\ Fg_m \end{bmatrix}, \tag{2.56}$$

so that the kth block is Fg_k except for blocks i and j.

Frequently, we will have (possibly disjoint) products of Givens rotations. We obtain similar results when applied to these products.

Chapter 3

Properties of Chebyshev-Vandermonde Matrices

In this chapter we discuss properties of Chebyshev polynomials and Chebyshev-Vandermonde (C-V) matrices that we will use in later chapters to develop fast algorithms for solving interpolation and differential equations.

3.1 C-V Matrix Properties

Recalling our definitions of the Chebyshev polynomials T_k from Sec. 2.1.2, we define the infinite C-V matrix V_∞ by

$$[V_\infty]_{ij} = T_{j-1}\left(z_i^n\right), \quad i \in \{1, \cdots, n\} \quad j \in \mathbb{N}. \tag{3.1}$$

Any other C-V matrix V will contain a finite number of columns of V_∞. Furthermore, we let

$$C_{ij} = T_{j-1}\left(z_i^n\right), \quad i, j \in \{1, \cdots, n\}; \tag{3.2}$$

that is, C is just the first $n \times n$ block of V_∞. This C corresponds to our choice of DCT, discussed in Sec. 2.1.5. We show

$$
\begin{aligned}
C^*V_\infty &= F\begin{bmatrix} I & 0 & -\Lambda & 0 & \Lambda & 0 & -\Lambda & 0 & \cdots \end{bmatrix} \\
&= FW_\infty, \qquad (3.3)
\end{aligned}
$$

where I is the $n \times n$ identity matrix, 0 is a column of zeros,

$$
F = \frac{n}{2}\operatorname{diag}(2,1,1,\cdots,1), \qquad (3.4)
$$

and

$$
\Lambda = \begin{bmatrix}
 & & & 1 & & & \\
 & & 1 & & 1 & & \\
 & \iddots & & & & \ddots & \\
 1 & & & & & & 1 \\
1 & & & & & & & 1
\end{bmatrix}, \qquad (3.5)
$$

an $n \times 2n-1$ matrix. We see that the zero columns of W_∞ are located at n, $3n$, $5n$, $\ldots$, and that the first row of W has ± 1 located at column 0, $2n$, $4n$, $\ldots$. Explicitly, we have

$$
\begin{aligned}
[W_\infty]_{k,k+4(\ell-1)n} &= 1, \quad k \in \{1,\cdots,n\}, \quad \ell \in \mathbb{N} \\
[W_\infty]_{k,(4\ell-2)n+2-k} &= -1, \quad k \in \{2,\cdots,n\}, \quad \ell \in \mathbb{N} \\
[W_\infty]_{k,k+(4\ell-2)n} &= -1, \quad k \in \{1,\cdots,n\}, \quad \ell \in \mathbb{N} \\
[W_\infty]_{k,4\ell n+2-k} &= 1, \quad k \in \{2,\cdots,n\}, \quad \ell \in \mathbb{N}. \qquad (3.6)
\end{aligned}
$$

We prove this. First, we have

$$
\begin{aligned}
[C^*V_\infty]_{i+1,j+1} &= \sum_{k=1}^{n} T_i(z_k^n)\,T_j(z_k^n) \\
&= \sum_{k=1}^{n} \cos\left[\frac{i\pi}{n}\left(n-k+\frac{1}{2}\right)\right]\cos\left[\frac{j\pi}{n}\left(n-k+\frac{1}{2}\right)\right] \\
&= \frac{1}{2}\sum_{k=1}^{n}\left\{\cos\left[\frac{i+j}{n}\pi\left(n-k+\frac{1}{2}\right)\right]+\cos\left[\frac{i-j}{n}\pi\left(n-k+\frac{1}{2}\right)\right]\right\} \\
&= \frac{1}{2}\sum_{k=0}^{n-1}\left\{\cos\left[\frac{i+j}{2n}\pi(2k+1)\right]+\cos\left[\frac{i-j}{2n}\pi(2k+1)\right]\right\}, \qquad (3.7)
\end{aligned}
$$

where the reductions are standard product-to-sum trigonometric identities and reversing the sum. As we can see, we are finished if we can sum

$$\sum_{k=0}^{n-1} \cos\left[\frac{\alpha\pi}{2n}(2k+1)\right]. \tag{3.8}$$

If $\alpha \in \{0, \pm 4\pi, \pm 8\pi, \cdots\}$ the above sum is n, while if $\alpha \in \{\pm 2\pi, \pm 6\pi, \pm 10\pi, \cdots\}$, the sum is $-n$. We show if α is any other integer then the previous sum is 0. To see this, we complexify the situation. Letting

$$\begin{aligned} \xi &= \exp\left[i\frac{\alpha\pi}{n}\right] \\ \eta &= \exp\left[i\frac{\alpha\pi}{2n}\right], \end{aligned} \tag{3.9}$$

so that ξ is the primitive nth root of unity and η is the primitive $2n$th root of unity, we see

$$\cos\left[\frac{\alpha\pi}{2n}(2k+1)\right] = \Re\left(\xi^k \eta\right). \tag{3.10}$$

Therefore, it follows

$$\begin{aligned} \sum_{k=0}^{n-1} \xi^k \eta &= \eta \frac{1-\xi^n}{1-\xi} \\ &= \frac{1-(-1)^\alpha}{\eta^{-1} - \xi\eta^{-1}} \\ &= \frac{1}{2i} \frac{(-1)^\alpha - 1}{\sin\left(\frac{\alpha\pi}{2n}\right)}. \end{aligned} \tag{3.11}$$

This is purely imaginary when $\alpha \neq 2n\gamma$ for $\gamma \in \mathbb{Z}$, so we have proven

$$\sum_{k=0}^{n-1} \cos\left[\frac{\alpha\pi}{2n}(2k+1)\right] = \begin{cases} n & \alpha = 4n\gamma, \quad \gamma \in \mathbb{Z} \\ -n & \alpha = 4n\gamma + 2, \quad \gamma \in \mathbb{Z} \\ 0 & \text{otherwise} \end{cases}. \tag{3.12}$$

From here, we see when $i, j \in \{0, 1, \cdots, n-1\}$, we have

$$[C^*V]_{ij} = \begin{cases} n & i = j = 0 \\ \frac{n}{2} & i = j \neq 0 \\ 0 & \text{otherwise} \end{cases} . \tag{3.13}$$

Furthermore, we see with $i, j \in \{1, 2, \cdots, n-1\}$, we have

$$[C^*V_\infty]_{i,n+j} = \begin{cases} -\frac{n}{2} & i = n - j \\ 0 & \text{otherwise} \end{cases} . \tag{3.14}$$

The properties

$$\begin{aligned} T_{j+2n}\left(z_k^n\right) &= -T_j\left(z_k^n\right) \\ T_{j+4n}\left(z_k^n\right) &= T_j\left(z_k^n\right) \end{aligned} \tag{3.15}$$

follow from the definitions. Taken together, we have the desired result.

When we have $2n+1$ columns, we see

$$W = \begin{bmatrix} 1 & & & & & & -1 \\ & 1 & & & & -1 & \\ & & \ddots & & \iddots & & \\ & & 1 & 0 & -1 & & \end{bmatrix} . \tag{3.16}$$

3.2 Multiplication of Chebyshev Polynomials

To allow for variable coefficient differential equations, we will need to multiply functions in terms of the Chebyshev basis. Using standard trigonometric multiplication results, we see

$$\begin{aligned} T_n(\cos\theta)T_m(\cos\theta) &= \cos n\theta \cos m\theta \\ &= \frac{1}{2}\left\{\cos\left[(n+m)\theta\right] + \cos\left[(n-m)\theta\right]\right\}, \end{aligned} \tag{3.17}$$

from which it follows

$$T_m T_n = \frac{1}{2}\left(T_{m+n} + T_{|m-n|}\right). \tag{3.18}$$

For $m \geq 1$, we let

$$M_m = \begin{bmatrix} & & & & \frac{1}{2} & & & & & & \\ & & & \frac{1}{2} & & \frac{1}{2} & & & & & \\ & & \iddots & & & & \ddots & & & & \\ & \frac{1}{2} & & & & & & \frac{1}{2} & & & \\ 1 & & & & & & & & \frac{1}{2} & & \\ & \frac{1}{2} & & & & & & & & \frac{1}{2} & \\ & & \ddots & & & & & & & & \ddots \end{bmatrix}. \tag{3.19}$$

The primary bands are located on the mth subdiagonal and superdiagonal, so that 1 is in the $m+1$ row and the $\frac{1}{2}$ in the first row is in the $m+1$ column. Explicitly, for $m \neq 0$ we have

$$\begin{aligned} [M_m]_{m+1,1} &= 1 \\ [M_m]_{m+k,k} &= \frac{1}{2}, \quad k \in \{2, 3, \cdots\} \\ [M_m]_{k,m+k} &= \frac{1}{2}, \quad k \in \{2, 3, \cdots\} \\ [M_m]_{k,m+2-k} &= \frac{1}{2}, \quad k \in \{1, 2, \cdots, m\}. \end{aligned} \tag{3.20}$$

All other entries are 0. Therefore

$$\operatorname{diag}(T_m(z^n))V_\infty = V_\infty M_m, \tag{3.21}$$

where

$$T_m(z^n) = \begin{bmatrix} T_m(z_1^n) \\ T_m(z_2^n) \\ \vdots \\ T_m(z_n^n) \end{bmatrix}. \tag{3.22}$$

When V is finite, we chop M_m to the appropriate size.

3.3 Differentiation of Chebyshev Polynomials

We would eventually like to solve ordinary and partial differential equations (ODEs and PDEs), so we must work with derivatives of Chebyshev polynomials. We define the infinite Chebyshev-Vandermonde derivative matrix V'_∞ by

$$[V'_\infty]_{ij} = T'_{j-1}\left(z_i^n\right), \quad i \in \{1, \cdots, n\} \quad j \in \mathbb{N}. \tag{3.23}$$

Any other Chebyshev-Vandermonde matrix V' will contain a finite number of columns of V'_∞.

We prove the following theorem:

Theorem 3.1 (Derivative of Chebyshev Polynomials)
We have the following relationship between Chebyshev polynomials and their derivatives:

$$\begin{aligned} T'_{2n+1} &= 2\left(2n+1\right)\left(T_{2n} + T_{2n-2} + \cdots + T_2 + \frac{1}{2}T_0\right) \\ T'_{2n} &= 2\left(2n\right)\left(T_{2n-1} + T_{2n-3} + \cdots + T_1\right). \end{aligned} \tag{3.24}$$

Proof. We proceed by induction. First, from the recurrence relation in Eq. (2.8), we see

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x. \end{aligned} \tag{3.25}$$

Direct computation shows us

$$\begin{aligned} T'_0(x) &= 0 \\ &= 2(2 \cdot 0) \\ T'_1(x) &= 1 \end{aligned}$$

$$
\begin{aligned}
&= 2(2\cdot 0+1)\frac{1}{2}T_0(x)\\
T_2'(x) &= 4x\\
&= 2(2\cdot 1)T_1(x)\\
T_3'(x) &= 12x^2-3\\
&= 2(2\cdot 1+1)\left[T_2(x)+\frac{1}{2}T_0(x)\right]. \qquad (3.26)
\end{aligned}
$$

Thus, we have proven the base cases $n=0$ and $n=1$.

Assume $n \geq 2$ and that the relationship holds for for T'_{2k} and T'_{2k+1} for all $k < n$. From the recurrence relationship of Chebyshev polynomials and recognizing $T_1(x) = x$, we see

$$
\begin{aligned}
T'_{2n} &= 2T_{2n-1} + 2T_1T'_{2n-1} - T'_{2n-2}\\
&= 2T_{2n-1} + 2T_1\left\{2\left(2n-1\right)\left[T_{2n-2}+T_{2n-4}+\cdots+T_2+\frac{1}{2}T_0\right]\right\}\\
&\quad - 2\left(2n-2\right)\left[T_{2n-3}+T_{2n-5}+\cdots+T_1\right]\\
&= 2T_{2n-1} + 2\left(2n-1\right)\left[T_{2n-1}+2T_{2n-3}+\cdots+2T_3+2T_1\right]\\
&\quad - 2\left(2n-2\right)\left[T_{2n-3}+T_{2n-5}+\cdots+T_1\right]\\
&= 2\left(2n\right)\left[T_{2n-1}+T_{2n-3}+\cdots+T_1\right]. \qquad (3.27)
\end{aligned}
$$

Similarly, we have

$$
\begin{aligned}
T'_{2n+1} &= 2T_{2n} + 2T_1T'_{2n} - T'_{2n-1}\\
&= 2T_{2n} + 2T_1\left\{2(2n)\left[T_{2n-1}+T_{2n-3}+\cdots+T_1\right]\right\}\\
&\quad - 2\left(2n-1\right)\left[T_{2n-2}+T_{2n-4}+\cdots+T_2+\frac{1}{2}T_0\right]\\
&= 2T_{2n} + \left(2n\right)\left[T_{2n}+2T_{2n-2}+2T_{2n-4}+\cdots+2T_2+T_0\right]\\
&\quad - 2\left(2n-1\right)\left[T_{2n-2}+T_{2n-4}+\cdots+T_2+\frac{1}{2}T_0\right]\\
&= 2\left(2n+1\right)\left[T_{2n}+T_{2n-2}+\cdots+T_2+\frac{1}{2}T_0\right]. \qquad (3.28)
\end{aligned}
$$

This is the desired result. □

This allows us to write

$$V'_\infty = V_\infty D_\infty, \tag{3.29}$$

where

$$D_\infty = 2 \begin{bmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \cdots \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & \cdots \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \operatorname{diag}(0,1,2,3,4,\cdots). \tag{3.30}$$

Explicitly, we see

$$\begin{aligned} [D_\infty]_{1,2k} &= 2k-1, \quad k \in \mathbb{N} \\ [D_\infty]_{j,j-1+2k} &= 2(2k+j-2), \quad j \in \{2,3,\cdots\}, \quad k \in \mathbb{N}. \end{aligned} \tag{3.31}$$

All other entries are 0.

When we have $n = 9$ rows and $2n+1$ columns, we see

$$WD = \begin{bmatrix} 0 & 1 & & 3 & & 5 & & 7 & & 9 & & 11 & & 13 & & 15 & & 17 & 0 \\ & & 4 & & 8 & & 12 & & 16 & & 20 & & 24 & & 28 & & 32 & & \\ & & & 6 & & 10 & & 14 & & 18 & & 22 & & 26 & & 30 & & & \\ & & & & 8 & & 12 & & 16 & & 20 & & 24 & & 28 & & & & \\ & & & & & 10 & & 14 & & 18 & & 22 & & 26 & & & & & \\ & & & & & & 12 & & 16 & & 20 & & 24 & & & & & & \\ & & & & & & & 14 & & 18 & & 22 & & & & & & & \\ & & & & & & & & 16 & & 20 & & & & & & & & \\ & & & & & & & & & 18 & & & & & & & & & \end{bmatrix}. \tag{3.32}$$

This pattern holds in general. For a simpler structure, we let

$$U = \begin{bmatrix} 1 & & -1 & & & & & & \\ & 1 & & -1 & & & & & \\ & & 1 & & -1 & & & & \\ & & & 1 & & -1 & & & \\ & & & & \ddots & & \ddots & & \\ & & & & & 1 & & -1 & \\ & & & & & & 1 & & -1 \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{bmatrix} \operatorname{diag}\left(1, \frac{1}{2}, \cdots, \frac{1}{2}\right). \tag{3.33}$$

Then, for general n rows and $2n+1$ columns, we have

$$UWD = \begin{bmatrix} 0 & 1 & & & & & & & & & & 2n-1 & 0 \\ & & 2 & & & & & & & & 2n-2 & & \\ & & & \ddots & & & & & & \iddots & & & \\ & & & & n-2 & & & & n+2 & & & & \\ & & & & & n-1 & & n+1 & & & & & \\ & & & & & & n & & & & & & \end{bmatrix}. \tag{3.34}$$

3.4 C-V Matrix Normal Equations

We will now look at the normal equations involving the C-V matrix. If $A \in \mathbb{R}^{m \times n}$ with $n > m$ has linearly independent rows (full row-rank, so that A^* is injective), then we can solve

$$\min_{Ax=b} ||x||_2 \tag{3.35}$$

by computing the LQ factorization of A or using the pseudoinverse of A, A^+; one reference for the properties of the matrix pseudoinverse is [27, Chapter 11]. Because we are assuming that A has full row-rank, we see

$$A^+ = A^* \left(AA^*\right)^{-1}. \tag{3.36}$$

Thus, in our case, we want to look at the matrix VD_s^{-1}:

$$\left(VD_s^{-1}\right)^+ = D_s^{-1}V^*\left(VD_s^{-2}V^*\right)^{-1}. \tag{3.37}$$

If $VD_s^{-2}V^*$ can be inverted quickly, then this will give us a fast algorithm for inversion.

First, we assume that V has N columns and compute the components of $VD_s^{-2}V^*$:

$$\begin{aligned}
\left[VD_s^{-2}V^*\right]_{k,\ell} &= \sum_{m=1}^{N}\left[VD_s^{-1}\right]_{k,m}\left[VD_s^{-1}\right]_{\ell,m} \\
&= \sum_{m=1}^{N}\frac{T_{m-1}\left(z_k^n\right)T_{m-1}\left(z_\ell^n\right)}{m^{2s}} \\
&= \frac{1}{2}\left\{\sum_{m=1}^{N}\frac{\cos\left[\frac{(m-1)\pi}{n}\left(2n-[k+\ell]+1\right)\right]}{m^{2s}} + \sum_{m=1}^{N}\frac{\cos\left[\frac{(m-1)\pi}{n}\left(k-\ell\right)\right]}{m^{2s}}\right\} \\
&= \frac{1}{2}\sum_{m=1}^{N}\frac{\cos\left[(m-1)\,\xi_{k+\ell}\right]}{m^{2s}} + \frac{1}{2}\sum_{m=1}^{N}\frac{\cos\left[(m-1)\,\eta_{k-\ell}\right]}{m^{2s}} \\
&= P(\xi_{k+\ell}^n) + Q(\eta_{k-\ell}^n).
\end{aligned} \tag{3.38}$$

Naturally, we have

$$\begin{aligned}
\xi_j^n &= \frac{\pi}{n}\left(2n-j+1\right) \\
\eta_j^n &= \frac{\pi}{n} \\
P(\xi) &= \frac{1}{2}\sum_{m=1}^{N}\frac{\cos\left[(m-1)\,\xi\right]}{m^{2s}} \\
Q(\eta) &= \frac{1}{2}\sum_{m=1}^{N}\frac{\cos\left[(m-1)\,\eta\right]}{m^{2s}}.
\end{aligned} \tag{3.39}$$

When $k+\ell$ is constant (on the antidiagonals), then we see that $\xi_{k+\ell}^n$ and $P(\xi_{k+\ell}^n)$ are constant, while when $k-\ell$ is constant (on the diagonals), then we see $\eta_{k-\ell}^n$ and $Q(\eta_{k-\ell}^n)$ are constant. This implies that

$$VD_s^{-2}V^* = H + T, \tag{3.40}$$

where H is a Hankel matrix and T is a Toeplitz matrix such that

$$\begin{aligned}
H &= \begin{bmatrix} P\left(\xi_2^n\right) & P\left(\xi_3^n\right) & P\left(\xi_4^n\right) & \iddots & P\left(\xi_{n+1}^n\right) \\ P\left(\xi_3^n\right) & P\left(\xi_4^n\right) & P\left(\xi_5^n\right) & \iddots & P\left(\xi_{n+2}^n\right) \\ P\left(\xi_4^n\right) & P\left(\xi_5^n\right) & P\left(\xi_6^n\right) & \iddots & P\left(\xi_{n+3}^n\right) \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ P\left(\xi_{n+1}^n\right) & P\left(\xi_{n+2}^n\right) & P\left(\xi_{n+3}^n\right) & \cdots & P\left(\xi_{2n}^n\right) \end{bmatrix} \\
T &= \begin{bmatrix} Q\left(\eta_0^n\right) & Q\left(\eta_1^n\right) & Q\left(\eta_2^n\right) & \cdots & Q\left(\eta_{n-1}^n\right) \\ Q\left(\eta_1^n\right) & Q\left(\eta_0^n\right) & Q\left(\eta_1^n\right) & \ddots & Q\left(\eta_{n-2}^n\right) \\ Q\left(\eta_2^n\right) & Q\left(\eta_1^n\right) & Q\left(\eta_0^n\right) & \ddots & Q\left(\eta_{n-3}^n\right) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ Q\left(\eta_{n-1}^n\right) & Q\left(\eta_{n-2}^n\right) & Q\left(\eta_{n-3}^n\right) & \cdots & Q\left(\eta_0^n\right) \end{bmatrix}.
\end{aligned} \tag{3.41}$$

We will only briefly pursue these matters further when proving convergence of interpolation using polynomials up to degree $2Ln$.

For MSN interpolation on a general grid, the normal equations will be structured in a similar way and could be inverted quickly. This will be related to the Clausen function

$$C_s(\theta) = \sum_{k=1}^{\infty} \frac{\cos k\theta}{k^s} \tag{3.42}$$

and the polylogarithm

$$\mathrm{Li}_s(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^s}. \tag{3.43}$$

$C_s(\theta)$ is a polynomial in θ for even integer values of s [1, Page 1005 (27.8.6)]. The resulting matrices will have off-diagonal blocks of low rank and could be inverted quickly by converting them to Sequentially Semi-Separable (SSS) [21] or Hierarchically Semi-Separable (HSS) [15] form.

3.5 Linear Combinations of C-V Matrices

In this section, we look at matrices which arise from combining differentiation and multiplication by Chebyshev polynomials. This arises when solving differential equations in Chapter 8.

Due to the better structure which arose in Sec. 3.3, we left-multiply all of our C-V matrices with U. Thus, when W has $2n+1$ columns we have

$$2UW = \begin{bmatrix} 2 & & - & & & & & & & & & & + & & -2 \\ & + & & - & & & & & & & & + & & - & \\ & & + & & - & & & & & & + & & - & & \\ & & & & \ddots & & & & & & \iddots & & & & \\ & & & & + & & - & 0 & + & & - & & & & \\ & & & & & + & & 0 & & - & & & & & \\ & & & & & & + & 0 & - & & & & & & \end{bmatrix}, \qquad (3.44)$$

where "+" stands for $+1$ and "−" stands for -1. It is on the UW matrix in Eq. (3.44) and UWD matrix in Eq. (3.34) that we multiply by M_m. Because $M_0 = I$, we do not need to worry about that case.

From the examples below, it is clear we can easily precompute the exact form needed for any of the necessary linear combinations that we will need to take.

3.5.1 Multiplication and Derivative C-V Matrices

Because UWD has only 1 "diagonal", we start with looking at the structure of $UWDM_m$ as it is simpler. In Eq. (3.45), we see the $UWDM_m(:, 1 : n+1)$, essentially the first half of the matrix. The explicit 0 in the first row is located in column $m+1$, while the $2m$ in the first column is located in row m. In Eq. (3.45), we see the $UWDM_m(:, n+1 : 2n+1)$, the second half of the matrix. The $2n$ in the first column (column $n+1$ of the actual matrix) occurs in row $n-m$, while the n in the last row occurs in column $n+m+1$.

$$
(2UWDM_m)(:,1:n+1) =
\left[\begin{array}{ccccccccccccccc}
 & & & & 1 & 0 & 1 & & & & & & & & \\
 & & & 2 & & & & 2 & & & & & & & \\
 & & \iddots & & & & & & \ddots & & & & & & \\
 & m-1 & & & & & & & & m-1 & & & & & \\
2m & & & & & & & & & & m & & & & \\
 & m+1 & & & & & & & & & & m+1 & & & \\
 & & \ddots & & & & & & & & & & \ddots & & \\
 & & & n-k-1 & & & & & & & & & & n-k-1 & \\
 & & & & n-k & & & & & & & & & & 2n \\
 & & & & & n-k+1 & & & & & & & & n+k-1 & \\
 & & & & & & \ddots & & & & & & \iddots & & \\
 & & & & & & & n-2 & & & & n+2 & & & \\
 & & & & & & & & n-1 & & n+1 & & & & \\
 & & & & & & & & & n & & & & &
\end{array}\right]
\tag{3.45}
$$

$$
(2UWDM_m)(:,n+1,2n+1) =
\left[\begin{array}{ccccccccccccc}
 & & & & & & & 2n-1 & & & & & \\
 & & & & & & \iddots & & & & & & \\
 & & & & & 2n-m+1 & & & & & & & \\
 & & & & 2n-m & & & & & & & & 2n-m \\
 & & & 2n-m-1 & & & & & & & & 2n-m-1 & \\
 & & \iddots & & & & & & & & \iddots & & \\
 & n+m+1 & & & & & & & & n+m+1 & & & \\
2n & & & & & & & & n+m & & & & \\
 & n-m+1 & & & & & & n+m-1 & & & & & \\
 & & \ddots & & & & \iddots & & & & & & \\
 & & & n-1 & & n+1 & & & & & & & \\
 & & & & n & & & & & & & &
\end{array}\right]
\qquad (3.46)
$$

3.5.2 Multiplication and Interpolation C-V Matrices

We now focus on computing UWM_m, which is more difficult because UW has two bands. We compute the first two matrices to understand the explicit structure. As above, "+" stands for $+1$ and "$-$" stands for -1.

$$4UWM_1 = \left[\begin{array}{ccccccccccccccccccc}
 & + & & - & & & & & & & & & & & & + & & - & \\
2 & & & & - & & & & & & & & & & + & & & & - \\
 & + & & & & & & & & & & & & & & & & - & \\
 & & + & & & & - & & & & & & + & & & & - & & \\
 & & & + & & \ddots & & - & & & & + & & \iddots & & - & & & \\
 & & & & + & & & & - & & + & & & & - & & & & \\
 & & & & & & & & & 0 & & & & & & & & & \\
 & & & & & & + & & + & 0 & - & & - & & & & & & \\
 & & & & & & & + & & 0 & & - & & & & & & &
\end{array}\right] \tag{3.47}$$

$$4UWM_2 = \left[\begin{array}{ccccccccccccccccccc}
-2 & & 2 & & - & & & & & & & & & & + & & -2 & & + \\
 & & & + & & & & & & & & & & & & - & & + & \\
2 & & - & & + & & - & & & & & & + & & - & & + & & - \\
 & + & & - & & & & - & & & & + & & & & + & & - & \\
 & & + & & - & \ddots & + & & - & & + & & - & \iddots & + & & - & & \\
 & & & + & & & & + & & 0 & & - & & & & - & & & \\
 & & & & + & & - & & 2 & 0 & -2 & & + & & - & & & & \\
 & & & & & & & & & 0 & & & & & & & & & \\
 & & & & & & + & & - & 0 & + & & - & & & & & &
\end{array}\right] \tag{3.48}$$

Chapter 4

C-V Matrices and Factorizations for 1D Interpolation

Using structured matrices from Chapter 2, we investigate Chebyshev-Vandermonde matrices in 1D and use this information to construct fast algorithms for interpolation problems. These factorizations will allow us to quickly compute the MSN solution to interpolation and differential equations (discussed later in Chapter 8).

4.1 General Algorithm for MSN Interpolation using LQ factorization

The algorithm for computing the MSN interpolation solution is easy once we have the LQ factorization; see Alg. 4. We will only have fast solutions of this form when we can quickly invert L and multiply by Q. Interpolation on Chebyshev nodes and expressing the result in Chebyshev polynomials gives L and Q the necessary structure.

Algorithm 4 Fast algorithm for solving structured MSN systems

```
function FAST_LQ_SOLVE($f$,$V$,$D_s$) ▷ Solve $\min_{Va=f} ||D_s a||_2$ where $C^{-1}VD_s^{-1} = LQ$.
    Set $\hat{f} = C^{-1}f$
    Compute $LQ = C^{-1}VD_s^{-1}$
    Solve $Ly = \hat{f}$
    Compute $x = Q^* y$
    Set $a = D_s^{-1}x$
    return $a$
end function
```

4.2 1D C-V Interpolation Matrix: $2n+1$ Columns

We take V to be our C-V matrix with $2n+1$ columns. Recalling that we need to compute the minimum norm solution of

$$VD_s^{-1}x = f. \tag{4.1}$$

D_s is a diagonal scaling matrix which we use to bound the sth derivative; it was defined in Eq. (2.30). From here, we see

$$\begin{aligned} C^{-1}VD_s^{-1} &= WD_s^{-1} \\ &= \begin{bmatrix} 1^{-s} & & & & & & & & -(2n+1)^{-s} \\ & 2^{-s} & & & & & & -(2n)^{-s} & \\ & & \ddots & & & & \iddots & & \\ & & & n^{-s} & 0 & -(n+2)^{-s} & & & \end{bmatrix}. \end{aligned} \tag{4.2}$$

We need to convert this matrix into a lower triangular matrix via rotations, and we use Givens rotations to zero the nonzero components. Specifically, we require the Givens rotation G_k to have

$$G_k\left([k, 2n+2-k], [k, 2n+2-k]\right) = \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix}. \tag{4.3}$$

In practice, we can compute

$$\begin{aligned}\tau_k &= \left(\frac{k}{2n+2-k}\right)^s \\ c_k &= \frac{1}{\sqrt{1+\tau_k^2}} \\ s_k &= \tau_k c_k \end{aligned} \tag{4.4}$$

stably using Alg. 3. All of these Givens rotations are disjoint and we set $G = G_1 G_2 \cdots G_n$, with

$$G = \begin{bmatrix} c_1 & & & & & & & & s_1 \\ & c_2 & & & & & & s_2 & \\ & & \ddots & & & & \iddots & & \\ & & & c_n & & s_n & & & \\ & & & & 1 & & & & \\ & & & -s_n & & c_n & & & \\ & & \iddots & & & & \ddots & & \\ & -s_2 & & & & & & c_2 & \\ -s_1 & & & & & & & & c_1 \end{bmatrix}. \tag{4.5}$$

Then, we find

$$WD_s^{-1}G = \begin{bmatrix} Y & 0 \end{bmatrix}, \tag{4.6}$$

where

$$\begin{aligned} Y &= \operatorname{diag}(y_1, \cdots, y_n) \\ y_k &= k^{-s}\sqrt{1+\left(\frac{k}{2n+2-k}\right)^{2s}} \\ &= \frac{1}{c_k k^s}. \end{aligned} \tag{4.7}$$

This is the desired LQ factorization, and we can now use Alg. 4 to solve this system quickly. As we can see, we are making small changes to the DCT coefficients.

4.3 Endpoint Interpolation

While most of the factorizations involve the Chebyshev nodes, we will at times include the endpoints ±1. If we want to interpolate function values at $f(-1) = f_0$ and $f(1) = f_{n+1}$, then the coefficients a must satisfy

$$\begin{aligned} Aa &= \begin{bmatrix} 1 & -1 & 1 & -1 & \cdots & -1 & 1 \\ 1 & 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{2n-1} \\ a_{2n} \end{bmatrix} \\ &= \begin{bmatrix} f_0 \\ f_{n+1} \end{bmatrix}. \end{aligned} \tag{4.8}$$

We define

$$\begin{aligned} C_0 &= \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \\ C_1 &= \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}. \end{aligned} \tag{4.9}$$

With these definitions, we see

$$\begin{aligned} C_0^{-1}A &= \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots & 0 & 1 \\ 0 & 1 & 0 & 1 & \cdots & 1 & 0 \end{bmatrix} \\ C_1^{-1}A &= \begin{bmatrix} 0 & 1 & 0 & 1 & \cdots & 1 & 0 \\ 1 & 0 & 1 & 0 & \cdots & 0 & 1 \end{bmatrix}. \end{aligned} \tag{4.10}$$

We will prefer one of these matrices (C_0 or C_1) over the other depending on the context. The distinction depends on whether n is even or odd.

4.4 1D C-V Interpolation Matrix: $2n + 1$ Columns with Endpoint Interpolation

In this section we look at a slight variation of interpolation on the Chebyshev nodes: we include endpoint interpolation from the previous section. This changes our C-V matrix slightly, for we have some additional linear requirements. In light of Sec. 4.3 and Eqs. (4.8) and (4.9), we list our additional requirements as

$$Aa = \begin{bmatrix} f_0 \\ f_{n+1} \end{bmatrix}, \tag{4.11}$$

where $f_0 = f(-1)$ and $f_{n+1} = f(1)$. We set

$$\overline{A} = C_0^{-1} A. \tag{4.12}$$

Using the previous computations for G, we find

$$\begin{bmatrix} W \\ \overline{A} \end{bmatrix} D_s^{-1} G =$$

$$\begin{bmatrix}
y_1 & & & & & & & & & & & & & \\
 & y_2 & & & & & & & & & & & & \\
 & & y_3 & & & & & & & & & & & \\
 & & & y_4 & & & & & & & & & & \\
 & & & & \ddots & & & & & & & & & \\
 & & & & & y_{n-1} & & & & & & & & \\
 & & & & & & y_n & & & & & & & \\
\alpha_1 & & \alpha_3 & & \cdots & \alpha_{n-1} & & \alpha_{n+1} & & \alpha_{n+3} & & \cdots & & \alpha_{2n+1} \\
 & \alpha_2 & & \alpha_4 & \cdots & & \alpha_n & & \alpha_{n+2} & & \alpha_{n+4} & \cdots & \alpha_{2n} &
\end{bmatrix}. \tag{4.13}$$

From Eq. (4.13), we see that we almost have the LQ factorization. We only need two Householder reflectors to convert this to lower triangular form, and it is convenient that the vectors are orthogonal. We let

$$v_1^* = \begin{bmatrix} \alpha_{n+1} & 0 & \alpha_{n+3} & 0 & \cdots & 0 & \alpha_{2n+1} \end{bmatrix}$$

$$v_2^* = \begin{bmatrix} 0 & \alpha_{n+2} & 0 & \alpha_{n+4} & \cdots & \alpha_{2n} & 0 \end{bmatrix} \tag{4.14}$$

and set

$$\begin{aligned} u_1 &= \frac{1}{\left|\left|v_1 - \left|\left|v_1\right|\right|_2 e_1\right|\right|_2} \left(v_1 - \left|\left|v_1\right|\right|_2 e_1\right) \\ u_2 &= \frac{1}{\left|\left|v_2 - \left|\left|v_2\right|\right|_2 e_2\right|\right|_2} \left(v_2 - \left|\left|v_2\right|\right|_2 e_2\right). \end{aligned} \tag{4.15}$$

We remember that care must be taken when computing $v_i - \left|\left|v_i\right|\right|_2 e_i$ to avoid large numerical errors; see Alg. 2.

With

$$\begin{aligned} \widehat{P}_1 &= I_{n+1} - 2u_1u_1^* \\ \widehat{P}_2 &= I_{n+1} - 2u_2u_2^* \\ \widehat{P} &= \widehat{P}_1\widehat{P}_2 \\ &= I_{n+1} - 2\begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} u_1 & u_2 \end{bmatrix}^* \\ P &= \begin{bmatrix} I_n & \\ & \widehat{P} \end{bmatrix}, \end{aligned} \tag{4.16}$$

we have

$$\begin{bmatrix} W \\ \hline A \end{bmatrix} D_s^{-1} G P = \begin{bmatrix} y_1 &&&&&&&& \\ & y_2 &&&&&&& \\ && y_3 &&&&&& \\ &&& y_4 &&&&& \\ &&&& \ddots &&&& \\ &&&&& y_{n-1} &&& \\ &&&&&& y_n && \\ \alpha_1 && \alpha_3 && \cdots & \alpha_{n-1} && \left|\left|v_1\right|\right|_2 & \\ & \alpha_2 && \alpha_4 & \cdots && \alpha_n && \left|\left|v_2\right|\right|_2 \end{bmatrix}, \tag{4.17}$$

where we have ignored the zero entries of the matrix. This lower triangular matrix can be quickly inverted to compute the minimum norm solution, from which we then obtain the MSN solution after computing fast Householder reflectors and Givens rotations.

All of the above work was in the case when n was even. We now look at the odd case. For this, we set

$$\overline{A} = C_1^{-1}A \tag{4.18}$$

and find

$$\begin{bmatrix} W \\ \hline \overline{A} \end{bmatrix} D_s^{-1}G =$$
$$\begin{bmatrix}
y_1 & & & & & & & & & & & & & \\
& y_2 & & & & & & & & & & & & \\
& & y_3 & & & & & & & & & & & \\
& & & y_4 & & & & & & & & & & \\
& & & & \ddots & & & & & & & & & \\
& & & & & y_{n-1} & & & & & & & & \\
& & & & & & y_n & & & & & & & \\
& \alpha_2 & & \alpha_4 & \cdots & \alpha_{n-1} & & \alpha_{n+1} & & \alpha_{n+3} & & \cdots & \alpha_{2n} & \\
\alpha_1 & & \alpha_3 & & \cdots & & \alpha_n & & \alpha_{n+2} & & \alpha_{n+4} & \cdots & & \alpha_{2n+1}
\end{bmatrix}. \tag{4.19}$$

After computing the correct Householder reflectors, we will have the LQ factorization.

4.5 1D C-V Interpolation Matrix: $3n + 1$ Columns

If we use $3n + 1$ Chebyshev polynomials for interpolation (up to degree $3n$), then we must compute the LQ factorization of the following matrix:

$$WD_s^{-1} =$$
$$\begin{bmatrix}
1^{-s} & & & & & & -(2n+1)^{-s} & & & \\
& \ddots & & & & \iddots & & \ddots & & \\
& & n^{-2} & 0 & -(n+2)^{-s} & & & & -(3n)^{-s} & 0
\end{bmatrix}. \tag{4.20}$$

By using disjoint Givens rotations, the correct rotation matrix is

$$G = G_1G_2, \tag{4.21}$$

where

$$G_1 = \begin{bmatrix} c_1 & & & & & & & & s_1 & & & \\ & c_2 & & & & & & s_2 & & & & \\ & & \ddots & & & & \iddots & & & & & \\ & & & c_n & & s_n & & & & & & \\ & & & & 1 & & & & & & & \\ & & & -s_n & & c_n & & & & & & \\ & & \iddots & & & & \ddots & & & & & \\ & -s_2 & & & & & & c_2 & & & & \\ -s_1 & & & & & & & & c_1 & & & \\ & & & & & & & & & 1 & & \\ & & & & & & & & & & \ddots & \\ & & & & & & & & & & & 1 \end{bmatrix} \tag{4.22}$$

$$G_2 = \begin{bmatrix} 1 & & & & 0 & & 0 & & & & \\ & \chi_2 & & & & & & \sigma_2 & & & \\ & & \ddots & & & & & & \ddots & & \\ & & & \chi_n & & & & & & \chi_n & \\ 0 & & & & 1 & & & & & & 0 \\ & & & & & \ddots & & & & & \\ 0 & & & & & & 1 & & & & 0 \\ & -\sigma_2 & & & & & & \chi_2 & & & \\ & & \ddots & & & & & & \ddots & & \\ & & & -\sigma_n & & & & & & \chi_n & \\ & & & & 0 & & 0 & & & & 1 \end{bmatrix}. \tag{4.23}$$

Here, we define

$$\begin{aligned} \tau_k &= \left(\frac{k}{2n+2-k}\right)^s \\ c_k &= \frac{1}{\sqrt{1+\tau_k^2}} \\ s_k &= \tau_k c_k \\ \xi_k &= \left(\frac{k}{2n+k}\right)^s c_k \\ \mu_k &= \frac{1}{\sqrt{1+\xi_k^2}} \\ \nu_k &= \xi_k \mu_k. \end{aligned} \tag{4.24}$$

In this case, we see

$$WD_s^{-1}G = \begin{bmatrix} Y & 0 \end{bmatrix}, \tag{4.25}$$

where Y is diagonal with entries

$$\begin{aligned} y_1 &= \sqrt{1^{-2s} + (2n+1)^{-2s}} \\ &= \sqrt{1 + \left(\frac{1}{2n+1}\right)^{2s}} \\ y_k &= \sqrt{k^{-2s} + (2n+2-k)^{-2s} + (2n+k)^{-2s}} \\ &= k^{-s}\sqrt{1 + \left(\frac{k}{2n+2-k}\right)^{2s}\left[1 + \left(\frac{2n+2-k}{2n+k}\right)^{2s}\right]} \qquad k \in \{2, \cdots, n\}. \end{aligned} \tag{4.26}$$

This is a numerically stable was to compute the entries.

We can include endpoints in our interpolation by adding two Householder reflectors like we did in Sec. 4.2; we omit the details.

4.6 1D C-V Interpolation Matrix: $4n+1$ Columns

If we use $4n+1$ Chebyshev polynomials for interpolation (up to order $4n$), then we must compute the LQ factorization of the following matrix:

$$\begin{aligned} &\left[WD_s^{-1}\right](:, 1 : 2n+1) = \\ &\begin{bmatrix} 1^{-s} & & & & & & & & -(2n+1)^{-s} \\ & 2^{-s} & & & & & & -(2n)^{-s} & \\ & & \ddots & & & & \iddots & & \\ & & & n^{-s} & 0 & -(n+2)^{-s} & & & \end{bmatrix} \\ &\left[WD_s^{-1}\right](:, 2n+1 : 4n+1) = \\ &\begin{bmatrix} -(2n+1)^{-s} & & & & & & & & (4n+1)^{-s} \\ & -(2n+2)^{-s} & & & & & & (4n)^{-s} & \\ & & \ddots & & & & \iddots & & \\ & & & -(3n)^{-s} & 0 & (3n+2)^{-s} & & & \end{bmatrix}. \end{aligned} \tag{4.27}$$

By using disjoint Givens rotations, we let rotation matrix is

$$G = G_1 G_2, \tag{4.28}$$

where

$$G_1 = \begin{bmatrix}
c_1 & & & & & & & & s_1 & & & & & & & & \\
& c_2 & & & & & & s_2 & & & & & & & & & \\
& & \ddots & & & & \iddots & & & & & & & & & & \\
& & & c_n & & s_n & & & & & & & & & & & \\
& & & & 1 & & & & & & & & & & & & \\
& & & -s_n & & c_n & & & & & & & & & & & \\
& & \iddots & & & & \ddots & & & & & & & & & & \\
& -s_2 & & & & & & c_2 & & & & & & & & & \\
-s_1 & & & & & & & & c_1 & & & & & & & & 0 \\
& & & & & & & & & \mu_2 & & & & & & \nu_2 & \\
& & & & & & & & & & \ddots & & & & \iddots & & \\
& & & & & & & & & & & \mu_n & & \nu_n & & & \\
& & & & & & & & & & & & 1 & & & & \\
& & & & & & & & & & & -\nu_n & & \mu_n & & & \\
& & & & & & & & & & \iddots & & & & \ddots & & \\
& & & & & & & & & -\nu_2 & & & & & & \mu_2 & \\
& & & & & & & & 0 & & & & & & & & 1
\end{bmatrix} \tag{4.29}$$

$$G_2 = \begin{bmatrix}
\chi_1 & & & & & & 0 & & & & & & & \sigma_1 \\
& \chi_2 & & & & & & \sigma_2 & & & & & & \\
& & \ddots & & & & & & \ddots & & & & & \\
& & & \chi_n & & & & & & \sigma_n & & & & \\
& & & & 1 & & & & & & 0 & & & \\
& & & & & \ddots & & & & & & & & \\
0 & & & & & & 1 & & & & & & & 0 \\
& -\sigma_2 & & & & & & \chi_2 & & & & & & \\
& & \ddots & & & & & & \ddots & & & & & \\
& & & -\sigma_n & & & & & & -\chi_n & & & & \\
& & & & 0 & & & & & & 1 & & & \\
& & & & & & & & & & & \ddots & & \\
& & & & & & & & & & & & 1 & \\
-\sigma_1 & & & & & & 0 & & & & & & & \chi_1
\end{bmatrix}. \tag{4.30}$$

Here, we define

$$\begin{aligned}
\tau_k &= \left(\frac{k}{2n+2-k}\right)^s \\
c_k &= \frac{1}{\sqrt{1+\tau_k^2}} \\
s_k &= \tau_k c_k \\
\xi_k &= \left(\frac{2n+k}{4n+2-k}\right)^s \\
\mu_k &= -\frac{1}{\sqrt{1+\xi_k^2}} \\
\nu_k &= \xi_k \mu_k \\
\zeta_1 &= -\left(\frac{1}{4n+1}\right)^s c_1 \\
\zeta_k &= \left(\frac{k}{2n+k}\right)^s \frac{c_k}{\mu_k} \quad k \in \{2, \cdots, n\} \\
\chi_k &= \frac{1}{\sqrt{1+\zeta_k^2}} \\
\sigma_k &= \zeta_k \chi_k.
\end{aligned} \tag{4.31}$$

For $k \in \{2, \cdots, n\}$, we note that $\zeta_k < 0$ because $\mu_k < 0$. In this case, we see

$$WD_s^{-1}G = \begin{bmatrix} Y & 0 \end{bmatrix}, \tag{4.32}$$

where Y is diagonal with entries

$$\begin{aligned} y_1 &= \sqrt{1^{-2s} + (2n+1)^{-2s} + (4n+1)^{-2s}} \\ &= \sqrt{1 + \left(\frac{1}{2n+1}\right)^{2s} \left[1 + \left(\frac{2n+1}{4n+1}\right)^{2s}\right]} \\ y_k &= \sqrt{k^{-2s} + (2n+2-k)^{-2s} + (2n+k)^{-2s} + (4n+2-k)^{-2s}} \\ &= k^{-s}\sqrt{1 + \left(\frac{k}{2n+2-k}\right)^{2s} \left\{1 + \left(\frac{2n+2-k}{2n+k}\right)^{2s} \left[1 + \left(\frac{2n+k}{4n+2-k}\right)^{2s}\right]\right\}} \\ &\quad k \in \{2, \cdots, n\}. \end{aligned} \tag{4.33}$$

This is a numerically stable was to compute the entries.

4.7 1D C-V Interpolation Matrix: $2Ln + 1$ Columns

The previous interpolation patterns showed us a standard way to compute the pseudoinverse: compute the necessary LQ factorization using structured rotations. This is always possible but becomes difficult as the interpolation degree becomes larger; previously, we have always computed the full Q matrix, but the Q factor became more involved when using $3n + 1$ and $4n + 1$ columns. If we wish to have a method that works for MSN interpolation up to degree $2Ln$, then we will need to use something else: the normal equations. While it is well-known the normal equations can increase the difficulty of solving linear systems because it results in a squaring of the condition number [32, 41], this is not an issue here because the system is well structured and we are computing the factorization exactly (by hand).

After performing an IDCT, we see

$$\left(WD_s^{-1}\right)^+ = D_s^{-1}W^* \left(WD_s^{-2}W^*\right)^{-1}, \tag{4.34}$$

where we are assuming W has $2Ln+1$ columns. W has the following form:

$$\begin{aligned}
W &= \begin{bmatrix} I & 0 & -\Lambda & 0 & \Lambda & \cdots & 0 & E \end{bmatrix}, \quad L \mod 4 = 0 \\
W &= \begin{bmatrix} I & 0 & -\Lambda & 0 & \Lambda & \cdots & 0 & -E \end{bmatrix}, \quad L \mod 4 = 2.
\end{aligned} \tag{4.35}$$

We see

$$WD_s^{-2}W^* = Y^2, \tag{4.36}$$

where Y is a diagonal matrix with

$$\begin{aligned}
y_1^2 &= 1^{-2s} + [2n+1]^{-2s} + [4n+1]^{-2s} + \cdots + [2Ln+1]^{-2s} \\
y_k^2 &= k^{-2s} + [2n+2-k]^{-2s} + [2n+k]^{-2s} + [4n+2-k]^{-2s} + [4n+k]^{-2s} \\
&\quad + \cdots + [2(L-1)n+2-k]^{-2s} + [2(L-1)n+k]^{-2s} + [2Ln+2-k]^{-2s} \\
&\quad k \in \{2, \cdots, n\}.
\end{aligned} \tag{4.37}$$

If there is concern about roundoff error in these calculations, the values could be first sorted into increasing order before being added or compensated summation could be used; see [41, Chapter 4]. In this case, we see

$$\left(WD_s^{-1}\right)^+ = D_s^{-1}W^*Y^{-2}, \tag{4.38}$$

and all of these operations can be performed quickly. The solution coefficients are given by

$$a = D_s^{-2}W^*Y^{-2}\hat{f}. \tag{4.39}$$

4.8 1D C-V Derivative Matrix: $2n+1$ Columns; First Factorization

We now look at computing a factorization for $V' = VD$. From our work in Sec. 3.3, we know that UWD is highly structured and similar to W in that it only has one "diagonal". If we include the diagonal scaling D_s^{-1}, we see

$$UWDD_s^{-1} = \begin{bmatrix} 0 & \frac{1}{2^s} & & & & & \frac{2n-1}{(2n)^s} & 0 \\ & & \ddots & & & \iddots & & \\ & & & \frac{n-1}{n^s} & & \frac{n+1}{(n+2)^s} & & \\ & & & & \frac{n}{(n+1)^s} & & & \end{bmatrix}, \tag{4.40}$$

We now want zero out the right half of the $UWDD_s^{-1}$. We let

$$\begin{aligned} \tau_k &= -\left(\frac{2n+1-k}{k-1}\right)\left(\frac{k}{2n+2-k}\right)^s \qquad k \in \{2, \cdots, n\} \\ c_k &= \frac{1}{\sqrt{1+\tau_k^2}} \\ s_k &= \tau_k c_k. \end{aligned} \tag{4.41}$$

Using these c and s values, our full set of Givens rotations is

$$G = \begin{bmatrix} 1 & & & & & & & & \\ & c_2 & & & & & & s_2 & \\ & & \ddots & & & & \iddots & & \\ & & & c_n & & s_n & & & \\ & & & & 1 & & & & \\ & & & -s_n & & c_n & & & \\ & & \iddots & & & & \ddots & & \\ & -s_2 & & & & & & c_2 & \\ & & & & & & & & 1 \end{bmatrix}, \tag{4.42}$$

and we set

$$\gamma_k = (k-1)\,k^{-s}\sqrt{1+\left(\frac{2n+1-k}{k-1}\right)^2\left(\frac{k}{2n+2-k}\right)^{2s}} \qquad k \in \{2, \cdots, n\}$$

$$\gamma_{n+1} = n\left(n+1\right)^{-s}. \tag{4.43}$$

Using all of the above information, we see

$$\begin{aligned} UWDD_s^{-1}G &= \begin{bmatrix}0 & \Gamma & 0\end{bmatrix} \\ \Gamma &= \operatorname{diag}\left(\gamma_2, \gamma_3 \cdots, \gamma_{n+1}\right). \end{aligned} \tag{4.44}$$

If Π is the circular downshift permutation matrix, then

$$UWDD_s^{-1}G\Pi = \begin{bmatrix}\Gamma & 0\end{bmatrix} \tag{4.45}$$

is the desired LQ factorization.

4.9 1D C-V Derivative Matrix: $2n+1$ Columns; Second Factorization

In this section we will look at another method for factorizing V'. This method was developed first but we think the “First Factorization” is more useful in practice; this factorization is recorded here on the chance it may be beneficial in the future.

From pervious work, we see

$$\begin{aligned} & WDD_s^{-1}\left(1,:\right) = \\ & \quad \begin{bmatrix}0 & 1\cdot 2^{-s} & 0 & 3\cdot 4^{-s} & 0 & 5\cdot 6^{-s} & \cdots & \left(2n-3\right)\left(2n-2\right)^{-s} & 0 & \left(2n-1\right)\left(2n\right)^{-s} & 0\end{bmatrix} \\ & WDD_s^{-1}\left(k, k+1\!:\!2n+1-k\right) = \\ & \quad \begin{bmatrix}2k\left(k+1\right)^{-s} & 0 & 2\left(k+2\right)\left(k+3\right)^{-s} & 0 & \cdots & 0 & 2\left(2n-k\right)\left(2n+1-k\right)^{-s}\end{bmatrix} \\ & \quad k \geq 2. \end{aligned} \tag{4.46}$$

The nonzero entries are (essentially) constant along the columns; the first row is off by a factor of $\frac{1}{2}$. To fix this, we define

$$\widehat{D} = \operatorname{diag}\left(1, \frac{1}{2}, \cdots, \frac{1}{2}\right). \tag{4.47}$$

Now, we look at the case when $n = 5$ to clearly see the structure:

$$\widehat{D}WDD_s^{-1} = \begin{bmatrix} 0 & 1\cdot 2^{-s} & & 3\cdot 4^{-s} & & 5\cdot 6^{-s} & & 7\cdot 8^{-s} & & 9\cdot 10^{-s} & 0 \\ & & 2\cdot 3^{-s} & & 4\cdot 5^{-s} & & 6\cdot 7^{-s} & & 8\cdot 9^{-s} & & \\ & & & 3\cdot 4^{-s} & & 5\cdot 6^{-s} & & 7\cdot 8^{-s} & & & \\ & & & & 4\cdot 5^{-s} & & 6\cdot 7^{-s} & & & & \\ & & & & & 5\cdot 6^{-s} & & & & & \end{bmatrix}. \tag{4.48}$$

We begin by applying the same set of Givens rotations G from Eq. (4.42). When n is even, then

$$\widehat{D}WDD_s^{-1}G = \begin{bmatrix} 0 & \gamma_2 & 0 & \gamma_4 & 0 & \gamma_6 & \cdots & \gamma_{n-2} & 0 & \gamma_n & 0 & 0 & \cdots & 0 \\ & & \gamma_3 & 0 & \gamma_5 & 0 & \cdots & 0 & \gamma_{n-1} & 0 & \gamma_{n+1} & 0 & \cdots & 0 \\ & & & \gamma_4 & 0 & \gamma_6 & \cdots & \gamma_{n-2} & 0 & \gamma_n & 0 & 0 & \cdots & 0 \\ & & & & \gamma_5 & 0 & \cdots & 0 & \gamma_{n-1} & 0 & \gamma_{n+1} & 0 & \cdots & 0 \\ & & & & & \gamma_6 & \cdots & \gamma_{n-2} & 0 & \gamma_n & 0 & 0 & \cdots & 0 \\ & & & & & & \ddots & & \vdots & & \vdots & & & \\ & & & & & & & \gamma_{n-2} & 0 & \gamma_n & 0 & 0 & \cdots & 0 \\ & & & & & & & & \gamma_{n-1} & 0 & \gamma_{n+1} & 0 & \cdots & 0 \\ & & & & & & & & & \gamma_n & 0 & 0 & \cdots & 0 \\ & & & & & & & & & & \gamma_{n+1} & 0 & \cdots & 0 \end{bmatrix}. \tag{4.49}$$

If n is odd, then

$$\widehat{D}WDD_s^{-1}G = \begin{bmatrix} 0 & \gamma_2 & 0 & \gamma_4 & 0 & \gamma_6 & \cdots & 0 & \gamma_{n-1} & 0 & \gamma_{n+1} & 0 & \cdots & 0 \\ & & \gamma_3 & 0 & \gamma_5 & 0 & \cdots & \gamma_{n-2} & 0 & \gamma_n & 0 & 0 & \cdots & 0 \\ & & & \gamma_4 & 0 & \gamma_6 & \cdots & 0 & \gamma_{n-1} & 0 & \gamma_{n+1} & 0 & \cdots & 0 \\ & & & & \gamma_5 & 0 & \cdots & \gamma_{n-2} & 0 & \gamma_n & 0 & 0 & \cdots & 0 \\ & & & & & \gamma_6 & \cdots & 0 & \gamma_{n-1} & 0 & \gamma_{n+1} & 0 & \cdots & 0 \\ & & & & & & \ddots & & \vdots & & \vdots & & & \\ & & & & & & & \gamma_{n-2} & 0 & \gamma_n & 0 & 0 & \cdots & 0 \\ & & & & & & & & \gamma_{n-1} & 0 & \gamma_{n+1} & 0 & \cdots & 0 \\ & & & & & & & & & \gamma_n & 0 & 0 & \cdots & 0 \\ & & & & & & & & & & \gamma_{n+1} & 0 & \cdots & 0 \end{bmatrix}. \tag{4.50}$$

In both cases, we see that the matrix is upper triangular, and we wish to put it in lower triangular form. Because of the similarities between the two cases (n even or odd), we will focus on the case when n is even. This is solely a choice of convenience and will not affect the results.

To convert this into a lower triangular matrix, we see that nonzero even and odd column pairs have nonzero terms following the form

$$\begin{bmatrix} \gamma & \delta \\ \gamma & \delta \\ \vdots & \vdots \\ \gamma & \delta \\ 0 & \delta \end{bmatrix}. \tag{4.51}$$

In both cases we can use Givens rotations to zero out of the second column. We will repeatedly do this and work backward to obtain the LQ factorization. After a permutation of the columns, the matrix will be in lower triangular form.

We define

$$\begin{aligned} \delta_{n+1} &= \gamma_{n+1} \\ \delta_n &= \gamma_n \\ \delta_k &= \gamma_k \sqrt{1 + \left(\frac{\delta_{k+2}}{\gamma_k}\right)^2} \quad k \in \{2, \cdots, n-1\} \end{aligned}$$

$$\begin{aligned}
\xi_k &= -\frac{\delta_{k+2}}{\gamma_k} \quad k \in \{2, \cdots, n-1\} \\
\mu_k &= \frac{1}{\sqrt{1+\xi_k^2}} \\
\nu_k &= \xi_k \mu_k.
\end{aligned} \tag{4.52}$$

Letting

$$\begin{aligned}
H_k &= \begin{bmatrix} I_{k-1} & & & & \\ & \mu_k & & \nu_k & \\ & & 1 & & \\ & -\nu_k & & \mu_k & \\ & & & & I_{2n-1-k} \end{bmatrix} \\
H &= H_{n-1} H_{n-2} \cdots H_3 H_2,
\end{aligned} \tag{4.53}$$

we see

$$\widehat{D} W D D_s^{-1} G H \Pi = \begin{bmatrix} L & 0 \end{bmatrix}, \tag{4.54}$$

where Π is the circular downshift permutation matrix as before, 0 is an $n \times n+1$ matrix of zeros, and L is a lower triangular matrix with

$$\begin{aligned}
L(2k+\ell, \ell) &= (-1)^k \delta_{2k+\ell} \nu_{2k+\ell-2} \nu_{2k+\ell-4} \cdots \nu_\ell \quad \ell \in \{2,3\}, k \geq 0, 2k+\ell \leq n \\
L(2k+\ell, \ell) &= (-1)^k \delta_{2k+\ell} \nu_{2k+\ell-2} \nu_{2k+\ell-4} \cdots \nu_\ell \mu_{\ell-2} \quad \ell \geq 4, k \geq 0, 2k+\ell \leq n.
\end{aligned} \tag{4.55}$$

As an example, we show the case when $n = 8$:

$$L = \begin{bmatrix}
\delta_2 & & & & & & & \\
& \delta_3 & & & & & & \\
-\delta_4 \nu_2 & & \delta_4 \mu_2 & & & & & \\
& -\delta_5 \nu_3 & & \delta_5 \mu_3 & & & & \\
\delta_6 \nu_4 \nu_2 & & -\delta_6 \nu_4 \mu_2 & & \delta_6 \mu_4 & & & \\
& \delta_7 \nu_5 \nu_3 & & -\delta_7 \nu_5 \mu_3 & & \delta_7 \mu_5 & & \\
-\delta_8 \nu_6 \nu_4 \nu_2 & & \delta_8 \nu_6 \nu_4 \mu_2 & & -\delta_8 \nu_6 \mu_4 & & \delta_8 \mu_6 & \\
& -\delta_9 \nu_7 \nu_5 \nu_3 & & \delta_9 \nu_7 \nu_5 \mu_3 & & -\delta_9 \nu_7 \mu_5 & & \delta_9 \mu_7
\end{bmatrix}. \tag{4.56}$$

A direct computation will show us

$$\delta_k = ||\widehat{D}WDD_s^{-1}\left(k-1,:\right)||_2 \quad k \in \left\{2, \cdots, n+1\right\}, \tag{4.57}$$

but it is easier to see that the only nonzero entry of the kth row of $\widehat{D}WDD_s^{-1}GH_n \cdots H_{k+1}$ is δ_k, and the 2-norm is rotationally invariant.

Working through some calculations we find that L^{-1} has a very simple form: bidiagonal matrix with nonzero entries on the second sub-diagonal.

$$L^{-1} =$$
$$\begin{bmatrix}
\delta_2^{-1} & & & & & & & \\
0 & \delta_3^{-1} & & & & & & \\
\delta_2^{-1}\xi_2 & 0 & \left(\delta_4\mu_2\right)^{-1} & & & & & \\
 & \delta_3^{-1}\xi_3 & 0 & \left(\delta_5\mu_3\right)^{-1} & & & & \\
 & & \ddots & & \ddots & & & \\
 & & & & & \left(\delta_{n-1}\mu_{n-3}\right)^{-1} & & \\
 & & & & \ddots & 0 & \left(\delta_n\mu_{n-2}\right)^{-1} & \\
 & & & & & \delta_{n-1}^{-1}\xi_{n-1} & 0 & \left(\delta_{n+1}\mu_{n-1}\right)^{-1}
\end{bmatrix} \tag{4.58}$$

Because of its simpler form, it will be easier to work with L^{-1} directly rather than L. If we P is the permutation matrix which permutes the even and odd rows together in increasing order, then it is clear

$$PL^{-1}P = \mathrm{diag}(L_1, L_2), \tag{4.59}$$

where L_1 and L_2 are lower triangular matrices with nonzeros only on the sub-diagonal, implying they are semiseparable [32, Chapter 12].

The methods we applied here could also be applied when V' has $3n+1$ or $4n+1$ columns, but we will not pursue those factorizations at this time. Left multiplying by U greatly simplifies matters because then UWD and $UWDD_s^{-1}$ have orthogonal rows; this is our reasoning for preferring the other factorization.

4.10 1D C-V Derivative Matrix: $3n+1$ Columns; First Factorization

If we use $3n+1$ columns for interpolating derivative information, then we see

$$UWDD_s^{-1} = \begin{bmatrix} 0 & \alpha_2 & & & & & & \alpha_{2n} & 0 & \alpha_{2n+2} & & & \\ & & \ddots & & & & \iddots & & & & \ddots & & \\ & & & \alpha_n & & \alpha_{n+2} & & & & & & \alpha_{3n} & \\ & & & & \alpha_{n+1} & & & & & & & & \alpha_{3n+1} \end{bmatrix}. \tag{4.60}$$

Here,

$$\begin{aligned} \alpha_k &= (k-1)\,k^{-s} \quad k \in \{2, \cdots, 2n\} \\ \alpha_k &= -\,(k-1)\,k^{-s} \quad k \in \{2n+2, \cdots, 3n+1\}\,. \end{aligned} \tag{4.61}$$

By using disjoint Givens rotations, we let rotation matrix is

$$G = G_1 G_2, \tag{4.62}$$

where

$$G_1 = \begin{bmatrix} 1 & & & & & & & & 0 & & \\ & c_2 & & & & & & s_2 & & & \\ & & \ddots & & & & \iddots & & & & \\ & & & c_n & & s_n & & & & & \\ & & & & 1 & & & & & & \\ & & & -s_n & & c_n & & & & & \\ & & \iddots & & & & \ddots & & & & \\ & -s_2 & & & & & & c_2 & & & \\ 0 & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{bmatrix} \tag{4.63}$$

$$G_2 = \begin{bmatrix} 1 & & & & & 0 & 0 & & & & \\ & \chi_2 & & & & & & \sigma_2 & & & \\ & & \ddots & & & & & & \ddots & & \\ & & & \chi_n & & & & & & \sigma_n & \\ & & & & \chi_{n+1} & & & & & & \sigma_{n+1} \\ 0 & & & & & 1 & & & & & 0 \\ & & & & & & \ddots & & & & \\ 0 & & & & & & 1 & & & & 0 \\ & -\sigma_2 & & & & & & \chi_2 & & & \\ & & \ddots & & & & & & \ddots & & \\ & & & -\sigma_n & & & & & & \chi_n & \\ & & & & -\sigma_{n+1} & 0 & 0 & & & & \chi_{n+1} \end{bmatrix}. \tag{4.64}$$

Here, we define

$$\begin{aligned} \tau_k &= -\left(\frac{2n+1-k}{k-1}\right)\left(\frac{k}{2n+2-k}\right)^s \\ c_k &= \frac{1}{\sqrt{1+\tau_k^2}} \\ s_k &= \tau_k c_k \\ \zeta_k &= \left(\frac{2n-1+k}{k-1}\right)\left(\frac{k}{2n+k}\right)^s c_k \quad k \in \{2, \cdots, n\} \\ \zeta_{n+1} &= 3\left(\frac{n+1}{3n+1}\right)^s \\ \chi_k &= \frac{1}{\sqrt{1+\xi_k^2}} \\ \sigma_k &= \zeta_k \chi_k. \end{aligned} \tag{4.65}$$

In this case, we see

$$UWDD_s^{-1}G\Pi = \begin{bmatrix} \Gamma & 0 \end{bmatrix}, \tag{4.66}$$

where Γ is diagonal with entries

$$\begin{aligned} \gamma_k = {} & \sqrt{(k-1)^2 k^{-2s} + (2n+1-k)^2(2n+2-k)^{-2s} + (2n-1+k)^2(2n+k)^{-2s}} \\ & k \in \{2, \cdots, n\} \end{aligned}$$

$$\gamma_{n+1} = \sqrt{n^2(n+1)^{-2s} + (3n)^2(3n+1)^{-2s}}. \tag{4.67}$$

4.11 1D C-V Derivative Matrix: $4n+1$ Columns; First Factorization

If we use $4n+1$ Chebyshev polynomials for interpolation (up to degree $4n$), then we must compute the LQ factorization of the following matrix:

$$UWDD_s^{-1} = \begin{bmatrix} 0 & \alpha_2 & & & & \alpha_{2n} & 0 & \alpha_{2n+2} & & & & \alpha_{4n} & 0 \\ & & \ddots & & \iddots & & & & \ddots & & \iddots & & \\ & & & \alpha_{n+1} & & & & & & \alpha_{3n+1} & & & \end{bmatrix}. \tag{4.68}$$

Here, we have

$$\begin{aligned} \alpha_k &= (k-1)\, k^{-s} \quad k \in \{2, \cdots, 2n\} \\ \alpha_k &= -\,(k-1)\, k^{-s} \quad k \in \{2n+2, \cdots, 4n\} \end{aligned} \tag{4.69}$$

By using disjoint Givens rotations, we let rotation matrix is

$$G = G_1 G_2, \tag{4.70}$$

where

$$G_1 = \begin{bmatrix}
1 & & & & & & & & 0 & & & & & & & & \\
& c_2 & & & & & & s_2 & & & & & & & & & \\
& & \ddots & & & & \iddots & & & & & & & & & & \\
& & & c_n & & s_n & & & & & & & & & & & \\
& & & & 1 & & & & & & & & & & & & \\
& & & -s_n & & c_n & & & & & & & & & & & \\
& & \iddots & & & & \ddots & & & & & & & & & & \\
& -s_2 & & & & & & c_2 & & & & & & & & & \\
0 & & & & & & & & 1 & & & & & & & & 0 \\
& & & & & & & & & \mu_2 & & & & & & \nu_2 & \\
& & & & & & & & & & \ddots & & & & \iddots & & \\
& & & & & & & & & & & \mu_n & & \nu_n & & & \\
& & & & & & & & & & & & 1 & & & & \\
& & & & & & & & & & & -\nu_n & & \mu_n & & & \\
& & & & & & & & & & \iddots & & & & \ddots & & \\
& & & & & & & & & -\nu_2 & & & & & & \mu_2 & \\
& & & & & & & & 0 & & & & & & & & 1
\end{bmatrix} \tag{4.71}$$

$$G_2 = \begin{bmatrix}
1 & & & & & & & 0 & & & & & & & \\
& \chi_2 & & & & & & & \sigma_2 & & & & & & \\
& & \ddots & & & & & & & \ddots & & & & & \\
& & & \chi_n & & & & & & & \sigma_n & & & & \\
& & & & \chi_{n+1} & & & & & & & \sigma_{n+1} & & & \\
& & & & & 1 & & & & & & & & & \\
& & & & & & \ddots & & & & & & & & \\
0 & & & & & & & 1 & & & & & & & \\
& -\sigma_2 & & & & & & & \chi_2 & & & & & & \\
& & \ddots & & & & & & & \ddots & & & & & \\
& & & -\sigma_n & & & & & & & \chi_n & & & & \\
& & & & -\sigma_{n+1} & & & & & & & \chi_{n+1} & & & \\
& & & & & & & & & & & & 1 & & \\
& & & & & & & & & & & & & \ddots & \\
& & & & & & & & & & & & & & 1
\end{bmatrix} . \tag{4.72}$$

Here, we define

$$\begin{aligned}
\tau_k &= -\left(\frac{2n+1-k}{k-1}\right)\left(\frac{k}{2n+2-k}\right)^s \\
c_k &= \frac{1}{\sqrt{1+\tau_k^2}} \\
s_k &= \tau_k c_k \\
\xi_k &= -\left(\frac{4n+1-k}{2n-1+k}\right)\left(\frac{2n+k}{4n+2-k}\right)^s \\
\mu_k &= -\frac{1}{\sqrt{1+\xi_k^2}} \\
\nu_k &= \xi_k \mu_k \\
\zeta_k &= \left(\frac{2n-1+k}{k-1}\right)\left(\frac{k}{2n+k}\right)^s \frac{c_k}{\mu_k} \quad k \in \{2, \cdots, n\} \\
\zeta_{n+1} &= 3\left(\frac{n+1}{3n+1}\right)^s \\
\chi_k &= \frac{1}{\sqrt{1+\zeta_k^2}} \\
\sigma_k &= \zeta_k \chi_k.
\end{aligned} \tag{4.73}$$

In this case, we see

$$UWDD_s^{-1}G\Pi = \begin{bmatrix}\Gamma & 0\end{bmatrix}, \tag{4.74}$$

where Γ is diagonal with entries

$$\begin{aligned}
\gamma_k = &\left[(k-1)^2 k^{-2s} + (2n+1-k)^2(2n+2-k)^{-2s}\right. \\
&\left. +(2n-1+k)^2(2n+k)^{-2s} + (4n+1-k)^2(4n+2-k)^{-2s}\right]^{1/2} \\
&k \in \{2, \cdots, n\} \\
\gamma_{n+1} = &\sqrt{n^2(n+1)^{-2s} + (3n)^2(3n+1)^{-2s}}.
\end{aligned} \tag{4.75}$$

4.12 1D C-V Derivative Matrix: $2n+1$ Columns with Point Interpolation; First Factorization

In this section we compute the resulting LQ factorization for the interpolation problem

$$\begin{cases} p'(z_k^n) = f'(z_k^n), & k \in \{1, \cdots, n\} \\ p(0) = f(0) \end{cases}. \tag{4.76}$$

Ensuring equality at $p(0) = f(0)$ gives the MSN approximation the correct constant term. If we denote the linear interpolation requirement by A, then we have

$$\begin{bmatrix} A \\ UWD \end{bmatrix} D_s^{-1} G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \Gamma & 0 \end{bmatrix}, \tag{4.77}$$

where G is from Eq. (4.42) and we remember $Ge_1 = e_1$. This is the LQ factorization.

4.13 1D C-V Derivative Matrix: $2n+1$ Columns with Endpoint Interpolation; First Factorization

In this section we compute the solution when interpolating derivatives on Chebyshev nodes and function values on the endpoints:

$$\begin{cases} p'(z_k^n) = f'(z_k^n), & k \in \{1, \cdots, n\} \\ p(z) = f(z), & z \in \{-1, 1\} \end{cases}. \tag{4.78}$$

We will use tools from Sec. 4.8. When n is even, we see

$$\begin{bmatrix} UWD \\ \overline{A} \end{bmatrix} D_s^{-1} G \Pi =$$

$$\begin{bmatrix} \gamma_2 & & & & & & & & & & & \\ & \gamma_3 & & & & & & & & & & \\ & & & \ddots & & & & & & & & \\ & & & & \gamma_n & & & & & & & \\ & & & & & \gamma_{n+1} & & & & & & \\ \alpha_2 & & \alpha_4 & \cdots & \alpha_n & & \alpha_{n+2} & & \cdots & \alpha_{2n} & & \\ & \alpha_3 & & \cdots & & \alpha_{n+1} & & \alpha_{n+3} & \cdots & & \alpha_{2n+1} & \alpha_1 \end{bmatrix}, \tag{4.79}$$

where $\overline{A} = C_1^{-1}A$ with definitions coming from Eqs. (4.8) and (4.9). We must then compute 2 Householder reflectors by setting

$$\begin{aligned} v_1^* &= \begin{bmatrix} \alpha_{n+2} & 0 & \alpha_{n+4} & \cdots & \alpha_{2n} & 0 & 0 \end{bmatrix} \\ v_2^* &= \begin{bmatrix} 0 & \alpha_{n+3} & 0 & \cdots & 0 & \alpha_{2n+1} & \alpha_1 \end{bmatrix} \\ u_1 &= \frac{1}{\left|\left|v_1 - \left|\left|v_1\right|\right|_2 e_1\right|\right|_2} \left(v_1 - \left|\left|v_1\right|\right|_2 e_1\right) \\ u_2 &= \frac{1}{\left|\left|v_2 - \left|\left|v_2\right|\right|_2 e_2\right|\right|_2} \left(v_2 - \left|\left|v_2\right|\right|_2 e_2\right). \end{aligned} \tag{4.80}$$

From here, we let

$$\begin{aligned} \widehat{P}_1 &= I_{n+1} - 2u_1 u_1^* \\ \widehat{P}_2 &= I_{n+1} - 2u_2 u_2^* \\ \widehat{P} &= \widehat{P}_1 \widehat{P}_2 \\ &= I_{n+1} - 2\begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} u_1 & u_2 \end{bmatrix}^* \\ P &= \begin{bmatrix} I_n & 0 \\ 0 & \widehat{P} \end{bmatrix}, \end{aligned} \tag{4.81}$$

so that

$$\begin{bmatrix} UWD \\ \overline{A} \end{bmatrix} D_s^{-1} G \Pi P = \begin{bmatrix} \gamma_2 &&&&&&& \\ & \gamma_3 &&&&&& \\ && \gamma_4 &&&&& \\ &&& \ddots &&&& \\ &&&& \gamma_n &&& \\ &&&&& \gamma_{n+1} && \\ \alpha_2 & 0 & \alpha_4 & \cdots & \alpha_n & 0 & ||v_1||_2 & \\ 0 & \alpha_3 & 0 & \cdots & 0 & \alpha_{n+1} & 0 & ||v_2||_2 \end{bmatrix}. \tag{4.82}$$

When n is odd, we let

$$\overline{A} = C_0^{-1} A \tag{4.83}$$

so that

$$\begin{bmatrix} UWD \\ \overline{A} \end{bmatrix} D_s^{-1} G \Pi = \begin{bmatrix} \gamma_2 &&&&&&&&&&& \\ & \gamma_3 &&&&&&&&&& \\ && \gamma_4 &&&&&&&&& \\ &&& \ddots &&&&&&&& \\ &&&& \gamma_n &&&&&&& \\ &&&&& \gamma_{n+1} &&&&&& \\ & \alpha_3 && \cdots & \alpha_n && \alpha_{n+2} && \cdots && \alpha_{2n+1} & \alpha_1 \\ \alpha_2 && \alpha_4 & \cdots && \alpha_{n+1} && \alpha_{n+3} & \cdots & \alpha_{2n} && \end{bmatrix}. \tag{4.84}$$

We set

$$\begin{aligned} v_1^* &= \begin{bmatrix} \alpha_{n+2} & 0 & \cdots & 0 & \alpha_{2n+1} & \alpha_1 \end{bmatrix} \\ v_2^* &= \begin{bmatrix} 0 & \alpha_{n+3} & 0 & \cdots & 0 & \alpha_{2n} & 0 & 0 \end{bmatrix} \end{aligned} \tag{4.85}$$

and proceed as before.

4.14 1D C-V Derivative Matrix: $2n+1$ Columns with Endpoint Interpolation; Second Factorization

In this section we compute the LQ factorization from last section using the second factorization of the V'; the endpoint function values are included as before and begin by assuming n is even.

We know that $C^{-1}V'D_s^{-1}GH$ essentially reduces our matrix to lower triangular form, and the circular downshift permutation Π puts the significant portion into lower triangular form. Thus, after these applications, we arrive at the matrix

$$\begin{bmatrix} \widehat{D}WD \\ \overline{A} \end{bmatrix} D_s^{-1}GH\Pi =$$
$$\begin{bmatrix}
* & & & & & & & & & & & \\
* & * & & & & & & & & & & \\
* & * & * & & & & & & & & & \\
\vdots & & & \ddots & & & & & & & & \\
* & * & * & \cdots & * & & & & & & & \\
* & * & * & \cdots & * & * & & & & & & \\
\alpha_2 & & \alpha_4 & \cdots & \alpha_n & & \alpha_{n+2} & & \cdots & \alpha_{2n} & & \\
& \alpha_3 & & \cdots & & \alpha_{n+1} & & \alpha_{n+3} & \cdots & & \alpha_{2n+1} & \alpha_1
\end{bmatrix}. \tag{4.86}$$

The last two rows can be computed quickly by evaluating $\overline{A}D_s^{-1}GH\Pi$.

All we have left is to compute 2 Householder reflectors, setting

$$\begin{aligned}
v_1^* &= \begin{bmatrix}\alpha_{n+2} & 0 & \alpha_{n+4} & \cdots & \alpha_{2n} & 0 & 0\end{bmatrix} \\
v_2^* &= \begin{bmatrix}0 & \alpha_{n+3} & 0 & \cdots & 0 & \alpha_{2n+1} & \alpha_1\end{bmatrix} \\
u_1 &= \frac{1}{\left|\left|v_1 - \left|\left|v_1\right|\right|_2 e_1\right|\right|_2}\left(v_1 - \left|\left|v_1\right|\right|_2 e_1\right) \\
u_2 &= \frac{1}{\left|\left|v_2 - \left|\left|v_2\right|\right|_2 e_2\right|\right|_2}\left(v_2 - \left|\left|v_2\right|\right|_2 e_2\right).
\end{aligned} \tag{4.87}$$

From here, we let

$$
\begin{aligned}
\widehat{P}_1 &= I_{n+1} - 2u_1u_1^* \\
\widehat{P}_2 &= I_{n+1} - 2u_2u_2^* \\
\widehat{P} &= \widehat{P}_1\widehat{P}_2 \\
&= I_{n+1} - 2\begin{bmatrix} u_1 & u_2 \end{bmatrix}\begin{bmatrix} u_1 & u_2 \end{bmatrix}^* \\
P &= \begin{bmatrix} I_n & 0 \\ 0 & \widehat{P} \end{bmatrix}.
\end{aligned} \tag{4.88}
$$

Putting this together, we have

$$
\begin{bmatrix} \widehat{D}WD \\ \overline{A} \end{bmatrix} D_s^{-1}GH\Pi P = \begin{bmatrix}
* & & & & & & & \\
* & * & & & & & & \\
* & * & * & & & & & \\
\vdots & & & \ddots & & & & \\
* & * & * & \cdots & * & & & \\
* & * & * & \cdots & * & * & & \\
\alpha_2 & 0 & \alpha_4 & \cdots & \alpha_n & 0 & ||v_1||_2 & \\
0 & \alpha_3 & 0 & \cdots & 0 & \alpha_{n+1} & 0 & ||v_2||_2
\end{bmatrix}, \tag{4.89}
$$

where we have only kept the nonzero terms. The inverse of this triangular matrix can be computed quickly.

When n is odd, we let

$$
\overline{A} = C_0^{-1}A. \tag{4.90}
$$

We then find

$$
\begin{bmatrix} \widehat{D}WD \\ \overline{A} \end{bmatrix} D_s^{-1}GH\Pi =
\begin{bmatrix}
* & & & & & & & & & & & \\
* & * & & & & & & & & & & \\
* & * & * & & & & & & & & & \\
\vdots & & & \ddots & & & & & & & & \\
* & * & * & \cdots & * & & & & & & & \\
* & * & * & \cdots & * & * & & & & & & \\
 & \alpha_3 & & \cdots & \alpha_n & & \alpha_{n+2} & & \cdots & & \alpha_{2n+1} & \alpha_1 \\
\alpha_2 & & \alpha_4 & \cdots & & \alpha_{n+1} & & \alpha_{n+3} & \cdots & \alpha_{2n} & &
\end{bmatrix}. \tag{4.91}
$$

Setting

$$\begin{aligned} v_1^* &= \begin{bmatrix}\alpha_{n+2} & 0 & \cdots & 0 & \alpha_{2n+1} & \alpha_1\end{bmatrix} \\ v_2^* &= \begin{bmatrix}0 & \alpha_{n+3} & 0 & \cdots & 0 & \alpha_{2n} & 0 & 0\end{bmatrix}, \end{aligned} \tag{4.92}$$

we proceed as before to arrive at the LQ factorization.

4.15 1D C-V Birkhoff Interpolation Matrix: $2n+1$ Columns, First Factorization

In this section we investigate the full Birkhoff interpolation problem in 1D: interpolation and derivative information on the Chebyshev nodes.

We must compute the LQ factorization of

$$\begin{bmatrix}V \\ V'\end{bmatrix} D_s^{-1}. \tag{4.93}$$

After applying the inverse DCT, this reduces to

$$\begin{bmatrix}C^{-1} & \\ & C^{-1}\end{bmatrix}\begin{bmatrix}V \\ V'\end{bmatrix} D_s^{-1} = \begin{bmatrix}W \\ WD\end{bmatrix} D_s^{-1}. \tag{4.94}$$

If G is taken from Eq. (4.5) and U from Eq. (3.33), then we see

$$\begin{bmatrix}WD_s^{-1}G \\ EUWDD_s^{-1}G\end{bmatrix} =$$

$$\begin{bmatrix}
y_1 & & & & & & & & & & & & & & \\
& y_2 & & & & & & & & & & & & & \\
& & y_3 & & & & & & & & & & & & \\
& & & y_4 & & & & & & & & & & & \\
& & & & \ddots & & & & & & & & & & \\
& & & & & y_{n-1} & & & & & & & & & \\
& & & & & & y_n & & & & & & & & \\
& & & & & & & \alpha_{n+1} & & & & & & & \\
& & & & & & \alpha_n & & \alpha_{n+2} & & & & & & \\
& & & & & \alpha_{n-1} & & & & \alpha_{n+3} & & & & & \\
& & & & \iddots & & & & & & \ddots & & & & \\
& & & \alpha_4 & & & & & & & & \alpha_{2n-2} & & & \\
& & \alpha_3 & & & & & & & & & & \alpha_{2n-1} & & \\
0 & \alpha_2 & & & & & & & & & & & & \alpha_{2n} & 0
\end{bmatrix}. \tag{4.95}$$

Here, if α is the vector which contains α_i, then we see

$$\begin{aligned}
w^* &= \begin{bmatrix} 0 & 1 & 2 & 3 & \cdots & 2n-3 & 2n-2 & 2n-1 & 0 \end{bmatrix} \\
\alpha &= G^* D_s^{-1} w.
\end{aligned} \tag{4.96}$$

We now have the LQ factorization.

Chapter 5

C-V Matrices and Factorizations for Interpolation in Higher Dimensions

After working through 1D interpolation in Chapter 4, we now turn our attention to interpolation problems in higher dimensions.

5.1 C-V Matrices in Higher Dimensions

One difficulty going from 1D to 2D interpolation, as noted in Chapter 1, is how to chose the correct set of basis functions [30]. Because we are not constrained by matching the number of interpolation requirements with basis functions, our work is easier. We let $\overline{V} = V^y \otimes V^x$ be our 2D C-V matrix. Here, $V^x \in \mathbb{R}^{n\times(2n+1)}$ is our 1D C-V matrix in x and $V^y \in \mathbb{R}^{m\times(2m+1)}$ is our 1D C-V matrix in y. We similarly define $\overline{C} = C^y \otimes C^x$ and $\overline{G} = G^y \otimes G^x$. Due to the nature of the Kronecker product, many of the properties from 1D interpolation are kept but the method for computing them must be modified.

For interpolation on the Chebyshev nodes in 2D, we want to solve the following system:

$$\min_{\overline{V}a=f} \left|\left|\overline{D}_s a\right|\right|_2 . \tag{5.1}$$

The solution will be of the form

$$p(x,y) = \sum_{k=0}^{2n} \sum_{\ell=0}^{2m} a_{k,\ell} T_k(x) T_\ell(y). \tag{5.2}$$

Because of how $\overline{V}$ is defined, the coefficients are ordered

$$a^* = \begin{bmatrix} a_{0,0} & a_{1,0} & \cdots & a_{2n,0} & a_{0,1} & a_{1,1} & \cdots & a_{2n,1} & \cdots & a_{0,2m} & a_{1,2m} & \cdots & a_{2n,2m} \end{bmatrix}. \tag{5.3}$$

Furthermore, our interpolation points are ordered

$$\left(x_1, y_1\right), \left(x_2, y_1\right), \cdots, \left(x_n, y_1\right), \left(x_1, y_2\right), \cdots. \tag{5.4}$$

Thus, if $x = \overline{D}_s a$ where then we need to compute

$$\min_{\overline{VD}_s^{-1} x = f} \left\|x\right\|_2. \tag{5.5}$$

Our work reduces to computing the LQ factorization of $\overline{VD}_s^{-1}$.

Now, we know

$$\overline{C}^{-1} \overline{VD}_s^{-1} \overline{G} = \begin{bmatrix} Y^y & 0 \end{bmatrix} \otimes \begin{bmatrix} Y^x & 0 \end{bmatrix} \tag{5.6}$$

This is *not* the *LQ* factorization, but it is after a permutation of the columns.

We now assume $m = n$. Let

$$I = \bigcup_{k=1}^{n} \left[\left(k-1\right)\left(2n+1\right) + 1 : \left(k-1\right)\left(2n+1\right) + n\right], \tag{5.7}$$

so that

$$\left(Y^y \otimes Y^x\right) y\left(I\right) = f, \tag{5.8}$$

where we initialize the remaining elements of y to zero. At this point, we compute

$$a = \overline{D}_s^{-1} \overline{G} y. \tag{5.9}$$

Similar results can be extended to higher dimensions. While it is not necessary to require $m = n$, we will in the future because it simplifies matters.

5.2 2D C-V Interpolation Matrix: $2n + 1$ Columns with Boundary

We assume $V^x, V^y \in \mathbb{R}^{n\times(2n+1)}$.

In this section we work through the matrix factorization of for interpolating on the tensor product of C-V matrices given by

$$\min_{\overline{V}a=f} \left|\left|\overline{D}_s a\right|\right|_2 , \tag{5.10}$$

where

$$\overline{C}^{-1}\overline{V} = \begin{bmatrix} W^y \otimes W^x \\ \overline{A^y} \otimes W^x \\ W^y \otimes \overline{A^x} \end{bmatrix} . \tag{5.11}$$

This corresponds to function interpolation on the points

$$Z = \left\{\left(z_j^n, z_k^n\right)\right\}_{j\in\{1,\cdots,n\},k\in\{1,\cdots,n\}} \cup \left\{\left(\hat{z}, z_k^n\right)\right\}_{\hat{z}\in\{-1,1\},k\in\{1,\cdots,n\}} \cup \left\{\left(z_k^n, \hat{z}\right)\right\}_{\hat{z}\in\{-1,1\},k\in\{1,\cdots,n\}} . \tag{5.12}$$

We let G and P be the Givens rotation matrixes and Householder reflectors from Sec. (4.4) for size n, so that

$$\begin{bmatrix} W^y \otimes W^x \\ \overline{A^y} \otimes W^x \\ W^y \otimes \overline{A^x} \end{bmatrix} \left(D_s^{-1}GP \otimes D_s^{-1}GP\right) = \begin{bmatrix} \begin{bmatrix} Y^y & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} Y^x & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} \widehat{A^y} & \widehat{L^y} & 0 \end{bmatrix} \otimes \begin{bmatrix} Y^x & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} Y^y & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} \widehat{A^x} & \widehat{L^x} & 0 \end{bmatrix} \end{bmatrix} . \tag{5.13}$$

After a permutation of the columns, this is the desired LQ factorization.

To do so, we let

$$
\begin{aligned}
I_1 &= \bigcup_{k=1}^{n} \left[(k-1)(2n+1)+1 : (k-1)(2n+1)+n\right] \\
I_2 &= \bigcup_{k=n+1}^{n+2} \left[(k-1)(2n+1)+1 : (k-1)(2n+1)+n\right] \\
I_3 &= \bigcup_{k=1}^{n} \left[(k-1)(2n+1)+n+1 : (k-1)(2n+1)+n+2\right]. \qquad (5.14)
\end{aligned}
$$

In this case, we require

$$
\begin{aligned}
(Y^y \otimes Y^x)\, y(I_1) &= \hat{f}_1 \\
(\widehat{A^y} \otimes Y^x) y(I_1) + (\widehat{L^y} \otimes Y^x) y(I_2) &= \hat{f}_2 \\
(Y^y \otimes \widehat{A^x}) y(I_1) + (Y^y \otimes \widehat{L^x}) y(I_3) &= \hat{f}_3. \qquad (5.15)
\end{aligned}
$$

All other values are initialized to zero. Finally, we see

$$
a = \overline{D}_s^{-1}\overline{GP}y. \qquad (5.16)
$$

5.3 2D Full Birkhoff Interpolation Problem

We now look at the full Birkhoff interpolation problem in 2D:

$$
\begin{aligned}
f(z_i^n, z_j^n) &= p(z_i^n, z_j^n), \quad i, j \in \{1, \cdots, n\} \\
f_x(z_i^n, z_j^n) &= p_x(z_i^n, z_j^n), \quad i, j \in \{1, \cdots, n\} \\
f_y(z_i^n, z_j^n) &= p_y(z_i^n, z_j^n), \quad i, j \in \{1, \cdots, n\}. \qquad (5.17)
\end{aligned}
$$

That is, we interpolate all function and first derivative values on the $n \times n$ Chebyshev tensor grid.

After some of the standard matrix preprocessing, we must compute the LQ factorization of the following matrix:

$$\begin{bmatrix} WD_s^{-1} \otimes WD_s^{-1} \\ EUWD_s^{-1} \otimes WD_s^{-1} \\ WD_s^{-1} \otimes EUWD_s^{-1} \end{bmatrix}. \tag{5.18}$$

We assume that we are interpolating up to degree N in both x and y (not specified except that $N \geq 2n$). If G is the structured rotation matrix so that

$$WD_s^{-1}G = \begin{bmatrix} Y & 0 \end{bmatrix}, \tag{5.19}$$

then

$$\begin{bmatrix} WD_s^{-1} \otimes WD_s^{-1} \\ EUWD_s^{-1} \otimes WD_s^{-1} \\ WD_s^{-1} \otimes EUWD_s^{-1} \end{bmatrix} (G \otimes G) = \begin{bmatrix} \begin{bmatrix} Y & 0 \end{bmatrix} \otimes \begin{bmatrix} Y & 0 \end{bmatrix} \\ EUWD_s^{-1}G \otimes \begin{bmatrix} Y & 0 \end{bmatrix} \\ \begin{bmatrix} Y & 0 \end{bmatrix} \otimes EUWD_s^{-1}G \end{bmatrix}. \tag{5.20}$$

We now take H to be the rotation matrix so that

$$\begin{aligned} \begin{bmatrix} Y & 0 \end{bmatrix} H &= \begin{bmatrix} Y & 0 \end{bmatrix} \\ \left(EUWD_s^{-1}G\right) H &= \begin{bmatrix} A & B & 0 \end{bmatrix}. \end{aligned} \tag{5.21}$$

This rotation matrix H comes up in the full Birkhoff problem in 1D and will contain some Givens rotations when $N > 2n+1$. Using this, we see

$$\begin{bmatrix} \begin{bmatrix} Y & 0 \end{bmatrix} \otimes \begin{bmatrix} Y & 0 \end{bmatrix} \\ EUWD_s^{-1}G \otimes \begin{bmatrix} Y & 0 \end{bmatrix} \\ \begin{bmatrix} Y & 0 \end{bmatrix} \otimes EUWD_s^{-1}G \end{bmatrix} (H \otimes H) = \begin{bmatrix} \begin{bmatrix} Y & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} Y & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} A & B & 0 \end{bmatrix} \otimes \begin{bmatrix} Y & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} Y & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} A & B & 0 \end{bmatrix} \end{bmatrix}. \tag{5.22}$$

At this point, we can compute the solution, once we use the following index sets:

$$\begin{aligned} I_1 &= \bigcup_{k=1}^{2n} \left[(k-1)N + 1 : (k-1)N + n\right] \\ I_2 &= \bigcup_{k=1}^{n} \left[(k-1)N + (n+1) : (k-1)N + 2n\right]. \end{aligned} \tag{5.23}$$

In this case, we need to solve

$$\begin{aligned}\left(\begin{bmatrix} Y & 0 \\ A & B \end{bmatrix} \otimes Y\right) y(I_1) &= \begin{bmatrix} \hat{f} \\ (U \otimes I)\, \hat{f}_y \end{bmatrix} \\ (Y \otimes B)\, y(I_2) &= (I \otimes U)\, \hat{f}_x. \end{aligned} \tag{5.24}$$

The rest of the entries of y are initialized to zero. Here, we remember that we compute hat's by the 2D IDCT. From here, we can easily solve for the coefficients:

$$a = \overline{D}_s^{-1} \overline{GH} y. \tag{5.25}$$

This problem is useful because of how difficult it would be to solve in another setting. If we wish to keep the tensor structure with each matrix having the same number of columns N, then we have $3n^2$ linear constraints and N^2 total columns. It is not possible for $3n^2 = N^2$ to hold for all n and N when we are forced to have a square system. The MSN method does not care about this restriction and so we can choose N freely, so long as it is large enough that the structured rotations go through. While fast algorithms may exist without the tensor structure, it is not clear how this could be determined or exploited.

5.4 Extending Previous 2D C-V Interpolation Matrix Results

By looking at the results we have just computed, we see that to compute fast MSN interpolation problems, we "stack" our linear interpolation requirements and then use factorizations from 1D to build up the structured LQ factorization needed. This should work with other 2D interpolation problems in addition to problems in higher dimensions; the main difficulty in higher dimensions may be keeping track of all the linear requirements (as well as the appropriate index sets) and then inverting and applying the fast, structured matrices appropriately. Even so, the bulk of the computational complexity comes from the initial IDCT. To ensure that we can take advantage of the tensor product structure, it is necessary to make sure we have enough columns in each matrix.

Chapter 6

Examples of MSN Function Interpolation

In this chapter we present results using the fast MSN algorithms. We begin by interpolating smooth functions before attempting to interpolate rough functions (discontinuous functions or functions with infinite derivatives). Unless otherwise stated, all computations will be performed in double precision. In 1D, we compare the results with Lagrange interpolation on Chebyshev nodes. In 2D, we will compare MSN with interpolation on a tensor grid of Chebyshev nodes; this is computed using the 2D IDCT. When comparing against rough functions, we use some standard filters (discussed below).

6.1 Functions for Smooth Interpolation

The functions we will be approximating will be

$$\begin{aligned} f_R(x,y) &= \frac{1}{1+R(x^2+y-0.3)^2} + \frac{1}{1+R(x+y-0.4)^2} \\ &\quad + \frac{1}{1+R(x+y^2-0.5)^2} + \frac{1}{1+R(x^2+y^2-0.25)^2} \\ g_R(x) &= f_R(x,0.8). \end{aligned} \tag{6.1}$$

f_R is a complicated function: it has a Runge function on one line, one circle, and two parabolas. The function g_R is a coordinate slice of f_R.

6.2 Results for Fast MSN Interpolation in 1D for Smooth Functions

6.2.1 Interpolation Comparison

We begin by comparing results with MSN on Chebyshev nodes with standard Chebyshev interpolation in Fig. 6.1; this contains results for interpolation of g_{25} and g_{100}. The results for large s values closely match the Chebyshev interpolation; this makes sense because we are smoothly cutting off the Chebyshev coefficients, and larger s values lead to a sharper cutoff.

6.2.2 Birkhoff Interpolation Comparison

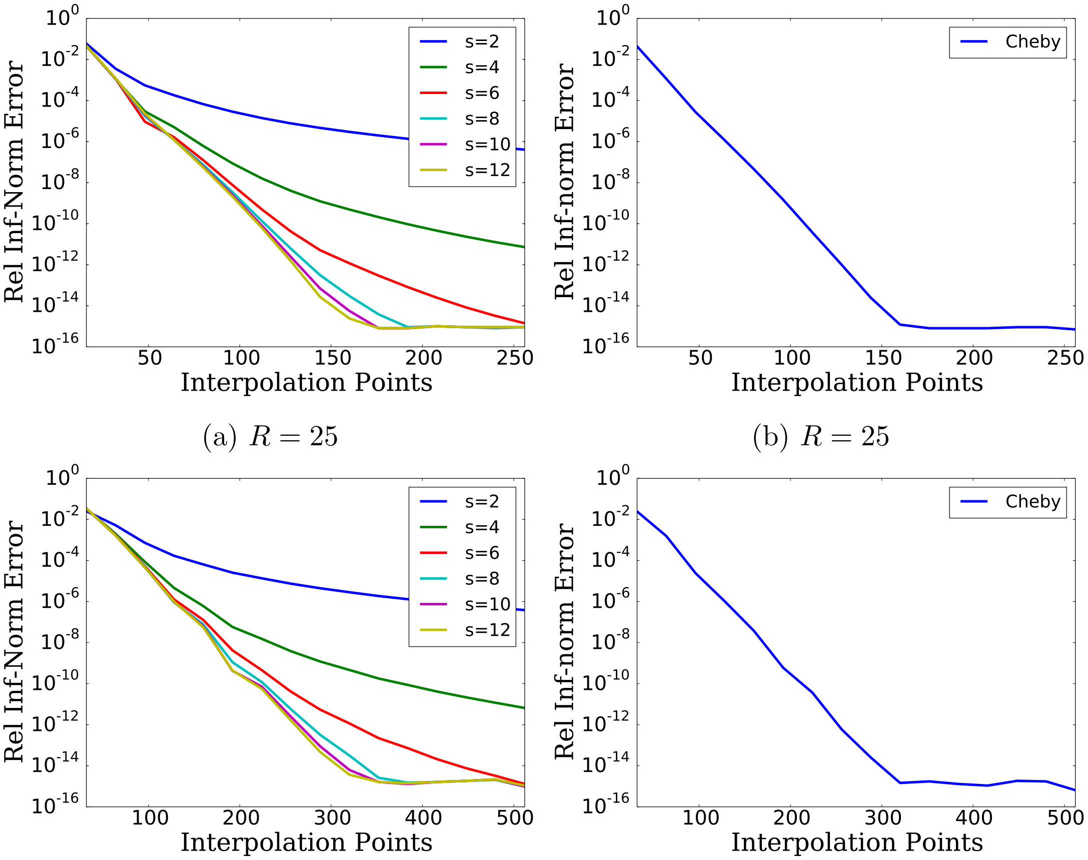
We now look at a Birkhoff interpolation problem for an approximation p:

$$\begin{aligned} p'(z_k^n) &= g_R'(z_k^n), \quad k \in \{1, \cdots, n\} \\ p(-1) &= g_R(-1) \\ p(1) &= g_R(1). \end{aligned} \tag{6.2}$$

Fast MSN methods exist and we compare against Chebyshev interpolation. In particular, the system

$$\begin{bmatrix} UWD \\ A \end{bmatrix} a = \begin{bmatrix} UC^{-1}f' \\ f \end{bmatrix}, \tag{6.3}$$

where f' holds the derivative information and f holds the endpoint interpolation, is solved using pivoted QR. The UWD matrix is truncated so that the entire system is square, and from our previous work we know UWD is mostly zero. The standard method for



(a) $R = 25$

(b) $R = 25$

(c) $R = 100$

(d) $R = 100$

Figure 6.1: MSN interpolation and Chebyshev interpolation results of the 1D Runge function g_{25} and g_{100} for various s values.

solving linear systems in Julia is pivoted LU [6], although there is no noticeable difference between pivoted QR and pivoted LU (results not shown). The results for can be seen in Fig. 6.2. When solving the square system, it is clear numerical difficulties are keeping the minimal error around 10^{-14}. The error decay of both methods are approximately the same.

Due to the structure of the linear system, it would be possible to implement a similar fast solver for the square system.

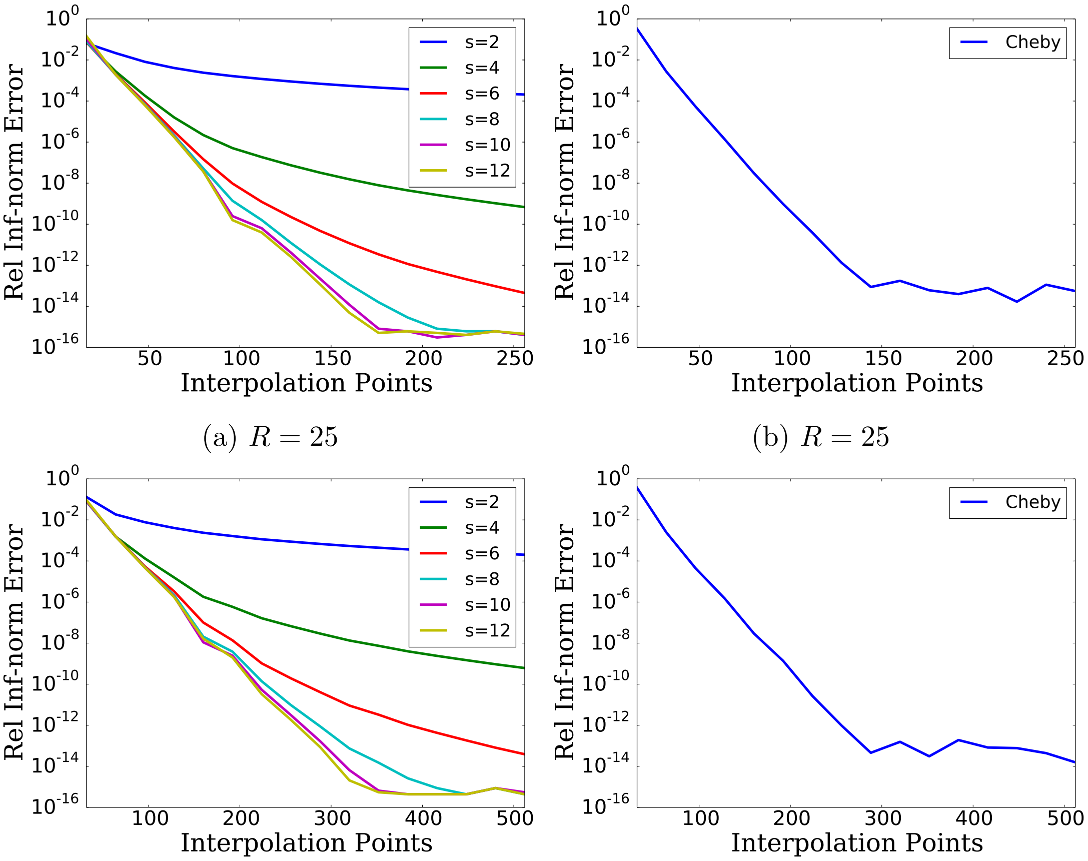
6.3 Results for Fast MSN Interpolation in 2D for Smooth Functions

We now turn our attention to the more difficult challenge of interpolating the 2D function f_R. For $R = 25$, we have results in single and double precision presented in Fig. 6.3. For $R = 100$, we have results in single and double precision presented in Fig. 6.4. From these examples, we see that, as before, MSN interpolation has the same level of approximation as Chebyshev interpolation for large s values.

6.4 Gibbs Phenomenon and Smooth Cutoff Filters

We now turn our attention from interpolating smooth functions to interpolating rough functions.

The Gibbs phenomenon [76, Chapter 2] is a well-known problem when one attempts to approximate a discontinuous function by a finite sum of continuous functions, frequently chosen to be a Fourier series; this is also true for Chebyshev expansions. Recent work has been focused on determining filters which smoothly cutoff the Fourier series; one review article is [34]. The filters σ in the review article take the form



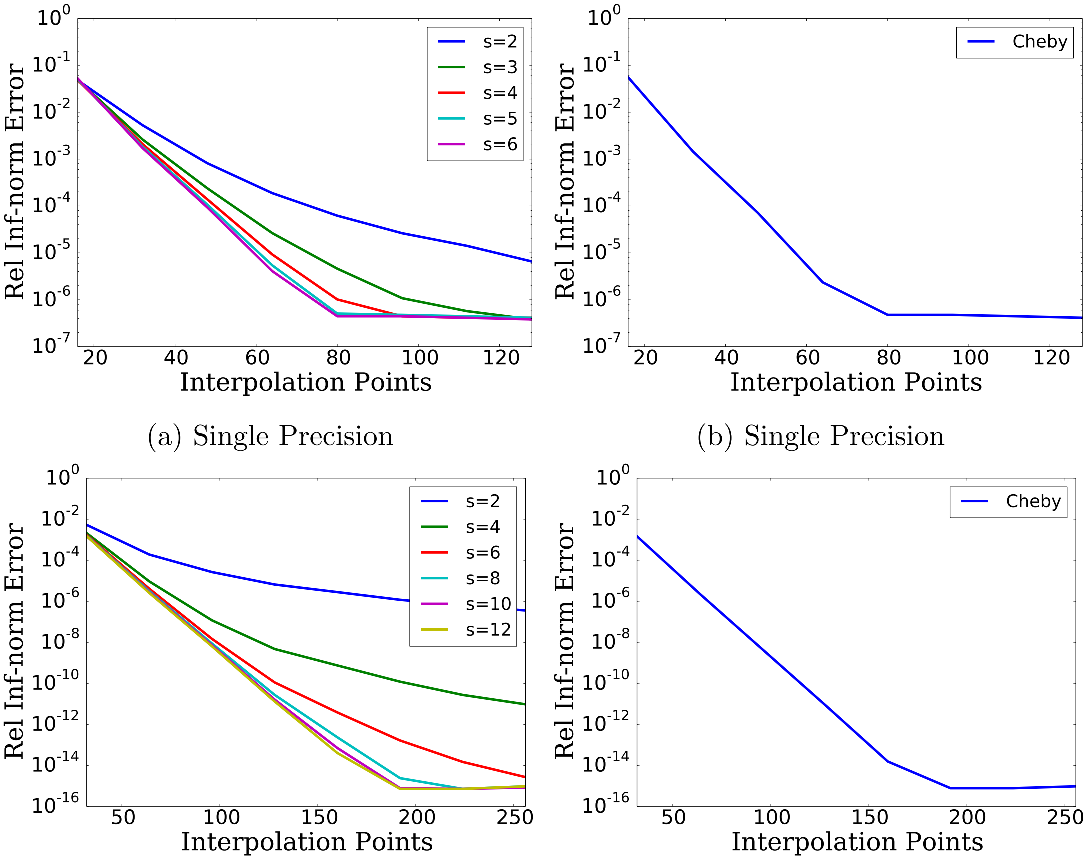
(a) $R = 25$

(b) $R = 25$

(c) $R = 100$

(d) $R = 100$

Figure 6.2: MSN Birkhoff interpolation and Chebyshev interpolation results of the 1D Runge function g_{25} and g_{100} for various s values.



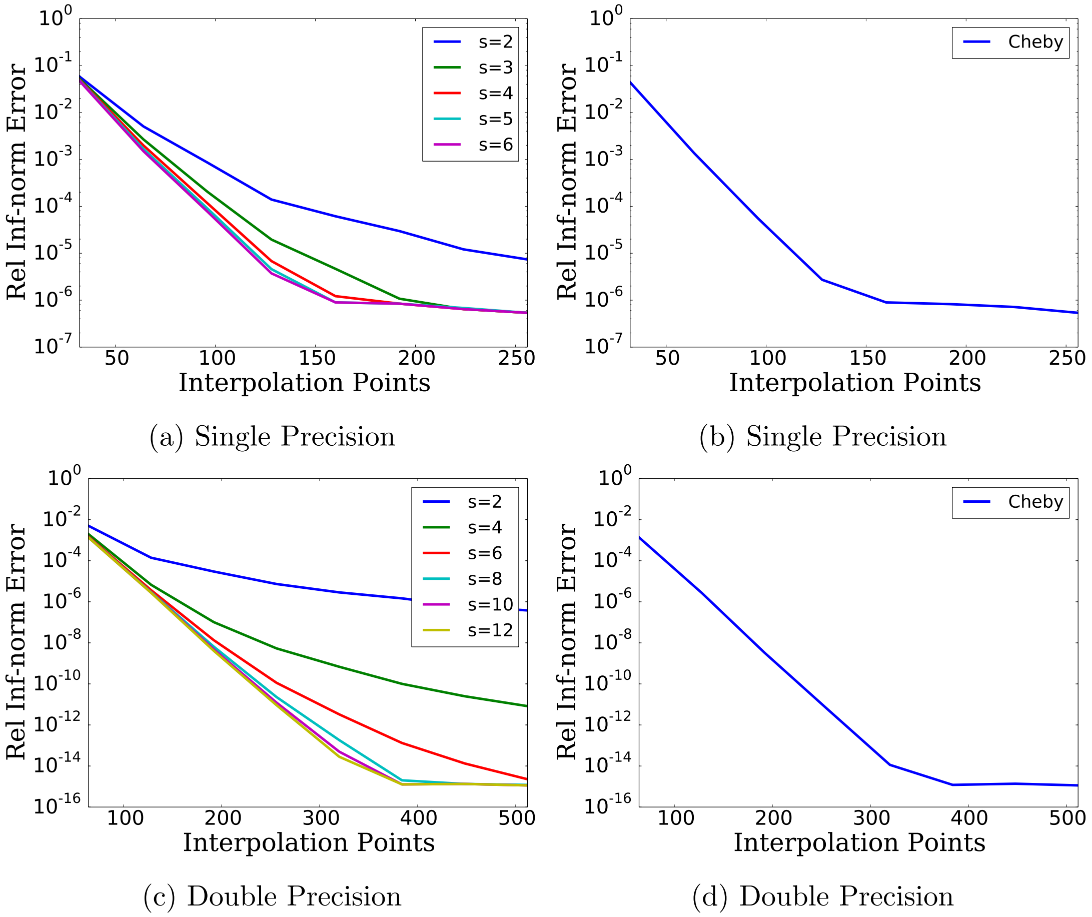
(a) Single Precision

(b) Single Precision

(c) Double Precision

(d) Double Precision

Figure 6.3: MSN interpolation and Chebyshev interpolation results of the 2D Runge function f_{25} for various s values. Here, n interpolation points refers to interpolation on the $n \times n$ tensor grid of Chebyshev points.



(a) Single Precision

(b) Single Precision

(c) Double Precision

(d) Double Precision

Figure 6.4: MSN interpolation and Chebyshev inteprolation results of the 2D Runge function f_{100} for various s values. Here, n interpolation points refers to interpolation on the $n \times n$ tensor grid of Chebyshev points.

$$f_N^\sigma(x) = \sum_{k=-\infty}^{\infty} \hat{f}_k \sigma\left(\frac{k}{N}\right) e^{ikx}, \tag{6.4}$$

where σ smoothly cuts off the higher order frequencies; a more precise definition is given below. This has a similar form of the smooth cutoff approximations for the MSN method found in [16, 18] as well as our proofs of Fast MSN convergence in Chapter 7. In this way, MSN interpolation could be thought of as a smoothing filter which acts in such a way so as to retain equality at certain nodes (in this case, the Chebyshev interpolation nodes). We will choose $s \in \{1, 2, 3, 4, 5, 6\}$ for our simulations, although in practice s can be any real number.

From [34], we say a filter σ is of order p if

- σ is an even function,
- $\sigma(0) = 1$, $\sigma^{(\ell)}(0) = 0$ for $1 \leq \ell \leq p - 1$,
- $\sigma(\eta) = 0$ for $|\eta| \geq 1$, and
- $\sigma \in C^{p-1}$ for $\eta \in (-\infty, \infty)$.

We reproduce some of the filters from [34] which we will use to compare with our MSN results. In all instances, we set $\sigma(\eta) = 0$ when $\eta > 1$.

- Chebyshev interpolation:

  $$\sigma_0(\eta) = 1. \tag{6.5}$$

  This results in no filtering but is kept as a baseline comparison. Clearly σ_0 is not continuous at $\eta = 1$.

- The Fejér filter:

  $$\sigma_1(\eta) = 1 - \eta. \tag{6.6}$$

- The Lanczos filter:

$$\sigma_2(\eta) = \frac{\sin(\pi\eta)}{\pi\eta}. \tag{6.7}$$

  This is the normalized sinc function.

- The Raised Cosine Filter:

$$\sigma_3(\eta) = \frac{1}{2}\left(1 + \cos(\pi\eta)\right). \tag{6.8}$$

- The Sharpened Raised Cosine Filter:

$$\sigma_4(\eta) = \sigma_3^4(\eta)\left(35 - 84\sigma_3(\eta) + 70\sigma_3^2(\eta) - 20\sigma_3^3(\eta)\right). \tag{6.9}$$

  This is an order 8 filter and will be denoted as "Cos8" in error plots.

- The Exponential Filter of order p:

$$\sigma_5(\eta) = \exp\left(-\alpha\eta^p\right). \tag{6.10}$$

  We will always take p to be even and $\alpha = -\ln(\varepsilon_{\text{mach}})$, so that the largest frequency will be $O(\varepsilon_{\text{mach}})$. Naturally, σ_5 is not actually continuous at $\eta = 1$.

6.5 Functions for Rough Interpolation

We will focus on interpolation where the function is not continuous or whose derivatives are not continuous and compare the results with standard Chebyshev interpolation and Chebyshev filters in 1D. Because there are fewer theoretical results, we will focus on analyzing some carefully chosen examples; in particular, we will investigate interpolating

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

$$\begin{aligned}
H_2(x) &= H(x+\tfrac{1}{2}) + H(x) + H(x-\tfrac{1}{2})\\
R(x) &= \begin{cases} \frac{1}{1+25x^2} & x<0\\ \frac{2}{1+25x^2} & x\geq 0\end{cases}\\
G(x,\alpha) &= |x|^{\alpha}\\
G_2(x,\alpha) &= G(x+\tfrac{1}{2},\alpha) + G(x,\alpha) + G(x-\tfrac{1}{2},\alpha).
\end{aligned} \tag{6.11}$$

Naturally, H is the Heaviside function, R has a jump discontinuity between two Runge functions, and $G(\cdot,\alpha)$ has infinite derivatives at the origin when $\alpha \in (0,1)$. All of these functions have difficulties at $x=0$, making them challenging interpolation problems. The error will be determined by computing the relative error at 1000 points in $[-1,-0.1)\cup(0.1,1]$. In the case of H_2 and G_2, the functions have multiple difficulties and the relative error will be computed in the region $[-1,-0.6)\cup(-0.4,-0.1)\cup(0.1,0.4)\cup(0.6,1]$.

6.6 Results for Fast MSN Interpolation in 1D for Rough Functions

6.6.1 Interpolation Comparison

The relative error results for the Heaviside function H can be found in Fig. 6.5, the results for the Runge jump function $R(x)$ can be found in Fig. 6.6, and the results for $G(\cdot,0.5)$ can be found in Fig. 6.7. In the case of having multiple difficulties, the results for H_2 can be found in Fig. 6.8, while the results for $G_2(\cdot,0.5)$ can be found in Fig. 6.9. We note that the Sharpened Cosine Filter σ_4 is denoted by “Cos8” in the legend. We will keep the unfiltered Chebyshev interpolant in both filter plots for reference.

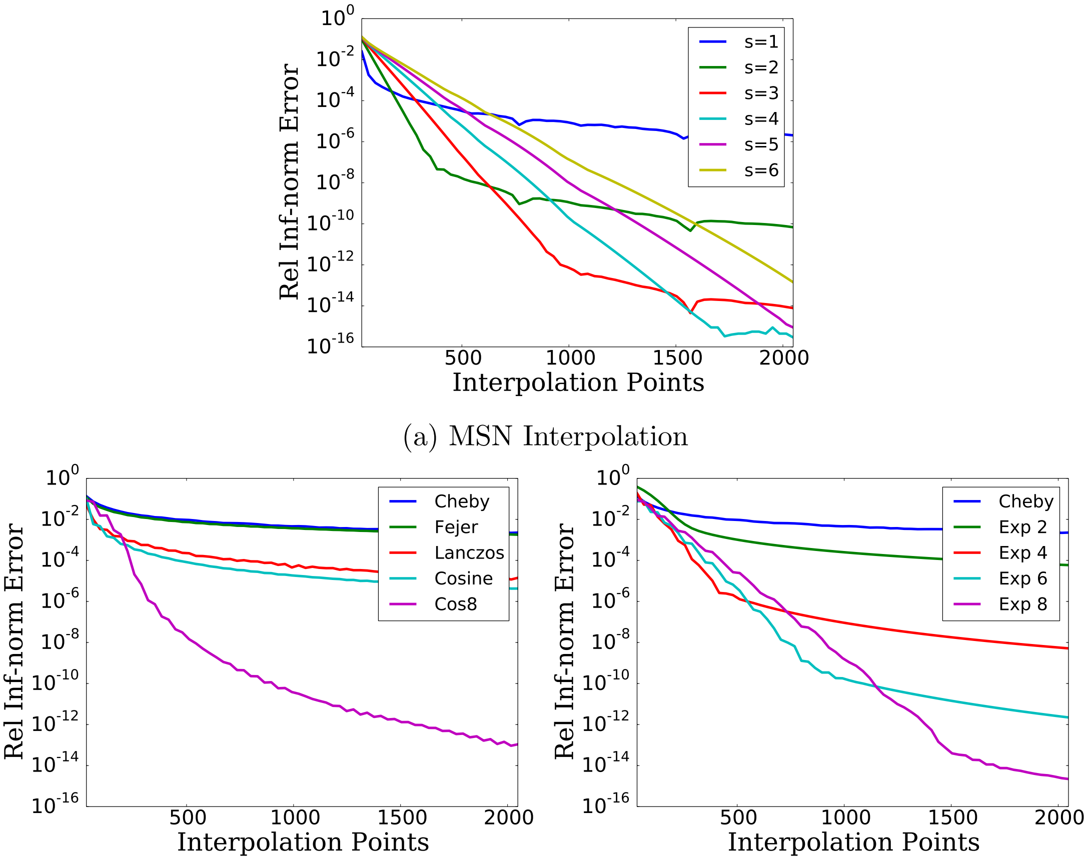
In every instance, MSN interpolation with $s=4$ appears to have the quickest error decrease while also reaching machine precision first. The best Chebyshev filters appear to to be the Sharpened Cosine filter (Cos8) as well as the higher order Exponential filters (order 6 and 8). The error profiles are approximately the same in every instance, regardless of whether the function is discontinuous (H and R), has infinite derivatives

(G), or multiple jumps or infinite derivatives (H_2 and G_2). The worst case for MSN interpolation is when $s = 1$. Even in this case, it has lower error than the Chebyshev, Fejér, Lanczos, Cosine, and Exp 2 filters in most instances. To make the comparison clear, in Fig. 6.10 where we plot the minimum error for both MSN interpolation and Chebyshev filters for the H_2 function. As we can see, the minimal MSN error usually approximately the same or better than the best filters. To see how different s values in MSN affect interpolation results, we include plots for various s values and interpolation points when approximating H_2 in Fig. 6.11.

In these examples so far we only used polynomials of degree $2n$ for n interpolation points; we have included results for degree $2n$, $4n$, $6n$, and $8n$ interpolation in Fig. 6.12 when attempting to interpolate H_2. As we can see, there is little difference between them. The difference in using larger interpolation degree only becomes apparent when using small s values, such as when $s \in (0, 1)$. By looking at the explicit form of the coefficients, we see that they do not play a large part; furthermore, by construction, once the IDCT gives somewhat accurate approximates of the true Chebyshev coefficients, larger s values closely match the first n coefficients while also matching the interpolation constraints. Thus, degree $2n$ MSN interpolation appears sufficient in these tests.

6.6.2 Birkhoff Interpolation Comparison

We now look the Birkhoff interpolation problem when we have a discontinuous derivative. The results for interpolating $G(\cdot, 0.5)$ can be found in Fig. 6.13; we compare MSN interpolation results against applying standard filters to the coefficients obtained from the solution in Eq. (6.3). As we can see, MSN interpolation performs significantly better than any of the filters; in fact, the MSN plot appears the same as the corresponding plot in Fig. 6.7 except for $s = 1$. Results for Birkhoff interpolation of $G_2(\cdot, 0.5)$ can be found in Fig. 6.14. The results are nowhere near as good, and neither of the methods (MSN or Chebyshev filters) perform well in this case. Better results may be possible

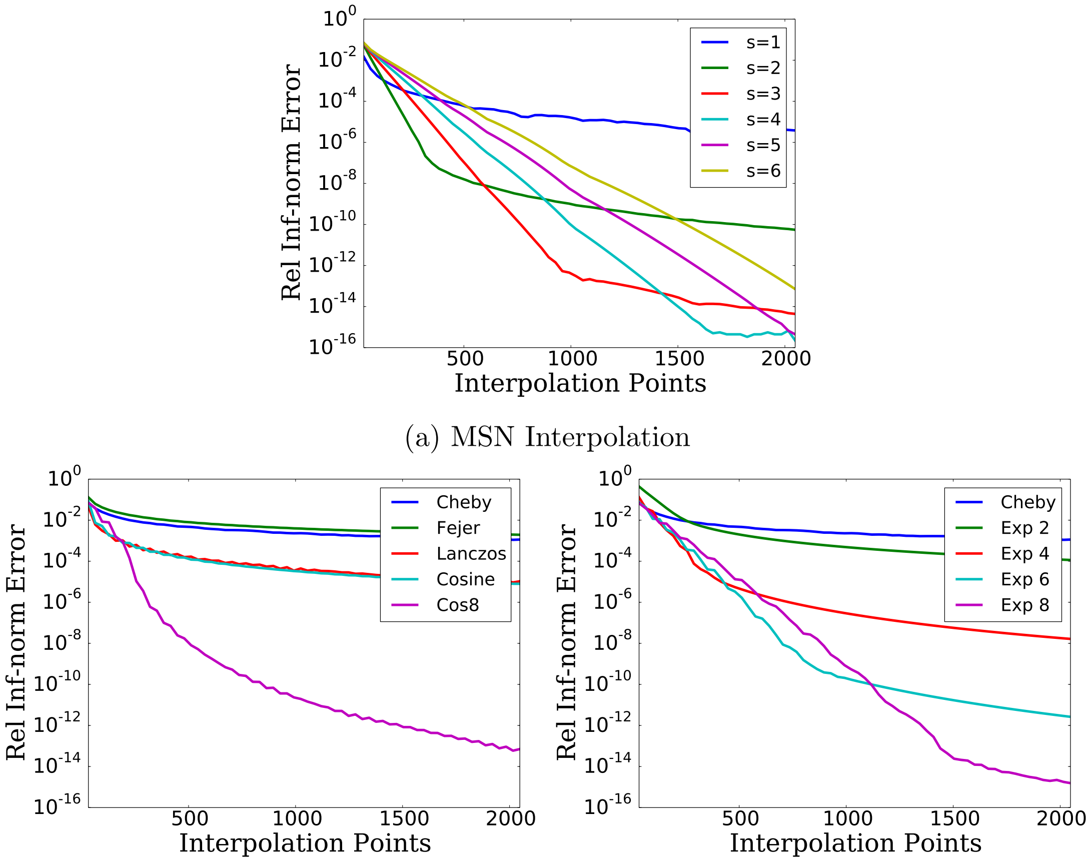


(a) MSN Interpolation

(b) Filters, Plot 1

(c) Filters, Plot 2

Figure 6.5: MSN interpolation and Chebyshev filtering results of the Heaviside jump function H for various s values and filters. We include standard Chebyshev interpolant in both filter examples for reference.

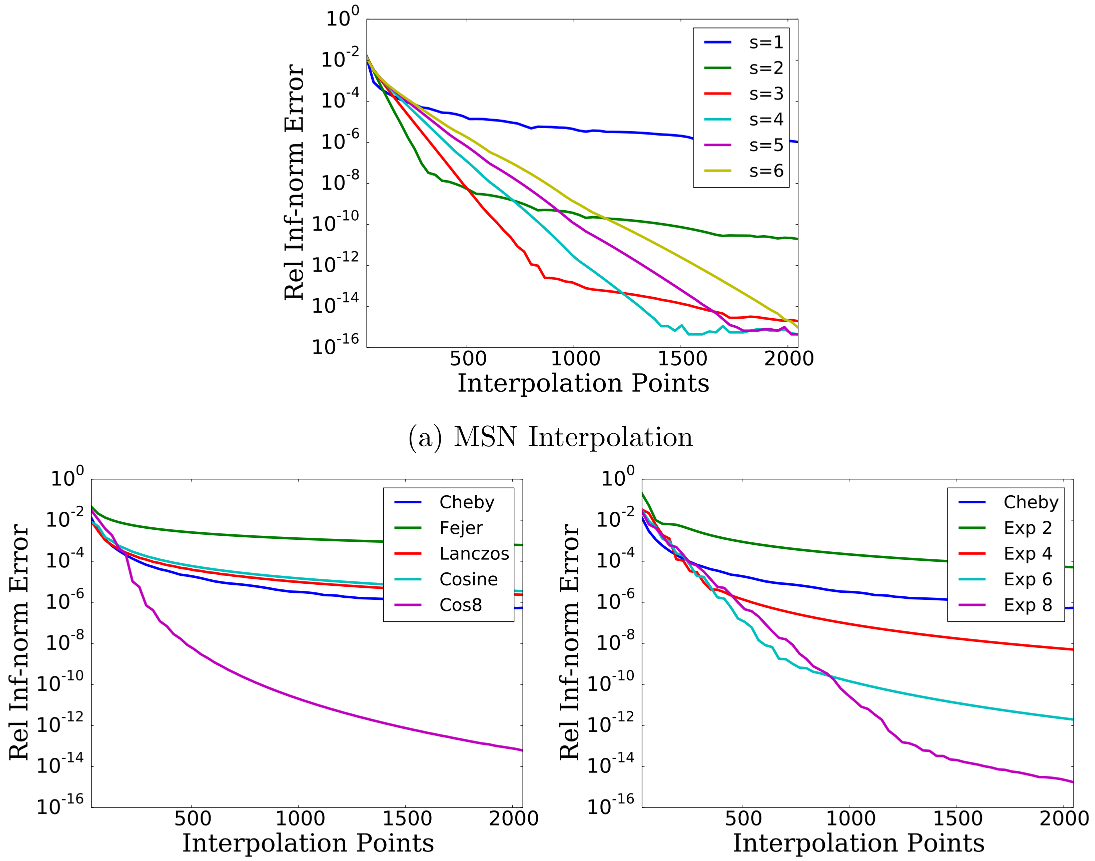


(a) MSN Interpolation

(b) Filters, Plot 1

(c) Filters, Plot 2

Figure 6.6: MSN interpolation and Chebyshev filtering results of the Runge jump function R for various s values and filters. We include standard Chebyshev interpolant in both filter examples for reference.

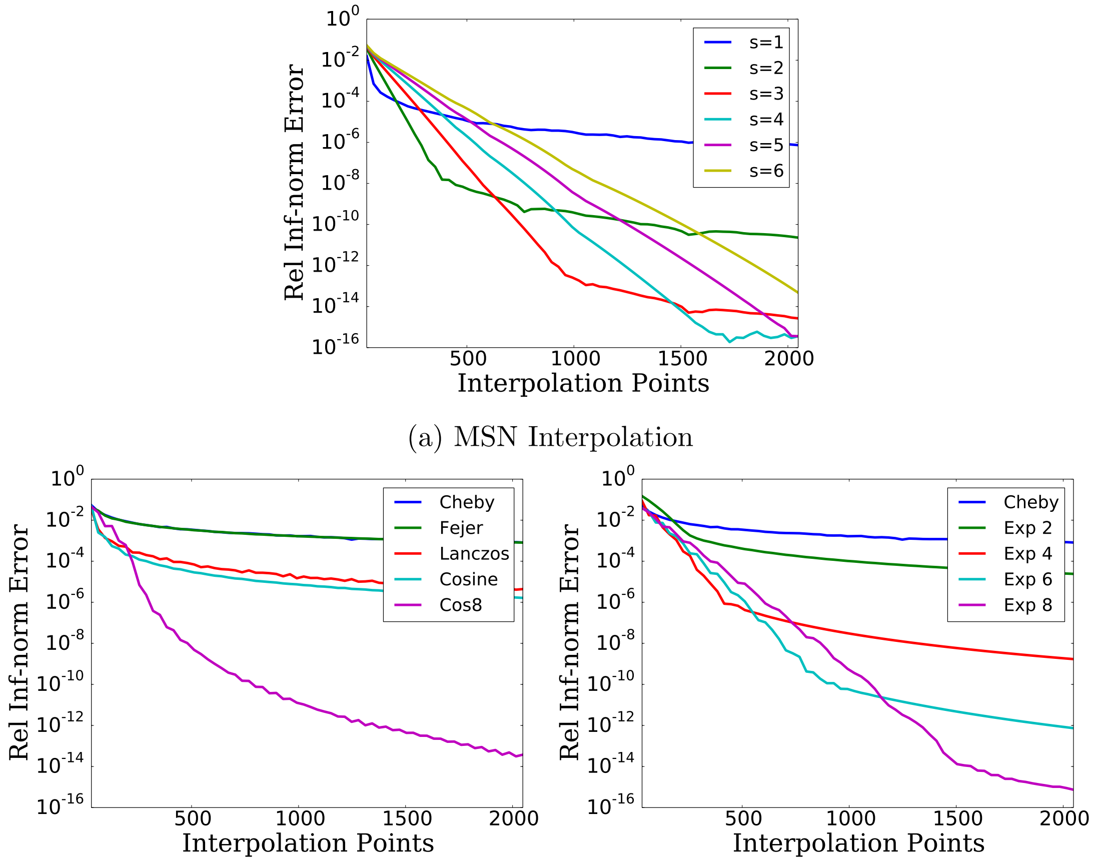


(a) MSN Interpolation

(b) Filters, Plot 1

(c) Filters, Plot 2

Figure 6.7: MSN interpolation and Chebyshev filtering results of the Sharp Function $G(\cdot, 0.5)$ for various s values and filters. We include standard Chebyshev interpolant in both filter examples for reference.

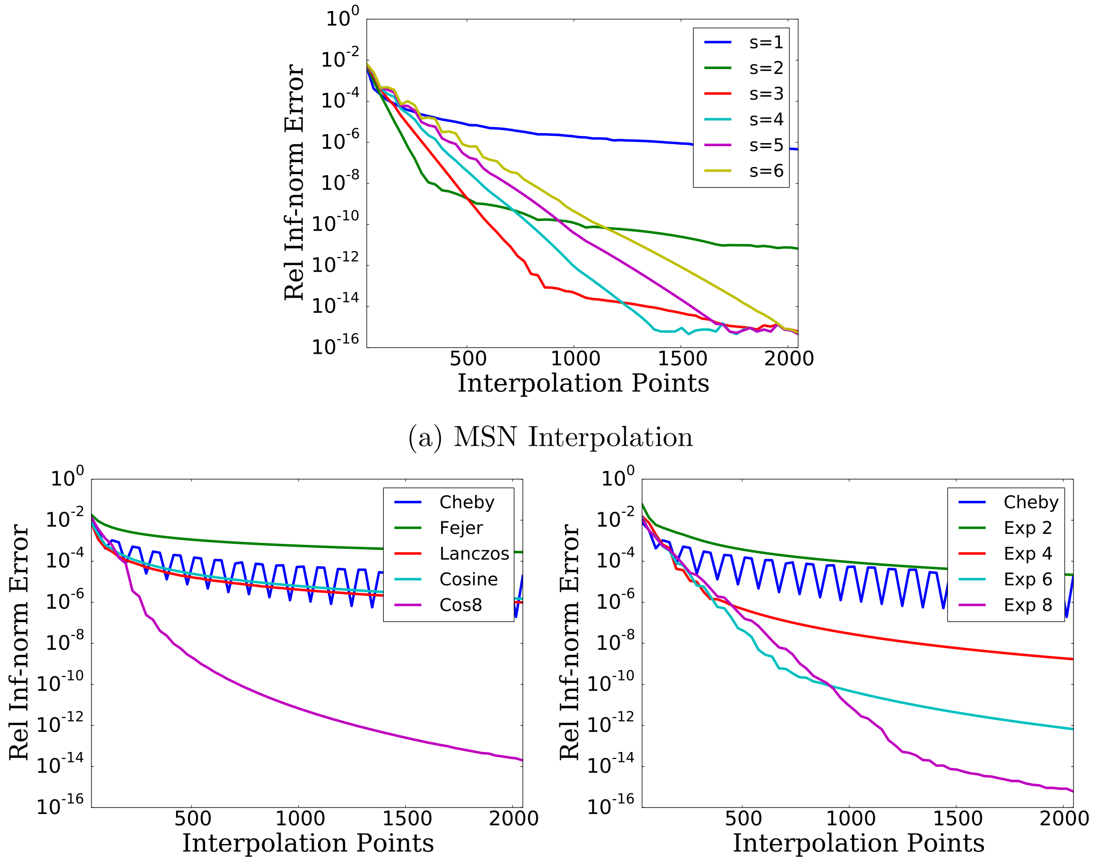


(a) MSN Interpolation

(b) Filters, Plot 1

(c) Filters, Plot 2

Figure 6.8: MSN interpolation and Chebyshev filtering results of the Heaviside jump function H_2 for various s values and filters. We include standard Chebyshev interpolant in both filter examples for reference.



(a) MSN Interpolation

(b) Filters, Plot 1

(c) Filters, Plot 2

Figure 6.9: MSN interpolation and Chebyshev filtering results of the Sharp Function $G_2(\cdot, 0.5)$ for various s values and filters. We include standard Chebyshev interpolant in both filter examples for reference.

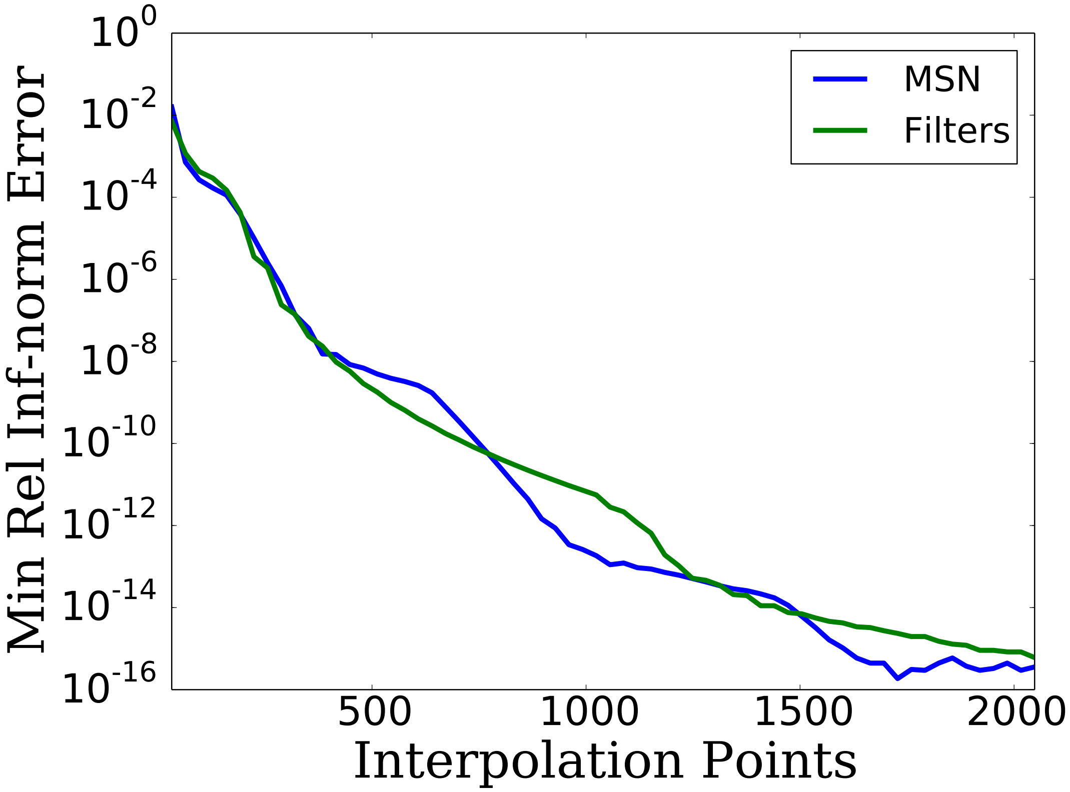


Figure 6.10: We plot the minimum MSN interpolation error and compare it to the minimum filter error over all Chebshev filters. These error results are from attempting to approximate H_2.

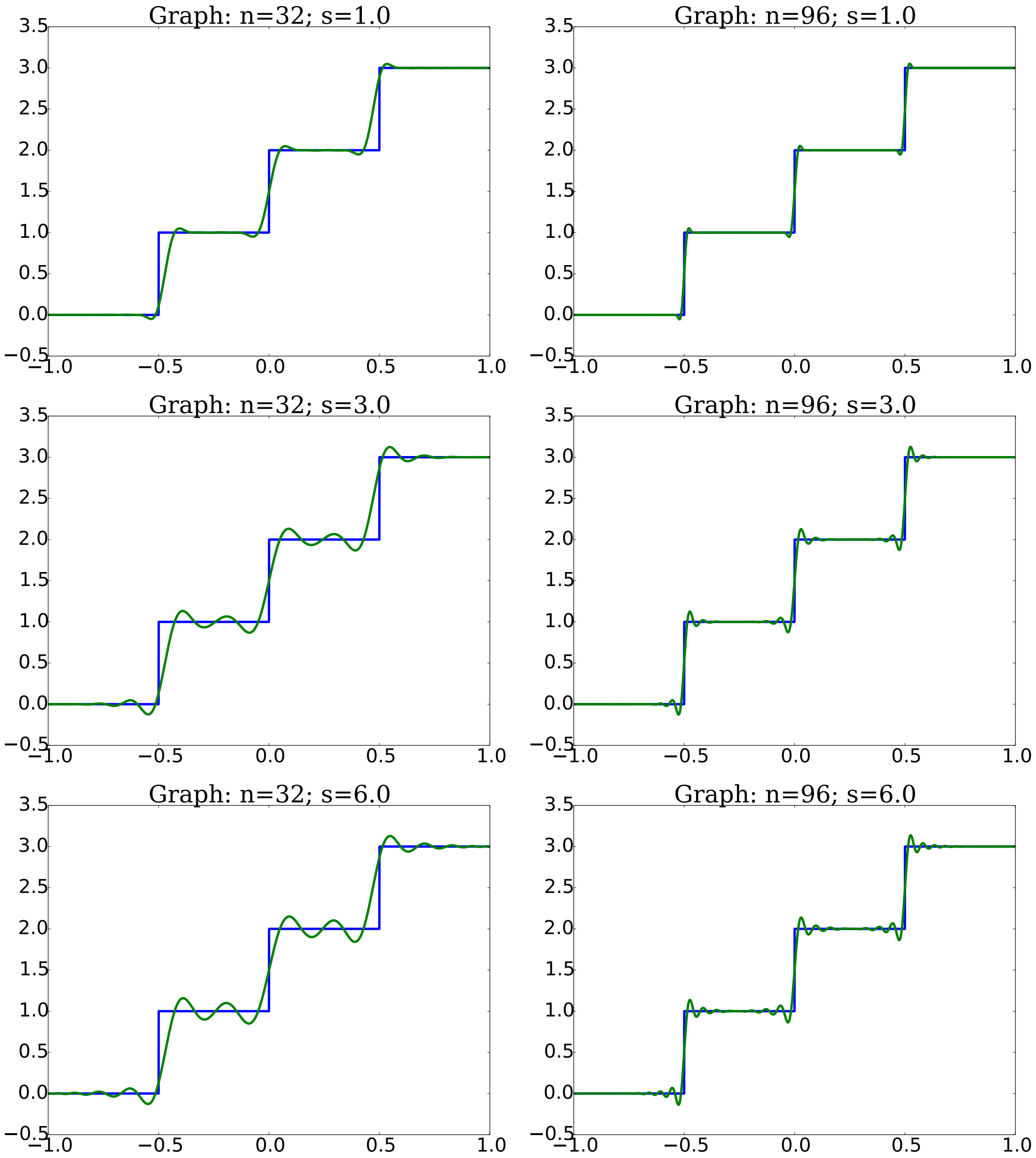
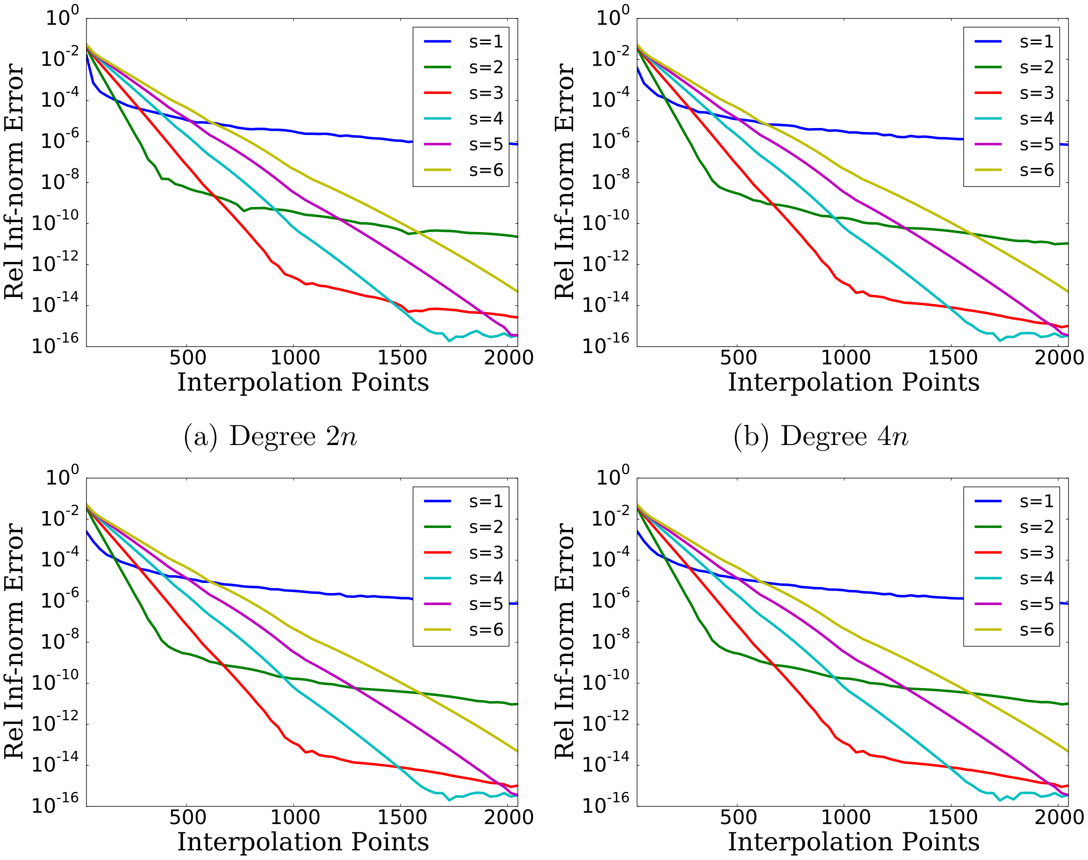


Figure 6.11: We plot MSN approximations against the true solution H_2 for various s and n values.



(a) Degree $2n$

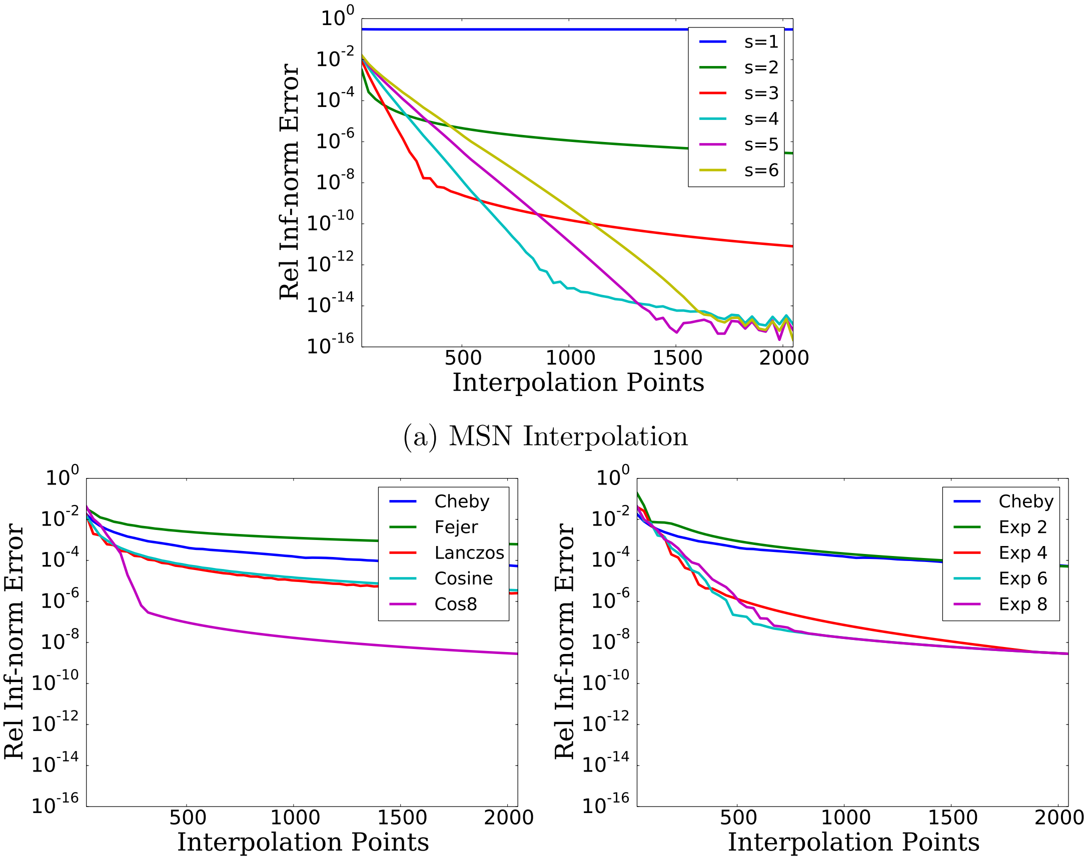
(b) Degree $4n$

(c) Degree $6n$

(d) Degree $8n$

Figure 6.12: We show results for different interpolation degree. For large s values, there is no apparent advantage for using a higher interpolation degree.

but likely require more interpolation nodes. In this case, even though the interpolation could be solved well with MSN, in this case the Birkhoff interpolation problem for general functions cannot.

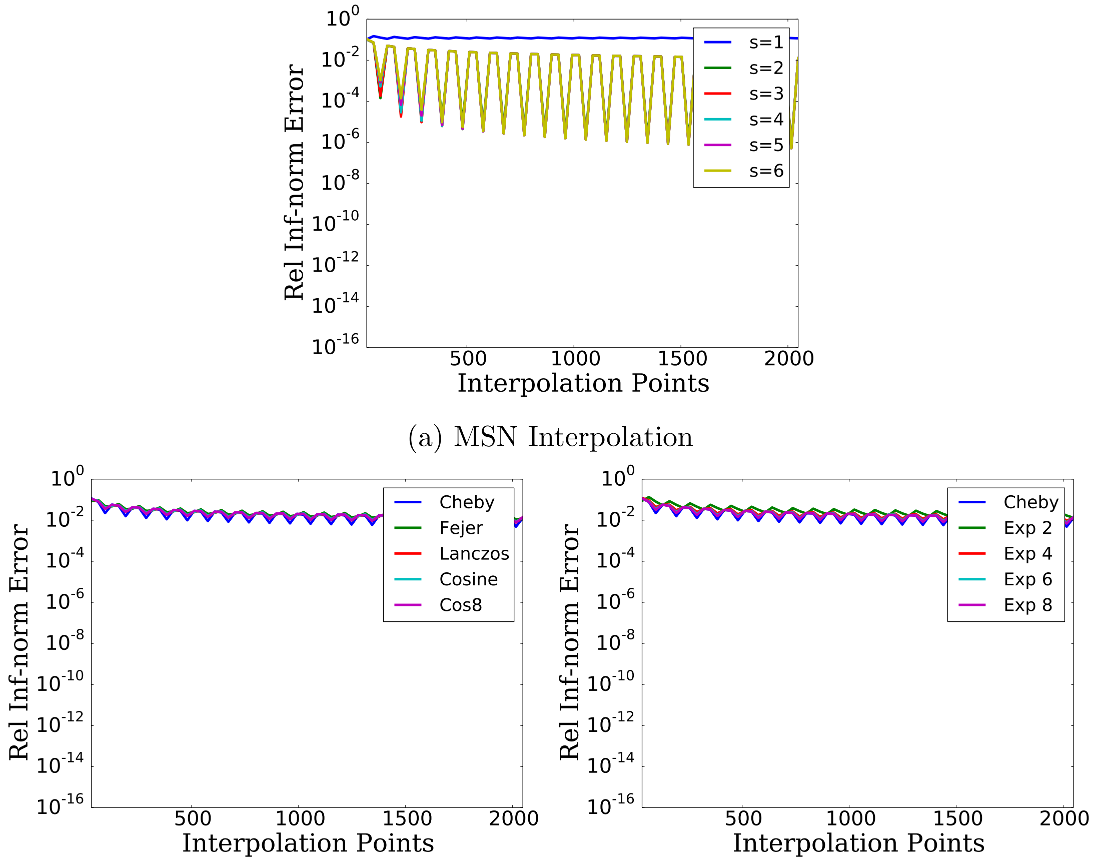


(a) MSN Interpolation

(b) Filters, Plot 1

(c) Filters, Plot 2

Figure 6.13: MSN interpolation and Chebyshev filtering results of the Sharp Function $G(\cdot, 0.5)$ for various s values and filters. We include standard Chebyshev birkhoff interpolant in both filter examples for reference.



(a) MSN Interpolation

(b) Filters, Plot 1

(c) Filters, Plot 2

Figure 6.14: MSN interpolation and Chebyshev filtering results of the Sharp Function $G_2(\cdot, 0.5)$ for various s values and filters. We include standard Chebyshev birkhoff interpolant in both filter examples for reference.

Chapter 7

Interpolation Convergence Proofs

> The proof of Fubini's Theorem is long, tedious, and painful. Let us begin.
>
> Gustavo Ponce, Math 201A, Fall 2013

This chapter works through the convergence proofs for MSN interpolation on Chebyshev nodes. We focus on interpolating up to degree $2n$.

7.1 Main Idea

In this chapter we will compute error bounds for the MSN method. In [16], Chandrasekaran et al. showed the MSN solution was guaranteed to converge to the underlying solution based on a compactness argument by showing the polynomial approximations had bounded derivative in combination with the interpolation conditions and the Arzelà-Ascoli theorem. We take a different route by carefully looking at the linear system and solving for the interpolation coefficients, which is possible due to its structured nature.

Given

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x), \tag{7.1}$$

we know

$$\begin{aligned}||f||_{\infty,[-1,1]} &\leq \sum_{k=0}^{\infty} |a_k| \\ &\equiv ||a||_{p,1}\,, \end{aligned} \tag{7.2}$$

where we obtain equality of norm at $x = 0$ and a_k all the same sign. Naturally, $||a||_{p,1}$ is the 1-norm of the sequence of coefficients. We find

$$\begin{aligned}\sum_{k=N}^{\infty} |a_k| &\leq \sqrt{\sum_{k=N}^{\infty} (k+1)^{-2s}}\, ||a||_s \\ &\leq \frac{1}{\sqrt{2s-1}} \frac{||a||_s}{N^{s-\frac{1}{2}}}. \end{aligned} \tag{7.3}$$

If a_c are the true coefficients chopped to N and $\tilde{a}$ are the coefficients of a polynomial approximation p_N of degree N, then we see

$$\begin{aligned}||f - p_N||_{\infty,[-1,1]} &\leq ||a - \tilde{a}||_{p,1} \\ &= ||a_c - \tilde{a}||_{p,1} + ||a - a_c||_{p,1} \\ &\leq ||a_c - \tilde{a}||_{p,1} + \frac{C_s\, ||a||_s}{N^{s-\frac{1}{2}}}. \end{aligned} \tag{7.4}$$

Because of this, for our MSN approximation p_n, the best asymptotic error we can have is

$$||f - p_n||_{\infty,[-1,1]} \leq \frac{C_s\, ||a||_s}{n^{s-\frac{1}{2}}} \tag{7.5}$$

using this method. Therefore, we will focus on bounding $||a_c - \tilde{a}||_{p,1}$.

7.2 IDCT Coefficients

We now explicitly work out what coefficients are computed using the IDCT. We first define

$$\begin{aligned}
f_k &= f(z_k^n) \\
f_k' &= f'(z_k^n) \\
\bar{f} &= \begin{bmatrix} f(-1) \\ f(1) \end{bmatrix} \\
\widehat{C} &= \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.
\end{aligned} \tag{7.6}$$

It is clear

$$\begin{aligned}
\hat{f}_k &= \left[C^{-1} f\right]_k \\
&= d_k \sum_{j=1}^{n} T_{k-1}(z_j^n) f(z_j^n) \\
&= d_k \sum_{j=1}^{n} T_{k-1}(z_j^n) \left[\sum_{\ell=0}^{\infty} a_\ell T_\ell(z_j^n)\right] \\
&= d_k \sum_{\ell=0}^{\infty} a_\ell \left[\sum_{j=1}^{n} T_{k-1}(z_j^n) T_\ell(z_j^n)\right].
\end{aligned} \tag{7.7}$$

Here, d_k is the appropriate constant. From work in Sec. 3.1, we find

$$\begin{aligned}
\hat{f}_1 &= \sum_{\ell=0}^{\infty} a_{4\ell n} - \sum_{\ell=0}^{\infty} a_{2n+4\ell n} \\
\hat{f}_k &= \sum_{\ell=0}^{\infty} \left[a_{k-1+4\ell n} + a_{4n+1-k+4\ell n}\right] - \sum_{\ell=0}^{\infty} \left[a_{2n+1-k+4\ell n} + a_{2n-1+k+4\ell n}\right] \\
&\quad k \in \{2, \cdots, n\}.
\end{aligned} \tag{7.8}$$

In particular, this becomes

$$\hat{f}_k = a_{k-1} + \varepsilon_k, \tag{7.9}$$

where

$$\begin{aligned}\sum_{k=1}^{n}|\varepsilon_k| &\le \sum_{k=n+1}^{\infty}|a_k| \\ &\le \frac{1}{\sqrt{2s-1}}\frac{||a||_s}{n^{s-\frac{1}{2}}}.\end{aligned} \tag{7.10}$$

Similarly, we also have

$$\begin{aligned}\hat{f}'_k &= \left[UC^{-1}f'\right]_k \\ &= \sum_{\ell=0}^{\infty}\left[(k+4\ell n)\,a_{k+4\ell n} + (2n-k+4\ell n)\,a_{2n-k+4\ell n}\right] \\ &\quad - \sum_{\ell=0}^{\infty}\left[(2n+k+4\ell n)\,a_{2n+k+4\ell n} + (4n-k+4\ell n)\,a_{4n-k+4\ell n}\right] \\ &\quad k \in \{1,\cdots,n-1\} \\ \hat{f}'_n &= \left[UC^{-1}f'\right]_n \\ &= \sum_{\ell=0}^{\infty}(n+4\ell n)\,a_{n+4\ell n} - \sum_{\ell=0}^{\infty}(3n+4\ell n)\,a_{3n+4\ell n}\end{aligned} \tag{7.11}$$

and

$$\begin{aligned}\widehat{C}^{-1}\bar{f} &= \begin{bmatrix}\tilde{f}_1 \\ \tilde{f}_2\end{bmatrix} \\ \tilde{f}_1 &= \sum_{\ell=0}^{\infty}a_{2\ell} \\ \tilde{f}_2 &= \sum_{\ell=0}^{\infty}a_{2\ell+1}.\end{aligned} \tag{7.12}$$

This gives

$$\hat{f}'_k = ka_k + \eta_k \tag{7.13}$$

with

$$
\begin{aligned}
\sum_{k=1}^{n} |\eta_k| &\leq \sum_{k=n+1}^{\infty} k\,|a_k| \\
&\leq \frac{1}{\sqrt{2s-3}} \frac{||a||_s}{n^{s-\frac{3}{2}}}. \qquad (7.14)
\end{aligned}
$$

7.3 Important Summation Bounds

In this section we go over sums which must be bounded above or below. These bounds easily follow from the Mean Value Theorem, Intermediate Value Theorem, and Cauchy-Schwarz inequality.

One sum we see is $\sum_{k=1}^{n} k^p$. Because

$$\sum_{k=1}^{n} k^p = n^{p+1} \sum_{k=1}^{N} \left(\frac{k}{n}\right)^p \frac{1}{n}, \qquad (7.15)$$

we note the right sum approximates $\int_0^1 x^p dx$. When $p \geq 1$, we have the error bound

$$\left| \frac{1}{p+1} - \sum_{k=1}^{n} \left(\frac{k}{n}\right)^p \frac{1}{n} \right| \leq \frac{p}{2n}. \qquad (7.16)$$

For large enough n, then we have the following upper bound:

$$\sum_{k=1}^{n} k^p \leq \frac{2}{p+1} n^{p+1}, \quad n \geq \frac{p(p+1)}{2}. \qquad (7.17)$$

We cannot apply the same idea when $0 < p < 1$ because the derivative of x^p is unbounded on $[0, 1]$. Even so, we also have this error bound:

$$
\begin{aligned}
\sum_{k=1}^{n} k^p &\leq \int_1^{n+1} x^p dx \\
&= \frac{1}{p+1} \left[(n+1)^{p+1} - 1\right] \\
&\leq \frac{2}{p+1} n^{p+1}, \quad 0 < p < 1, \quad n \geq 2. \qquad (7.18)
\end{aligned}
$$

Because x^p is concave down for $p > 0$, we have the lower bound

$$\sum_{k=1}^{n} k^p \geq \frac{1}{p+1} n^{p+1}. \tag{7.19}$$

Similarly, we have can have sums of the form $\sum_{k=1}^{n} k^{-p}$ for $p \geq 0$. It is easy to see for $p > 0$ we have

$$\int_1^n x^{-p} dx \leq \sum_{k=1}^{n} \frac{1}{k^p} \leq 1 + \int_1^n x^{-p} dx. \tag{7.20}$$

For $0 < p < 1$, we see

$$\begin{aligned} \sum_{k=1}^{n} \frac{1}{k^p} &\leq 1 + \frac{n^{1-p}}{1-p} \\ &\leq \frac{2n^{1-p}}{1-p} \end{aligned} \tag{7.21}$$

and

$$\sum_{k=1}^{n} \frac{1}{k^p} \geq \frac{n^{1-p}}{1-p}. \tag{7.22}$$

When $p = 1$, we see

$$\begin{aligned} \sum_{k=1}^{n} \frac{1}{k} &\leq 1 + \ln n \\ &\leq 2 \ln n, \quad (n \geq 3). \end{aligned} \tag{7.23}$$

Additionally, we have the lower bound

$$\sum_{k=1}^{n} \frac{1}{k} \geq \ln n. \tag{7.24}$$

For $p > 1$, we see

$$\sum_{k=1}^{n} \frac{1}{k^p} \leq 1 + \frac{1}{p-1}. \tag{7.25}$$

Additionally, we find

$$\sum_{k=1}^{n} \frac{1}{k^p} \geq \frac{1}{2(p-1)}, \quad n \geq 2^{\frac{1}{p-1}}. \tag{7.26}$$

We accumulate all of the previous bounds, which hold for large enough n:

$$\begin{aligned} \sum_{k=1}^{n} k^p &\leq \begin{cases} \frac{2}{p+1} n^{p+1} & p > -1 \\ 2 \ln n & = -1 \\ \frac{-p}{-p-1} & p < -1 \end{cases} \\ \sum_{k=1}^{n} k^p &\geq \begin{cases} \frac{1}{p+1} n^{p+1} & p > -1 \\ \ln n & = -1 \\ \frac{1}{2(-p-1)} & p < -1 \end{cases} . \end{aligned} \tag{7.27}$$

We also encounter sums of the form

$$\sum_{k=1}^{n} (n+k)^p = n^{p+1} \sum_{k=1}^{n} \left(1 + \frac{k}{n}\right)^p \frac{1}{n}. \tag{7.28}$$

This sum approximates $\int_1^2 x^p dx$. For all p, we see

$$\max_{x \in [1,2]} \left| \frac{d}{dx} x^p \right| \leq |p| \max\left\{1, 2^{p-1}\right\}. \tag{7.29}$$

Thus, we can bound

$$\sum_{k=1}^{n} \left[1 + \frac{k}{n}\right]^p \frac{1}{n} \leq 2 \int_1^2 x^p dx \quad n \geq \frac{|p| \max\{1, 2^{p-1}\}}{\int_1^2 x^p dx} \tag{7.30}$$

and

$$\sum_{k=1}^{n} \left[1 + \frac{k}{n}\right]^p \frac{1}{n} \geq \frac{1}{2} \int_1^2 x^p dx \quad n \geq \frac{|p| \max\{1, 2^{p-1}\}}{\int_1^2 x^p dx}. \tag{7.31}$$

Therefore, we have the following bounds for sufficiently large n:

$$\sum_{k=1}^{n} [n+k]^p \leq \begin{cases} 2\left(\frac{2^{p+1}-1}{p+1}\right) n^{p+1} & p \neq -1 \\ 2 \ln 2 & p = -1 \end{cases}$$

$$\sum_{k=1}^{n} [n+k]^p \geq \begin{cases} \frac{1}{2}\left(\frac{2^{p+1}-1}{p+1}\right) n^{p+1} & p \neq -1 \\ \frac{1}{2}\ln 2 & p = -1 \end{cases}. \tag{7.32}$$

The bounds we just computed are useful, but there are some situations when we have a few extra terms. So, we present similar results. First, we assume $p > 1$ and $\alpha \in \mathbb{N}$ and see

$$\begin{aligned} \sum_{k=1}^{n+\alpha} [n+k]^{-p} &\leq \left\{ \sum_{k=1}^{n} \left[1 + \frac{k}{n}\right]^p \frac{1}{n} + \frac{\alpha}{n} \right\} \frac{1}{n^{p-1}} \\ &\leq \frac{2}{p-1}\frac{1}{n^{p-1}}, \quad n \geq \frac{\max\{p, 2\alpha\}}{\int_1^2 x^{-p} dx}. \end{aligned} \tag{7.33}$$

Naturally, we also have this bound when $\alpha = 0$; in this case, we have simpler constants.

We remember the Riemann Zeta function:

$$\zeta(s) = \sum_{k=1}^{\infty} \frac{1}{n^s}. \tag{7.34}$$

It is clear

$$\int_1^n \frac{1}{x^s} dx \leq \sum_{k=1}^{n} \frac{1}{n^s} \leq 1 + \int_1^n \frac{1}{x^s} dx, \tag{7.35}$$

so that $\zeta(s) < \infty$ when $s > 1$ with the bound

$$\frac{1}{s-1} \leq \zeta(s) \leq 1 + \frac{1}{s-1}, \tag{7.36}$$

and $\zeta(s) \to \infty$ as $s \to 1^+$. Better bounds for $\zeta(s)$ exist and can be found in [53].

Finally, this last bound is useful:

$$\begin{aligned} \sum_{k=0}^{\infty} |a_k| &= \sum_{k=0}^{\infty} (1+k)^{-\sigma} \left[(1+k)^{\sigma} |a_k|\right] \\ &\leq \sqrt{\sum_{k=0}^{\infty} (1+k)^{-2\sigma}} \sqrt{\sum_{k=0}^{\infty} (1+k)^{2\sigma} |a_k|^2} \\ &= \sqrt{\zeta(2\sigma)}\, ||a||_{\sigma}, \end{aligned} \tag{7.37}$$

where we require $\sigma > \frac{1}{2}$. Similarly, we have

$$\sum_{k=1}^{\infty} k^{\alpha} \left| a_k \right| \leq \sqrt{\zeta(2\sigma - 2\alpha)}\, \left|\left| a \right|\right|_{\sigma}, \tag{7.38}$$

where we insist $\sigma > \alpha + \frac{1}{2}$. Similarly, we have

$$\begin{aligned} \sum_{k=n}^{\infty} k^{\alpha} \left| a_k \right| &\leq \sqrt{\sum_{k=n}^{\infty} (k+1)^{-2s+2\alpha}}\, \left|\left| a \right|\right|_{s} \\ &\leq \frac{\left|\left| a \right|\right|_{s}}{\sqrt{2s - 2\alpha - 1}} \frac{1}{n^{s-\alpha-\frac{1}{2}}}, \end{aligned} \tag{7.39}$$

where we must have $s > \alpha + \frac{1}{2}$.

7.4 Sobolev Embedding Theorems and Related Work

This work has focused on functions with bounded Sobolev norm, but because it is usually more convenient to think it terms of continuous and continuously-differentiable functions, we prove some embedding theorems.

We assume

$$g(\theta) = \sum_{k \in \mathbb{Z}} a_k e^{ik\theta}. \tag{7.40}$$

In this case,

$$\begin{aligned} \sum_{k=-N}^{N} \left| a_k \right| &= \sum_{k=-N}^{N} (1 + |k|)^{-s} \left[(1 + |k|)^{s} \left| a_k \right| \right] \\ &\leq \sqrt{\sum_{k=-N}^{N} (1 + |k|)^{-2s}}\, \left|\left| a \right|\right|_{s} \\ &\leq \sqrt{2\zeta(2s)}\, \left|\left| a \right|\right|_{s}. \end{aligned} \tag{7.41}$$

Thus, if $s > \frac{1}{2}$ and $\left|\left| a \right|\right|_{s} < \infty$, this upper bound is finite and holds for all N, so $g(\theta)$ is continuous. Similarly, for an integer $m \geq 1$, we have

$$\begin{aligned}\sum_{\substack{k=-N\\k\neq 0}}^{N} |k|^m \, |a_k| &\leq \sum_{k=-N}^{N} (1+|k|)^m \, |a_k| \\ &\leq \sqrt{2\zeta(2s-2m)} \, ||a||_s \, . \end{aligned} \tag{7.42}$$

Thus, for $s > m + \frac{1}{2}$ and $||a||_s < \infty$, the above bound is finite and holds for all N, showing $g^{(m)}$ is continuous. Therefore, we have shown

$$H_s \subseteq C^m, \quad s > m + \frac{1}{2} \tag{7.43}$$

for integers $m \geq 0$.

In the other direction, if $g \in C^{m,\alpha}$, so that $g^{(m)}$ is α-Hölder continuous with constant L, then with integration by parts we find

$$\begin{aligned} a_k &= \frac{1}{2\pi} \int_{-\pi}^{\pi} g(\theta) e^{-ik\theta} d\theta \\ &= \frac{1}{2\pi \, (ik)^m} \int_{-\pi}^{\pi} g^{(m)}(\theta) e^{-ik\theta} d\theta. \end{aligned} \tag{7.44}$$

Similarly, by changing the limits of integration, we see

$$a_k = -\frac{1}{2\pi \, (ik)^m} \int_{-\pi}^{\pi} g^{(m)} \left(\theta + \frac{\pi}{k}\right) e^{-ik\theta} d\theta, \quad k \neq 0. \tag{7.45}$$

Therefore, we find

$$\begin{aligned} |a_k| &\leq \frac{1}{4\pi \, |k|^m} \int_{-\pi}^{\pi} \left| g(\theta) - g\left(\theta + \frac{\pi}{k}\right) \right| d\theta \\ &\leq \frac{L\pi^\alpha}{2 \, |k|^{m+\alpha}}. \end{aligned} \tag{7.46}$$

Thus, we see $a_k = O(k^{-m-\alpha})$. We let $C = \max\{|a_0| \, , L\pi^\alpha/2\}$. In this case

$$
\begin{aligned}
\sum_{k=-N}^{N} (1+|k|)^{2s} |a_k|^2 &\leq C^2 + C^2 \sum_{\substack{k=-N \\ n\neq 0}}^{N} (1+|k|)^{2s} \frac{1}{|k|^{2m+2\alpha}} \\
&\leq C^2 \left[1 + 2^{2s+1} \sum_{k=1}^{N} |k|^{2s-2m-2\alpha} \right] \\
&\leq C^2 \left[1 + 2^{2s+1} \zeta(2m+2\alpha-2s) \right]. \qquad (7.47)
\end{aligned}
$$

The above bound is holds for all N and is finite so long as $s < m + \alpha + \frac{1}{2}$, which implies $||a||_s < \infty$. Thus, we have shown

$$
C^{m,\alpha} \subseteq H_s, \quad s < m + \alpha + \frac{1}{2}. \qquad (7.48)
$$

This last bound can be slightly improved by looking at functions of bounded variation (see [66, 76]), but we will not pursue the matter here.

7.5 Proof of 1D Interpolation for polynomials of degree $2n$

We begin by computing the error. First, we have the MSN coefficients

$$
\begin{aligned}
\tilde{a} &= D_s^{-1} G^* \begin{bmatrix} Y^{-1}\hat{f} \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} c_1^2 \hat{f}_1 \\ \vdots \\ c_n^2 \hat{f}_n \\ 0 \\ s_n^2 \hat{f}_n \\ \vdots \\ s_1^2 \hat{f}_1 \end{bmatrix}. \qquad (7.49)
\end{aligned}
$$

From Eqs. (4.4) and (4.7) in Sec. 4.2, we recall

$$\begin{aligned}
\tau_k &= \left(\frac{k}{2n+2-k}\right)^s \\
c_k &= \frac{1}{\sqrt{1+\tau^2}} \\
s_k &= \tau_k c_k \\
y_k &= \frac{1}{c_k k^s}.
\end{aligned} \tag{7.50}$$

It is clear

$$\begin{aligned}
c_k &\leq 1 \\
\tau_k &\leq \frac{k^s}{n^s} \\
s_k &\leq \tau_k \\
y_k &\geq k^{-s} \\
y_k &\leq \sqrt{2}\, k^{-s}.
\end{aligned} \tag{7.51}$$

As stated in Sec. 7.1, we only need to focus on computing

$$\begin{aligned}
||a_c - \tilde{a}||_{p,1} &\leq \sum_{k=1}^{n} |a_{k-1} - c_k^2 \hat{f}_k| + |a_n| + \sum_{k=1}^{n} |a_{2n+1-k} - s_k^2 \hat{f}_k| \\
&\leq 2 \sum_{k=1}^{n} s_k^2 \left|a_{k-1}\right| + 2 \sum_{k=n}^{\infty} \left|a_k\right|.
\end{aligned} \tag{7.52}$$

We look at these terms separately. First, we see

$$\begin{aligned}
\sum_{k=1}^{n} s_k^2 \left|a_{k-1}\right| &\leq \frac{1}{n^{2s}} \sum_{k=1}^{n} k^s \cdot k^s \left|a_{k-1}\right| \\
&\leq \frac{1}{n^{2s}} \sqrt{\sum_{k=1}^{n} k^{2s}}\; ||a||_s \\
&\leq \sqrt{\frac{2}{2s+1}} \frac{||a||_s}{n^{s-\frac{1}{2}}}.
\end{aligned} \tag{7.53}$$

Here, the bound in the last equality follows from Eq. (7.27). Similarly,

$$\sum_{k=n}^{\infty} |a_k| \le \frac{1}{\sqrt{2s-1}} \frac{||a||_s}{n^{s-\frac{1}{2}}}, \tag{7.54}$$

which follows from Eq. (7.3). Finally, we have the uniform bound

$$||f - p_n||_{\infty,[-1,1]} \le \frac{C_s}{n^{s-\frac{1}{2}}} ||a||_s \,. \tag{7.55}$$

Thus, $||f - p_n||_{\infty} \to 0$ as $n \to \infty$ when $\frac{1}{2} < s \le \sigma$.

From the above work, we can also compute the condition number of the matrix WD_s^{-1}. It is clear

$$\frac{n^s}{\sqrt{2}} \le y_1 y_n^{-1} \le \sqrt{2}\, n^s, \tag{7.56}$$

so it follows $\kappa_2(L) = \Theta(n^s)$.

The above work suggests that we must choose $s \le \sigma$, but it turns out that this restriction is not necessary. To see this, we will look at Eqs. (7.53) and (7.54) to see where we were too loose in our bounds. We will use the bound

$$(1+k)^{\sigma} |a_k| \le ||a||_{\sigma} \,. \tag{7.57}$$

In Eq. (7.53), we have

$$\begin{aligned}
\sum_{k=1}^{n} s_k^2 |a_{k-1}| &\le \frac{1}{n^{2s}} \sum_{k=1}^{n} k^{2s} |a_{k-1}| \\
&\le \frac{||a||_{\sigma}}{n^{2s}} \sum_{k=1}^{n} k^{2s-\sigma} \\
&\le \frac{2\,||a||_{\sigma}}{2s-\sigma+1} \frac{1}{n^{\sigma-1}}.
\end{aligned} \tag{7.58}$$

Here, we are assuming $2s - \sigma > -1$; otherwise, we have the bound

$$\sum_{k=1}^{n} s_k^2 |a_{k-1}| \le C_{s,\sigma} \frac{\log n}{n^{2s}}. \tag{7.59}$$

Thus, the requirement here is that $\sigma > 1$. Similarly, Eq. (7.54) can be replaced with

$$\sum_{k=n}^{\infty} |a_k| \leq \frac{1}{\sqrt{2\sigma - 1}} \frac{||a||_{\sigma}}{n^{\sigma - \frac{1}{2}}}, \tag{7.60}$$

so that $\sigma > \frac{1}{2}$.

Taken together, we have improved our bound for convergence to

$$||f - p_n||_{\infty,[-1,1]} \leq \frac{C_{s,\sigma}}{n^{\sigma-1}} \, ||a||_{\sigma} \tag{7.61}$$

when $\sigma > 1$, and we can choose $s > \frac{1}{2}$. Whether this can be extended to the case when $\sigma \in (\frac{1}{2}, 1]$ will be looked at in the future. This also shows that the results in future sections likely could be strengthened.

7.6 Proof of 1D Interpolation for polynomials of degree $2n$ with Endpoint Interpolation

In this section, we add interpolation conditions at 1 and -1. After an IDCT and a set of Givens rotations, the same from Sec. 7.5, we obtain the matrix

$$\begin{bmatrix}
y_1 & & & & & & & & & & & & \\
 & y_2 & & & & & & & & & & & \\
 & & \ddots & & & & & & & & & & \\
 & & & y_{n-2} & & & & & & & & & \\
 & & & & y_{n-1} & & & & & & & & \\
 & & & & & y_n & & & & & & & \\
\alpha_1 & & \cdots & & \alpha_{n-1} & & \beta_{n+1} & & \beta_{n-1} & & \cdots & & \beta_1 \\
 & \alpha_2 & & \alpha_{n-2} & & \alpha_n & & \beta_n & & \beta_{n-2} & & \beta_2 &
\end{bmatrix}. \tag{7.62}$$

Here,

$$\begin{aligned}
y_k &= \frac{1}{c_k k^s} \\
\alpha_k &= c_k k^{-s} \left[1 - \tau_k^2\right] \\
\beta_k &= 2 s_k k^{-s} \quad k \in \{1, \cdots, n\}
\end{aligned}$$

$$\beta_{n+1} = (n+1)^{-s}\,. \tag{7.63}$$

We are assuming n is even; similar results will hold when n is odd. From here, we get the easy bound

$$|\beta_k| \leq \frac{2}{n^s}. \tag{7.64}$$

After a pair of Householder reflectors

$$H = \begin{bmatrix} I_n & \\ & I_{n+1} - 2VT^{-1}V^* \end{bmatrix} \tag{7.65}$$

with

$$\begin{aligned} V^* &= \begin{bmatrix} \beta_{n+1} & & \cdots & & \beta_3 & & \beta_1 \\ & \beta_n & & \beta_4 & & \beta_2 & \end{bmatrix} \\ T &= \begin{bmatrix} \ell_1^2 & \\ & \ell_2^2 \end{bmatrix} \\ \ell_1^2 &= \beta_1^2 + \beta_3^2 + \cdots + \beta_{n-1}^2 + \beta_{n+1}^2 \\ \ell_2^2 &= \beta_2^2 + \beta_4^2 + \cdots + \beta_n^2, \end{aligned} \tag{7.66}$$

we have

$$\begin{bmatrix} y_1 & & & & & & & \\ & y_2 & & & & & & \\ & & \ddots & & & & & \\ & & & y_{n-2} & & & & \\ & & & & y_{n-1} & & & \\ & & & & & y_n & & \\ \alpha_1 & & \cdots & & \alpha_{n-1} & & \ell_1 & \\ & \alpha_2 & & \alpha_{n-2} & & \alpha_n & & \ell_2 \end{bmatrix} = \begin{bmatrix} Y & 0 \\ \widehat{A} & \widehat{L} \end{bmatrix}. \tag{7.67}$$

We know

$$\begin{bmatrix} Y & \\ \widehat{A} & \widehat{L} \end{bmatrix}^{-1} \begin{bmatrix} \hat{f} \\ \tilde{f} \end{bmatrix} = \begin{bmatrix} Y^{-1}\hat{f} \\ \widehat{L}^{-1}(\tilde{f} - \widehat{A}Y^{-1}\hat{f}) \end{bmatrix} \tag{7.68}$$

so by setting

$$\begin{aligned}\hat{g} &= \tilde{f} - \widehat{A}Y^{-1}\hat{f} \\ \hat{h} &= \widehat{L}^{-1}\hat{g},\end{aligned} \tag{7.69}$$

we determine the coefficients of the MSN approximation to be

$$\begin{aligned}\tilde{a} &= D_s^{-1}G^*H^* \begin{bmatrix} Y^{-1}\hat{f} \\ \hat{h} \\ 0 \end{bmatrix} \\ &= D_s^{-1}G^* \begin{bmatrix} Y^{-1}\hat{f} \\ 0 \\ 0 \end{bmatrix} + D_s^{-1}G^*H^* \begin{bmatrix} 0 \\ \hat{h} \\ 0 \end{bmatrix} \\ &= \tilde{a}_1 + \tilde{a}_2.\end{aligned} \tag{7.70}$$

Naturally $\tilde{a}_1$ are the coefficients we computed from Sec. 7.5. Because of this, we will compute the bound

$$||a_c - \tilde{a}||_{p,1} \leq ||a_c - \tilde{a}_1||_{p,1} + ||\tilde{a}_2||_{p,1}\,. \tag{7.71}$$

and only focus on $||\tilde{a}_2||_{p,1}$. First, though, we prove some bounds necessary for the rest of the calculations.

We begin by seeking bounds on $\hat{h}_i$ by looking at $\hat{g}_i$. We know

$$\begin{aligned}\hat{g}_1 &= \sum_{\substack{k=1 \\ k \text{ odd}}}^{n} \left(1 - \alpha_k y_k^{-1}\right) a_{k-1} - \sum_{\substack{k=1 \\ k \text{ odd}}}^{n} \alpha_k y_k^{-1} \varepsilon_k + \sum_{k=0}^{\infty} a_{n+2k} \\ \hat{g}_2 &= \sum_{\substack{k=1 \\ k \text{ even}}}^{n} \left(1 - \alpha_k y_k^{-1}\right) a_{k-1} - \sum_{\substack{k=1 \\ k \text{ even}}}^{n} \alpha_k y_k^{-1} \varepsilon_k + \sum_{k=0}^{\infty} a_{n+1+2k}.\end{aligned} \tag{7.72}$$

Now, we have

$$\begin{aligned}\alpha_k y_k^{-1} &= c_k^2 - s_k^2 \\ 1 - \alpha_k y_k^{-1} &= 2s_k^2\end{aligned} \tag{7.73}$$

so that

$$\begin{aligned}
|\hat{g}_1| + |\hat{g}_2| &\leq 2\sum_{k=1}^{n} s_k^2 \left|a_{k-1}\right| + 2\sum_{k=n}^{\infty} \left|a_k\right| \\
&\leq \frac{C_s \left|\left|a\right|\right|_s}{n^{s-\frac{1}{2}}}.
\end{aligned} \tag{7.74}$$

Additionally, we have

$$\begin{aligned}
\ell_1^2, \ell_2^2 &\geq \sum_{k=1}^{\frac{n}{2}} \left(n+2k\right)^{-2s} \\
&\geq \frac{1}{4}\left[\frac{1-2^{-2s+1}}{2s-1}\right]\frac{1}{n^{2s-1}} \\
&\geq \begin{cases} \frac{C_s}{n^{2s-1}} & s > \frac{1}{2} \\ \frac{1}{16sn^{2s-1}} & s \geq 1 \end{cases}.
\end{aligned} \tag{7.75}$$

We give a specific constant when $s \geq 1$ because of the simple form. The second inequality comes from Eq. (7.32). This bound implies

$$\ell_1^{-1}, \ell_2^{-1} \leq \begin{cases} C_s n^{s-\frac{1}{2}} & s > \frac{1}{2} \\ 4\sqrt{s}\, n^{s-\frac{1}{2}} & s \geq 1 \end{cases}, \tag{7.76}$$

which, combined with Eq. (7.74) and the fact

$$\begin{aligned}
\hat{h}_1 &= \ell_1^{-1}\hat{g}_1 \\
\hat{h}_2 &= \ell_2^{-1}\hat{g}_2,
\end{aligned} \tag{7.77}$$

gives us

$$|\hat{h}_1| + |\hat{h}_2| \leq C_s \left|\left|a\right|\right|_s. \tag{7.78}$$

If

$$\tilde{h} = \begin{bmatrix} \hat{h} \\ 0 \end{bmatrix}, \tag{7.79}$$

where we have $n-1$ zeros, then we see

$$\begin{aligned} H^* \begin{bmatrix} 0 \\ \tilde{h} \end{bmatrix} &= \begin{bmatrix} I & \\ & I - 2VT^{-1}V^* \end{bmatrix} \begin{bmatrix} 0 \\ \tilde{h} \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ \tilde{h} \end{bmatrix} - \begin{bmatrix} 0 \\ 2VT^{-1}V^*\tilde{h} \end{bmatrix}. \end{aligned} \tag{7.80}$$

We only need to bound

$$||\tilde{a}_2||_{p,1} \leq \left|\left| D_s^{-1} G^* \begin{bmatrix} 0 \\ \tilde{h} \end{bmatrix} \right|\right|_{p,1} + 2 \left|\left| D_s^{-1} G^* \begin{bmatrix} 0 \\ VT^{-1}V^*\tilde{h} \end{bmatrix} \right|\right|_{p,1}. \tag{7.81}$$

Looking at the first term, we see

$$D_s^{-1} G^* \begin{bmatrix} 0 \\ \tilde{h} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -n^{-s} s_n \hat{h}_2 \\ (n+1)^{-s} \, \hat{h}_1 \\ (n+2)^{-s} \, c_n \hat{h}_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{7.82}$$

so it follows

$$\left|\left| D_s^{-1} G^* \begin{bmatrix} 0 \\ \tilde{h} \end{bmatrix} \right|\right|_{p,1} \leq \frac{2}{n^s} \left[|\hat{h}_1| + |\hat{h}_2| \right]. \tag{7.83}$$

Similarly, we set

$$\begin{aligned} VT^{-1}V^*\tilde{h} &= q \\ &= \begin{bmatrix} q_{n+1} & q_n & \cdots & q_2 & q_1 \end{bmatrix}^*, \end{aligned} \tag{7.84}$$

with

$$q_k = \ell_i^{-2} \beta_i \hat{h}_i \beta_{n+2-k}. \tag{7.85}$$

In the previous equation, i is 1 if k is odd and i is 2 if k is even. From here, given the previous bounds, we see

$$\begin{aligned} |q_k| &\le C_s n^{2s-1} \cdot \frac{2}{n^s} \cdot \widetilde{C}_s \left|\left|a\right|\right|_s \cdot \frac{2}{n^s} \\ &\le \frac{C_s \left|\left|a\right|\right|_s}{n}. \end{aligned} \tag{7.86}$$

This ordering makes the next computation easier:

$$D_s^{-1} G^* \begin{bmatrix} 0 \\ q \end{bmatrix} = \begin{bmatrix} -s_1 1^{-s} q_1 \\ \vdots \\ -s_n n^{-s} q_n \\ (n+1)^{-s} q_{n+1} \\ c_n (n+2)^{-s} q_n \\ \vdots \\ c_1 (2n+1)^{-s} q_1 \end{bmatrix}. \tag{7.87}$$

Naturally, we have

$$\left|\left| D_s^{-1} G^* \begin{bmatrix} 0 \\ q \end{bmatrix} \right|\right|_{p,1} \le \frac{2}{n^s} \sum_{k=1}^{n+1} |q_k|. \tag{7.88}$$

Now, from Eq. (7.86), it is easy to see

$$\sum_{k=1}^{n+1} |q_k| \le C_s \left|\left|a\right|\right|_s. \tag{7.89}$$

All of these results we combine together to arrive at

$$\left|\left|\tilde{a}_2\right|\right|_{p,1} \le \frac{C_s \left|\left|a\right|\right|_s}{n^s}. \tag{7.90}$$

All of this implies

$$\left|\left|f - p_n\right|\right|_{\infty,[-1,1]} \le \frac{C_s \left|\left|a\right|\right|_s}{n^{s-\frac{1}{2}}} \tag{7.91}$$

where $\frac{1}{2} < s \le \sigma$.

From the work above, we can easily compute bounds on the condition number $\kappa(L)$. We assume $\begin{bmatrix} a & b \end{bmatrix}^*$ is a unit vector. First, we see

$$\begin{aligned} \left|\left| \begin{bmatrix} Y & \\ \widehat{A} & \widehat{L} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right|\right|_2^2 &= ||Ya||_2^2 + ||\widehat{A}a + \widehat{L}b||_2^2 \\ &\leq 2 + \left|\left| \begin{bmatrix} 2\left\{1^{-s} + 3^{-s} + \cdots + (n-1)^{-s}\right\} + \frac{4}{n^{1/2}} \\ 2\left\{2^{-s} + 4^{-s} + \cdots + n^{-s}\right\} + \frac{4}{n^{1/2}} \end{bmatrix} \right|\right|_2^2 \\ &\leq \begin{cases} 64\left[\ln n\right]^2 & s \geq 1 \\ 256 & s \geq 2 \end{cases}. \end{aligned} \tag{7.92}$$

In the above inequality, we note two different bounds on norm because it is independent of n; we note the logarithmic bound holds for large n. We note $||L||_2 \geq 1$. Additionally,

$$\begin{aligned} \left|\left| \begin{bmatrix} Y & \\ \widehat{A} & \widehat{L} \end{bmatrix}^{-1} \begin{bmatrix} a \\ b \end{bmatrix} \right|\right|_2^2 &= ||Y^{-1}a||_2^2 + ||\widehat{L}^{-1}(b - \widehat{A}Y^{-1}a)||_2^2 \\ &\leq 2n^{2s} + 128sn^{2s+1} \\ &\leq 256sn^{2s+1}. \end{aligned} \tag{7.93}$$

We find $||L^{-1}||_2 \geq n^s/2$. It follows

$$\frac{1}{2}n^s \leq \kappa(L) \leq \begin{cases} 128\sqrt{s}\,n^{s+\frac{1}{2}}\ln n & s \geq 1 \\ 256\sqrt{s}\,n^{s+\frac{1}{2}} & s \geq 2 \end{cases}. \tag{7.94}$$

Thus, we have $\kappa(L) = \Omega(n^s)$, $\kappa(L) = O_s(n^{s+\frac{1}{2}}\ln n)$ for $s \geq 1$, and $\kappa(L) = O_s(n^{s+\frac{1}{2}})$ for $s \geq 2$.

7.7 Proof of 1D Birkhoff Interpolation for polynomials of degree $2n$ with Point Interpolation

In this section, we interpolate derivative values on the Chebyshev nodes with the function value at 0. For ease of notation, *in this section only* we will label $f(0) = \hat{f}_1 = a_0$ and label

$$\hat{f}' = \begin{bmatrix} \hat{f}_2' & \cdots & \hat{f}_{n+1}' \end{bmatrix}^*. \tag{7.95}$$

The reason will become clear when we see the simplified nature of the interpolation coefficients. Now, from Eq. (7.13), we know

$$\hat{f}_k' = (k-1)\, a_{k-1} + \eta_{k-1}, \tag{7.96}$$

We find the MSN solution coefficients to be

$$\begin{aligned}
\tilde{a} &= D_s^{-1} G^* \begin{bmatrix} 1 & & \\ & \Gamma & 0 \end{bmatrix}^+ \begin{bmatrix} \hat{f}_1 \\ \hat{f}' \end{bmatrix} \\
&= \begin{bmatrix} \hat{f}_1 \\ \frac{c_2^2 \hat{f}_2'}{2-1} \\ \vdots \\ \frac{c_n^2 \hat{f}_n'}{n-1} \\ \frac{\hat{f}_{n+1}'}{n} \\ \frac{s_n^2 \hat{f}_n'}{n-1} \\ \vdots \\ \frac{s_2^2 \hat{f}_2'}{2-1} \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} a_0 \\ c_2^2 a_1 \\ \vdots \\ c_n^2 a_{n-1} \\ a_n \\ s_n^2 a_{n-1} \\ \vdots \\ s_2^2 a_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{c_2^2}{2-1}\eta_1 \\ \vdots \\ \frac{c_n^2}{n-1}\eta_{n-1} \\ \frac{1}{n}\eta_n \\ \frac{s_n^2}{n-1}\eta_{n-1} \\ \vdots \\ \frac{s_2^2}{2-1}\eta_1 \\ 0 \end{bmatrix} \\
&= \tilde{a}_1 + \tilde{a}_2
\end{aligned} \tag{7.97}$$

Here, $\tilde{a}_1$ includes the main coefficient terms and $\tilde{a}_2$ includes the η_k. We have

$$\begin{aligned}
\tau_k &= -\left(\frac{2n+1-k}{k-1}\right)\left(\frac{k}{2n+2-k}\right)^s \\
c_k &= \frac{1}{\sqrt{1+\tau_k^2}} \\
s_k &= \tau_k c_k \\
\gamma_k &= (k-1)\,k^{-s}\sqrt{1+\left(\frac{2n+1-k}{k-1}\right)^2\left(\frac{k}{2n+2-k}\right)^{2s}} \quad k \in \{2,\cdots,n\} \\
\gamma_{n+1} &= \frac{n}{(n+1)^s}
\end{aligned} \tag{7.98}$$

from Eqs. (4.41) and (4.43) in Sec. 4.8.

Therefore, we see we have

$$\begin{aligned}
||a_c - \tilde{a}||_{p,1} &\leq ||a_c - \tilde{a}_1||_{p,1} + ||\tilde{a}_2||_{p,1} \\
&\leq 2\sum_{k=1}^{n} s_k^2\,|a_{k-1}| + 2\sum_{k=n+1}^{\infty} k\,|a_k| \\
&\leq 2\sqrt{\frac{2}{2s+1}}\frac{||a||_s}{n^{s-\frac{1}{2}}} + \frac{1}{\sqrt{2s-3}}\frac{||a||_s}{n^{s-\frac{3}{2}}},
\end{aligned} \tag{7.99}$$

where we use bounds from Eqs. (7.27) and (7.39). Thus, we see

$$||f-p_n||_{\infty,[-1,1]} \leq \frac{C_s\,||a||_s}{n^{s-\frac{3}{2}}}, \tag{7.100}$$

with convergence when $\frac{3}{2} < s \leq \sigma$.

We now focus on bounding the condition number $\kappa(L)$. We see

$$\left|\left|\begin{bmatrix}1 & \\ & \Gamma\end{bmatrix}\right|\right|_2 = 1 \tag{7.101}$$

under the restriction $s > \frac{3}{2}$, which is required for convergence. Similarly, we see

$$\begin{aligned}
\left|\left|\begin{bmatrix}1 & \\ & \Gamma\end{bmatrix}^{-1}\right|\right|_2 &= \gamma_{n+1}^{-1} \\
&= n^{s-1}\left(1+\frac{1}{n}\right)^s
\end{aligned}$$

$$\leq 2n^{s-1}, \quad n \geq \left[2^{1/s} - 1\right]^{-1}, \tag{7.102}$$

while we also have $\gamma_{n+1}^{-1} \geq n^{s-1}$. Therefore, $\kappa(L) = \Theta(n^{s-1})$.

7.8 Proof of 1D Full Birkhoff Interpolation for polynomials of degree $2n$

We begin by computing the error. Now, we know the coefficients are

$$\begin{aligned}
\tilde{a} &= D_s^{-1} G^* \begin{bmatrix} Y & & \\ A & B & 0 \end{bmatrix}^+ \begin{bmatrix} \hat{f} \\ E\hat{f}' \end{bmatrix} \\
&= D_s^{-1} G^* \begin{bmatrix} Y^{-1}\hat{f} \\ B^{-1}(E\hat{f}' - AY^{-1}\hat{f}) \\ 0 \end{bmatrix} \\
&= D_s^{-1} G^* \begin{bmatrix} Y^{-1}\hat{f} \\ 0 \\ 0 \end{bmatrix} + D_s^{-1} G^* \begin{bmatrix} 0 \\ B^{-1}(E\hat{f}' - AY^{-1}\hat{f}) \\ 0 \end{bmatrix} \\
&= \tilde{a}_1 + \tilde{a}_2.
\end{aligned} \tag{7.103}$$

Y and G are the same as from Sec. 7.5. Additionally, we have

$$\begin{aligned}
A &= \begin{bmatrix} & & & & 0 \\ & & & & \alpha_n \\ & & & \iddots & \\ & & \alpha_3 & & \\ 0 & \alpha_2 & & & \end{bmatrix} \\
B &= \operatorname{diag}\left(\beta_{n+1}, \beta_n, \cdots, \beta_2\right)
\end{aligned} \tag{7.104}$$

with

$$\begin{aligned}
y_k &= \frac{1}{c_k k^s} \\
\alpha_k &= (k-1)k^{-s} c_k \left\{1 - \left(\frac{2n+1-k}{k-1}\right)\tau_k^2\right\}
\end{aligned}$$

$$\begin{aligned}\beta_k &= 2nk^{-s}s_k \quad k \in \{2, \cdots, n\} \\ \beta_{n+1} &= n\,(n+1)^{-s}\,. \end{aligned} \tag{7.105}$$

Now, $\tilde{a}_1$ is the exact term we obtain from Eq. (7.49). Thus, we focus on the second term because

$$||a_c - \tilde{a}||_{p,1} \le ||a_c - \tilde{a}_1||_{p,1} + ||\tilde{a}_2||_{p,1}\,. \tag{7.106}$$

We set

$$\begin{aligned} q &= \begin{bmatrix} B^{-1}(E\hat{f}' - AY^{-1}\hat{f}) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} q_{n+1} & q_n & \cdots & q_2 & q_1 \end{bmatrix}^* \end{aligned} \tag{7.107}$$

with

$$\begin{aligned} q_1 &= 0 \\ q_k &= \hat{f}'_{k-1} - \alpha_k y_k^{-1}\hat{f}_k \\ &= \left[1 - \alpha_k y_k^{-1}\right] a_{k-1} + \eta_{k-1} - \alpha_k y_k^{-1}\varepsilon_k \quad k \in \{2, \cdots, n\} \\ q_{n+1} &= \hat{f}'_n. \end{aligned} \tag{7.108}$$

The reason for this unusual ordering is because it makes the notation easier. Then, the second term in Eq. (7.103) becomes

$$\begin{aligned} \tilde{a}_2 &= D_s^{-1}G^* \begin{bmatrix} 0 \\ q \end{bmatrix} \\ &= \begin{bmatrix} -s_1 1^{-s} q_1 \\ \vdots \\ -s_n n^{-s} q_n \\ (n+1)^{-s} q_{n+1} \\ c_n (n+2)^{-s} q_n \\ \vdots \\ c_1 (2n+1)^{-s} q_1 \end{bmatrix}. \end{aligned} \tag{7.109}$$

Now, we see

$$\begin{aligned} \left|-s_k k^{-s}\right| &\leq n^{-s} \\ \left|c_k \left(2n+2-k\right)^{-s}\right| &\leq n^{-s} \end{aligned} \tag{7.110}$$

and

$$\begin{aligned} \left|1-\alpha_k y_k^{-1}\right| &= \left|2-k+2ns_k^2\right| \\ &\leq k+2ns_k^2 \\ \left|\alpha_k y_k^{-1}\right| &\leq k+2ns_k^2, \end{aligned} \tag{7.111}$$

implying

$$\left|-s_k k^{-s} q_k\right| \leq \left[n^{-s}k+2n^{1-s}s_k^2\right]\left|a_{k-1}\right| + n^{-s}\left|\eta_{k-1}\right| + \left[n^{-s}k+2n^{1-s}s_k^2\right]\left|\varepsilon_k\right|. \tag{7.112}$$

We also have the bound

$$\begin{aligned} \left|c_k\left(2n+2-k^{-s}q_k\right)\right| \leq \left[n^{-s}k+2n^{1-s}s_k^2\right]\left|a_{k-1}\right| + n^{-s}\left|\eta_{k-1}\right| + \left[n^{-s}k+2n^{1-s}s_k^2\right]\left|\varepsilon_k\right|, \\ k \in \left\{1,\cdots,n\right\}. \end{aligned} \tag{7.113}$$

When $k=n+1$, we have the easier bound

$$\left|\left(n+1\right)^{-s} q_{n+1}\right| \leq n^{-s}\left[\left|a_n\right|+\left|\eta_n\right|\right]. \tag{7.114}$$

All of the above work shows can be combined to show

$$\begin{aligned} \sum_{k=1}^{n}\left|-s_k k^{-s} q_k\right| \leq \frac{\sqrt{\zeta(2s-2)}\,||a||_s}{n^s} + 2\sqrt{\frac{2}{2s+1}}\frac{||a||_s}{n^{2s-\frac{3}{2}}} + \frac{1}{\sqrt{2s-1}}\frac{||a||_s}{n^{2s-\frac{1}{2}}} \\ + \frac{1}{\sqrt{2s-1}}\frac{||a||_s}{n^{2s-\frac{3}{2}}} + \frac{2}{\sqrt{2s-1}}\frac{||a||_s}{n^{2s-\frac{3}{2}}}. \end{aligned} \tag{7.115}$$

The same bound holds for $\sum_{k=1}^{n+1}\left|c_k\left(2n+2-k\right)^{-s}q_k\right|$. All of these terms are $O(n^{-s+\frac{1}{2}})$

and, naturally, we have the restriction $s > \frac{3}{2}$.

Taken altogether, we see

$$\begin{aligned} ||a_c - \tilde{a}||_{p,1} &\leq ||a_c - \tilde{a}_1||_{p,1} + ||\tilde{a}_2||_{p,1} \\ &\leq \frac{C_s \, ||a||_s}{n^{s-\frac{1}{2}}}, \end{aligned} \tag{7.116}$$

with convergence when $\frac{3}{2} < s \leq \sigma$.

From the above work, we can also compute the condition number. First, we see

$$\begin{aligned} \left|\left| \begin{bmatrix} Y & \\ A & B \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right|\right|_2^2 &= ||Ya||_2^2 + ||Aa + Yb||_2^2 \\ &\leq 25. \end{aligned} \tag{7.117}$$

We also note we have $||L||_2^2 \geq \frac{1}{2}$. We now look at the greater challenge of bounding the inverse, and we use similar methods to those above:

$$\begin{aligned} \left|\left| \begin{bmatrix} Y & \\ A & B \end{bmatrix}^{-1} \begin{bmatrix} a \\ b \end{bmatrix} \right|\right|_2^2 &= ||Y^{-1}a||_2^2 + ||B^{-1}(b - AY^{-1}a)||_2^2 \\ &\leq 2n^{2s} + \frac{2^7}{2s+1} n^{2s+1} \\ &\leq 2^8 n^{2s+1}. \end{aligned} \tag{7.118}$$

Due to Y, we also have $||L^{-1}||_2^2 \geq n^{2s}/2$. Therefore, we have

$$\frac{1}{2} n^s \leq \kappa(L) \leq 2^4 n^{s+\frac{1}{2}}. \tag{7.119}$$

Therefore, we have shown $\kappa(L) = \Omega(n^s)$ and $\kappa(L) = O(n^{s+\frac{1}{2}})$.

7.9 Proof of 1D Interpolation for polynomials of degree $2Ln$

We let p_n be the MSN polynomial approximation of degree $2Ln$. Now, as before, we can bound the error

$$||f - p_n||_\infty \le ||a - a_c||_{p,1} + ||a_c - \tilde{a}||_{p,1}\,, \tag{7.120}$$

where in this instance a_c is the true coefficients chopped to degree $2Ln$ and $\tilde{a}$ are the coefficients of the MSN polynomial. As before,

$$||a - a_c||_{p,1} \le \frac{1}{\sqrt{2s-1}} \frac{||a||_s}{[2Ln+1]^{s-\frac{1}{2}}}, \tag{7.121}$$

and so we only need to bound $||a_c - \tilde{a}||_{p,1}$.

From the normal equations of the MSN solution, we know

$$\tilde{a} = D_s^{-2} W^* \left(W D_s^{-2} W^*\right)^{-1} \hat{f} \tag{7.122}$$

where $\hat{f} = C^{-1} f$ and

$$Y^2 = W D_s^{-2} W^* \tag{7.123}$$

is a diagonal matrix with entries

$$\begin{aligned} y_1^2 &= 1^{-2s} + [2n+1]^{-2s} + [4n+1]^{-2s} + \cdots + [2Ln+1]^{-2s} \\ &= 1^{-2s} + \nu_1 \\ y_k^2 &= k^{-2s} + [2n+2-k]^{-2s} + [2n+k]^{-2s} + [4n+2-k]^{-2s} \\ &\quad + \cdots + [2(L-1)n+k]^{-2s} + [2Ln+2-k]^{-2s} \\ &= k^{-2s} + \nu_k, \quad k \in \{2, \cdots, n\}\,. \end{aligned} \tag{7.124}$$

The $\{\nu_i\}$ keep track of the small terms in the sums. From here, we see

$$\tilde{a}_{2\ell n+1-k} = (-1)^\ell \left[2\ell n + 2 - k\right]^{-2s} y_k^{-2s} \hat{f}_k, \quad \ell \in \{1, 2, \cdots, L\}$$

$$\tilde{a}_{2\ell n+k-1} = (-1)^{\ell} \left[2\ell n + k - 2\right]^{-2s} y_k^{-2s} \hat{f}_k, \quad \ell \in \{0, 1, \cdots, L-1\}. \tag{7.125}$$

Using this, we see

$$\begin{aligned} |a_{k-1} - \tilde{a}_{k-1}| &\leq |a_{k-1}| \left[\frac{k^{2s}\nu_k}{1 + k^{2s} n u_k}\right] + \frac{1}{1 + k^{2s}\nu_k} |\varepsilon_k| \\ &\leq k^{2s}\nu_k |a_{k-1}| + |\varepsilon_k|, \quad k \in \{1, \cdots, n\}. \end{aligned} \tag{7.126}$$

This fact allows us to bound the first few terms, which have been critical to prove convergence. It will be important for us to bound $k^{2s}\nu_k$, so we will look at those terms now. First, we see

$$\begin{aligned} 1^{-2s}\nu_1 &= \nu_1 \\ &= (2n+1)^{-2s} + (4n+1)^{-2s} + \cdots + (2Ln+1)^{-2s} \\ &\leq \sum_{\ell=1}^{\infty} (2\ell n + 1)^{-2s} \\ &\leq 4^{-s} n^{-2s} \zeta(2s) \end{aligned} \tag{7.127}$$

and

$$\begin{aligned} k^{2s}\nu_k &\leq k^{2s} \left[\sum_{\ell=1}^{\infty} (2\ell n + 2 - k)^{-2s} + \sum_{\ell=1}^{\infty} (2\ell n + k)^{-2s}\right] \\ &\leq \left(\frac{k}{n}\right)^{2s} \left[1 + 4^{-s}\right] \zeta(2s), \quad k \in \{2, \cdots, n\}. \end{aligned} \tag{7.128}$$

We now have

$$||a_c - \tilde{a}||_{p,1} \leq \sum_{k=1}^{n} k^{2s}\nu_k |a_{k-1}| + \sum_{k=n}^{2Ln} \left[|a_k| + |\tilde{a}_k|\right]. \tag{7.129}$$

The first term has the bound

$$\begin{aligned}\sum_{k=1}^{n} k^{2s}\nu_k\left|a_{k-1}\right| &\leq n^{-2s}\left[1+4^{-s}\right]\zeta(2s)\sum_{k=1}^{n} k^{2s}\left|a_{k-1}\right| \\ &\leq \left[1+4^{-s}\right]\zeta(2s)\sqrt{\frac{2}{2s+1}}\frac{\left|\left|a\right|\right|_s}{n^{s-\frac{1}{2}}}. \qquad (7.130)\end{aligned}$$

The second term has two sums. The first one is easy:

$$\begin{aligned}\sum_{k=n}^{2Ln}\left|a_k\right| &\leq \sum_{k=n}^{\infty}\left|a_k\right| \\ &\leq \frac{1}{\sqrt{2s-1}}\frac{\left|\left|a\right|\right|_s}{n^{s-\frac{1}{2}}}. \qquad (7.131)\end{aligned}$$

The second term is more involved. We see

$$\begin{aligned}\sum_{k=n}^{2Ln}\left|\tilde{a}_k\right| &= \sum_{k=1}^{n}\left\{(2Ln+2-k)^{-2s}+\sum_{\ell=1}^{L-1}\left[2\ell n+2-k\right]^{-2s}+\left[2\ell n+k\right]^{-2s}\right\}y_k^{-2}|\hat{f}_k| \\ &\leq 2Ln^{-2s}\sum_{k=1}^{n}y_k^{-2}|\hat{f}_k| \\ &\leq 2Ln^{-2s}\left[\sum_{k=1}^{n}k^{2s}\left|a_{k-1}\right|+n^{2s}\sum_{k=1}^{n}\left|\varepsilon_k\right|\right] \\ &\leq C(s,L)\frac{\left|\left|a\right|\right|_s}{n^{s-\frac{1}{2}}}. \qquad (7.132)\end{aligned}$$

Taken together, these sums imply

$$\left|\left|f-p_n\right|\right|_{\infty,[-1,1]} \leq C(s,L)\frac{\left|\left|a\right|\right|_s}{n^{s-\frac{1}{2}}}, \qquad (7.133)$$

with convergence when $\frac{1}{2} < s \leq \sigma$.

7.10 Proof of Norm Convergence for 1D Interpolation of polynomials of degree $2n$

In this section we determine when our MSN approximation controlling the sth derivative will converge to the true solution in $||\cdot||_\tau$ when $||a||_\sigma < \infty$. We will be assuming throughout that $\sigma > \frac{1}{2}$ and $s > \frac{1}{2}$ as this simplifies some of the assumptions. We will discuss the situation when $\sigma \in [0, \frac{1}{2}]$ at the end of this section.

If $\tilde{a}$ are the coefficients of the $2n$ MSN solution controlling the s derivative, then we see

$$\begin{aligned} ||a - \tilde{a}||_\tau^2 &= ||a - a_c||_\tau^2 + ||\tilde{a} - a_c||_\tau^2 \\ &= \sum_{k=2n+1}^{\infty} (1+k)^{2\tau} a_k^2 + ||\tilde{a} - a_c||_\tau^2 \\ &\le ||\tilde{a} - a_c||_\tau^2 + \frac{||a||_\sigma^2}{2\sigma - 2\tau - 1} \frac{1}{(2n)^{2\sigma - 2\tau - 1}}. \end{aligned} \tag{7.134}$$

As before, a_c is the true solution chopped to degree $2n$. Thus, we only need to look at bounding $||\tilde{a} - a_c||_\tau$. Furthermore, the above inequality shows we must have $\tau < \sigma - \frac{1}{2}$.

Before getting into the main details, we will need to use the bound

$$\begin{aligned} \sum_{k=1}^{n} |\varepsilon_k|^2 &\le \left(\max_j |\varepsilon_j| \right) \sum_{k=1}^{n} |\varepsilon_k| \\ &\le \left[\sum_{k=1}^{n} |\varepsilon_k| \right]^2 \\ &\le \frac{||a||_\sigma^2}{2\sigma - 1} \frac{1}{n^{2\sigma - 1}}, \end{aligned} \tag{7.135}$$

where ε_k is defined in Eq. (7.9) and we have the useful bound

$$\sum_{k=1}^{n} |\varepsilon_k| \le \frac{1}{\sqrt{2\sigma - 1}} \frac{||a||_\sigma}{n^{\sigma - \frac{1}{2}}} \tag{7.136}$$

from Eq. (7.10). This bound is just the fact $||x||_2^2 \leq ||x||_1 \, ||x||_\infty$. It turns out this bound is the only time that we must have $\sigma > \frac{1}{2}$.

We know

$$\begin{aligned} ||\tilde{a} - a_c||_\tau^2 = &\sum_{k=1}^{n} k^{2\tau}(a_{k-1} - c_k^2 \hat{f}_k)^2 + (1+n)^{2\tau} \, a_n^2 \\ &+ \sum_{k=n+1}^{2n} (1+k)^{2\tau} \, (a_k - s_{2n+1-k}^2 \hat{f}_{2n+1-k})^2. \end{aligned} \tag{7.137}$$

We will begin by looking at the first sum and then the remaining terms.

From our previous work, we know

$$\left(a_{k-1} - c_k^2 \hat{f}_k\right)^2 \leq 2\left(s_k^4 a_{k-1}^2 + |\varepsilon_k|^2\right). \tag{7.138}$$

The following summation bound is clear:

$$\begin{aligned} \sum_{k=1}^{n} k^{2\tau} s_k^4 a_{k-1}^2 &\leq \frac{||a||_\sigma^2}{n^{4s}} \sum_{k=1}^{n} k^{2\tau+4s-2\sigma} \\ &\leq \frac{2\,||a||_\sigma^2}{2\tau+4s-2\sigma+1} \frac{1}{n^{2\sigma-2\tau-1}}. \end{aligned} \tag{7.139}$$

The sum that we bound has the form as given in Eq. (7.27); we are only reproducing the bound when $2\tau + 4s - 2\sigma > -1$ for simplicity, as the other cases are similar and there is still decay in n: $O(n^{-4s} \ln n)$ or $O(n^{-4s})$. Additionally, We also have the bound

$$\begin{aligned} \sum_{k=1}^{n} k^{2\tau} \, |\varepsilon_k|^2 &\leq n^{2\tau} \sum_{k=1}^{n} |\varepsilon_k|^2 \\ &\leq \frac{||a||_\sigma^2}{2\sigma - 1} \frac{1}{n^{2\sigma-2\tau-1}}. \end{aligned} \tag{7.140}$$

We now look at

$$\begin{aligned}(1+n)^{2\tau}a_n^2 &+ \sum_{k=n+1}^{2n}(1+k)^{2\tau}\left(a_k - s_{2n+1-k}^2\hat{f}_{2n+1-k}\right)^2 \\ &\le 2\sum_{k=n}^{2n}(1+k)^{2\tau}a_k^2 + 2\sum_{k=1}^{n}(2n+2-k)^{2\tau}s_k^4\hat{f}_k^2. \qquad (7.141)\end{aligned}$$

It is clear that the first sum reduces to

$$\begin{aligned}\sum_{k=n}^{2n}(1+k)^{2\tau}a_k^2 &\le ||a||_\sigma^2\sum_{k=1}^{n+1}[n+k]^{2\tau-2\sigma} \\ &\le \frac{2\,||a||_\sigma^2}{2\sigma-2\tau-1}\frac{1}{n^{2\sigma-2\tau-1}}, \qquad (7.142)\end{aligned}$$

where we are using the bound form Eq. (7.33). In the second sum, we see

$$\begin{aligned}\sum_{k=1}^{n}(2n+2-k)^{2\tau}s_k^4\hat{f}_k^2 &\le (2n+1)^{2\tau}\sum_{k=1}^{n}s_k^4\hat{f}_k^2 \\ &\le 2\,(2n+1)^{2\tau}\sum_{k=1}^{n}s_k^4\left(a_k^2+|\varepsilon_k|^2\right). \qquad (7.143)\end{aligned}$$

The first portion of the sum simplifies as follows:

$$\begin{aligned}\sum_{k=1}^{n}s_k^4a_k^2 &\le \frac{||a||_\sigma^2}{n^{4s}}\sum_{k=1}^{n}k^{4s-2\sigma} \\ &\le \frac{||a||_\sigma^2}{4s-2\sigma+1}\frac{1}{n^{2\sigma-1}}. \qquad (7.144)\end{aligned}$$

Similarly, we have

$$\begin{aligned}\sum_{k=1}^{n}s_k^4\,|\varepsilon_k|^2 &\le \sum_{k=1}^{n}|\varepsilon_k|^2 \\ &\le \frac{||a||_\sigma^2}{2\sigma-1}\frac{1}{n^{2\sigma-1}}. \qquad (7.145)\end{aligned}$$

If we combine these inequalities, we find

$$\sum_{k=1}^{n} (2n+2-k)^{2\tau} s_k^4 \hat{f}_k^2 \leq \frac{C_{\tau,\sigma,s} \left|\left|a\right|\right|_\sigma^2}{n^{2\sigma-2\tau-1}}. \tag{7.146}$$

Taken together, we have

$$\begin{aligned} \left|\left|a-\tilde{a}\right|\right|_\tau^2 &= \left|\left|a-a_c\right|\right|_\tau^2 + \left|\left|\tilde{a}-a_c\right|\right|_\tau^2 \\ &\leq \frac{C_{s,\sigma,\tau} \left|\left|a\right|\right|_\sigma^2}{n^{2\sigma-2\tau-1}}, \end{aligned} \tag{7.147}$$

with the restrictions $\sigma > \frac{1}{2}$ and $\tau < \sigma - \frac{1}{2}$.

As we can see, the only instance where we require $\sigma > \frac{1}{2}$ occurs in bounding $\sum_{k=1}^{n} \left|\varepsilon_k\right|^2$. When proving convergence in the original MSN interpolation papers [16, 18], dyadic sums were used in order to determine a sufficient interpolation degree to ensure bounded derivative norm. The interpolation examples in Chapter 6 show that even when the function is discontinuous, we obtain convergence to the underlying solution where the function in continuous. This suggests that the convergence in norm results may be extended to the case when $\sigma \in \left[0, \frac{1}{2}\right]$ (e.g. when the function is integrable but not continuous) but the proof of this may require careful analysis due to the possibly non-absolutely converging terms ε_k; this will be looked at in future work. It is well-known conditionally convergent series are challenging to work with [45, 57].

7.11 Extension to Higher Dimensions

Throughout this chapter, we have been able to prove convergence to the underlying solution because the inherent structure of the underlying system, allowing us to compare the MSN approximation to the true solution. While this could be extended to higher dimensions, the extensions would be tedious and not enlightening. General MSN theory from [16, 18] shows that for any point distribution with point separation η, it is possible to guarantee that we can choose the interpolation degree $O(\eta^{-1})$; this shows the importance of our convergence results for degree $2Ln$. The exact situation where this would be needed, though, is not known and may only occur for functions in H_s for $s \in \left[0, \frac{1}{2}\right]$. Our examples in Chapter 6 show that polynomials of degree $2n$ are usually sufficient. Future work will investigate methods which would allow us to prove convergence without resorting to carefully looking at the linear system yet do not require the technical machinery used in the original MSN proofs of [16, 18].

Chapter 8

Fast Algorithms for ODEs

> The best Bell number is $S_7 = 877$.
>
> CHG

In this chapter we shift from interpolation problems to differential equations, the main goal of the dissertation. Interpolation problems look at function and derivative constraints at points of interest. Naturally, linear differential equations involve linear combinations of function and derivative values.

It is well-known that all differential equations can be rewritten to into first-order system [3, 4]. While this is not always done, especially for finite difference [46] or finite element methods [12], we do in this case. This was also done when solving partial differential equations with MSN [20].

8.1 General Setup for Linear ODEs

The general form for our linear ODE is given by

$$\begin{aligned} A(x)u'(x) + B(x)u(x) &= F(x), \quad x \in [-1, 1] \\ Lu &= G. \end{aligned} \tag{8.1}$$

Here, we are looking for a solution $u : [-1,1] \to \mathbb{R}^m$ with continuous $A, B : [-1,1] \to \mathbb{R}^{m \times m}$, continuous $F : [-1,1] \to \mathbb{R}^m$, and a linear operator L. Throughout this chapter we will assume Eq. (8.1) has a unique solution.

By making some assumptions on A and B, we are able to arrive at fast algorithms. We will first look at the scalar case before moving toward the general case of systems of linear ODEs.

8.2 Constant-Coefficient Scalar ODE

Here, we begin with the simplest ODE:

$$\alpha u'(x) + \beta u(x) = f(x), \quad x \in [-1,1]. \tag{8.2}$$

We assume $\alpha, \beta \in \mathbb{R}$, $\alpha \neq 0$. This does not completely determine the solution, so one additional equation is required to uniquely determine a solution. Initially, we will not worry about this constraint in order to first better understand the structure of main set of linear requirements.

Previously in Chapter 3 and 4, we looked at the interpolation C-V and derivative C-V matrices in order to understand their structure. If our approximation is

$$u(x) = \sum_{k=0}^{2n} a_k T_k(x), \tag{8.3}$$

then the resulting linear system is

$$V\left[\alpha D + \beta I\right] a = f, \tag{8.4}$$

so that we are enforcing these linear requirements on the n Chebyshev nodes. The MSN solution comes down to computing

$$\min_{V[\alpha D + \beta I]a = f} \left|\left|D_s a\right|\right|_2, \tag{8.5}$$

which is equivalent to

$$\min_{V[\alpha D+\beta I]D_s^{-1}x=f} ||x||_2 \tag{8.6}$$

where $x = D_s a$.

From our definitions, we have the matrices

$$UWD = \begin{bmatrix} 0 & 1 & & & & & & & & & & 2n-1 & 0 \\ & & 2 & & & & & & & & 2n-2 & & \\ & & & \ddots & & & & & & \iddots & & & \\ & & & & n-2 & & & & n+2 & & & & \\ & & & & & n-1 & & n+1 & & & & & \\ & & & & & & n & & & & & & \end{bmatrix} \tag{8.7}$$

All entries not specified are 0. Similarly, we have

$$UW = \frac{1}{2} \begin{bmatrix} 2 & & - & & & & & & & & & & + & & -2 \\ & + & & - & & & & & & & & + & & - & \\ & & + & & - & & & & & & + & & - & & \\ & & & & \ddots & & & & & & \iddots & & & & \\ & & & & + & & - & 0 & + & & - & & & & \\ & & & & & + & & 0 & & - & & & & & \\ & & & & & & + & 0 & - & & & & & & \end{bmatrix}, \tag{8.8}$$

where "+" stands for +1 and "−" stands for −1. These and similar matrices were described in Sec. 3.5.

To compute the minimum norm solution, we take care to control the condition number. We let $Aa = f_2$ represent the boundary conditions of the ODE which specify the unique solution, where A has small number of rows independent of n; in this case, A is one row. First, we see

$$V\left[\alpha D + \beta I\right]\Pi = \begin{bmatrix} H_1 & H_2 \end{bmatrix}, \tag{8.9}$$

where Π is the circular downshift permutation matrix. Here, we see that H_1 (the $n \times n$ subblock) and H_2 are rank-structured matrices: that is, their off-diagonal blocks have low-rank, and in this case the matrices are almost “tridiagonal”. In Sec. 8.5, we show that H_1 and other related matrices are well-conditioned. For completeness, we show H_1 and H_2 here:

$$H_1 = \begin{bmatrix} \alpha & -\frac{\beta}{2} & & & & & \\ \frac{\beta}{2} & 2\alpha & -\frac{\beta}{2} & & & & \\ & \frac{\beta}{2} & 3\alpha & -\frac{\beta}{2} & & & \\ & & & \ddots & & & \\ & & & \frac{\beta}{2} & (n-2)\alpha & -\frac{\beta}{2} & \\ & & & & \frac{\beta}{2} & (n-1)\alpha & 0 \\ & & & & & \frac{\beta}{2} & n\alpha \end{bmatrix} \tag{8.10}$$

$$H_2 = \begin{bmatrix} & & & & & \frac{\beta}{2} & (2n-1)\alpha & -\beta & \beta \\ & & & & \frac{\beta}{2} & (2n-2)\alpha & -\frac{\beta}{2} & & \\ & & & \frac{\beta}{2} & (2n-3)\alpha & -\frac{\beta}{2} & & & \\ & & & \iddots & & & & & \\ & \frac{\beta}{2} & (n+3)\alpha & -\frac{\beta}{2} & & & & & \\ \frac{\beta}{2} & (n+2)\alpha & -\frac{\beta}{2} & & & & & & \\ (n+1)\alpha & -\frac{\beta}{2} & & & & & & & \\ -\frac{\beta}{2} & & & & & & & & \end{bmatrix}. \tag{8.11}$$

In this case, we see our entire matrix can be factored in this way:

$$\begin{bmatrix} H_1 & H_2 \\ A_1 & A_2 \end{bmatrix} = \begin{bmatrix} H_1 & \\ & I \end{bmatrix} \begin{bmatrix} I & H_3 \\ A_1 & A_2 \end{bmatrix}, \tag{8.12}$$

where $H_3 = H_1^{-1} H_2$ and $A\Pi = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$. Now, the off-diagonal ranks of H_3 are at most the sum of the ranks of H_1 and H_2, but we assume these are small so the complexity has not increased much; furthermore, these ranks are exact and there have been no approximation errors. That is, we could represent H_1, H_2, and H_3 exactly by Sequentially Semiseparable (SSS) [21] or Hierarchically Semiseparable (HSS) [15] matrices. In the constant coefficient case, the HSS and SSS ranks of H_1 and H_2 are 1, while the rank of H_3 is 2. We are now ready to multiply by D_s^{-1}, but we need to remember that we have already right-multiplied by Π. So, we set

$$\begin{aligned}
\widetilde{D}_s^{-1} &= \Pi^* D_s^{-1} \Pi \\
&= \operatorname{diag}(D_1, D_2) \\
D_1 &= \operatorname{diag}(2^s, 3^s, \cdots, (n+1)^s) \\
D_2 &= \operatorname{diag}((n+2)^s, (n+3)^s, \cdots, (2n+1)^s, 1^s)
\end{aligned} \tag{8.13}$$

in order to see

$$\begin{bmatrix} H_1 & \\ & I \end{bmatrix} \begin{bmatrix} I & H_3 \\ A_1 & A_2 \end{bmatrix} \widetilde{D}_s^{-1} = \begin{bmatrix} H_1 D_1^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & D_1 H_3 D_2^{-1} \\ A_1 D_1^{-1} & A_2 D_2^{-1} \end{bmatrix}. \tag{8.14}$$

Because $||H_3||_2$ is not too large, $\kappa_2(\begin{bmatrix} I & D_1 H_3 D_2^{-1} \end{bmatrix})$ grows slowly, and we have successfully moved the ill-conditioning from column scaling to row scaling.

To proceed we compute a ULV factorization $D_1 H_3 D_2^{-1} = U_1 L_1 V_1^*$, where L_1 is lower triangular and U_1 and V_1 are orthogonal. Factoring this information, we find

$$\begin{bmatrix} H_1 D_1^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & D_1 H_3 D_2^{-1} \\ A_1 D_1^{-1} & A_2 D_2^{-1} \end{bmatrix} = \begin{bmatrix} H_1 D_1^{-1} U_1 & \\ & I \end{bmatrix} \begin{bmatrix} I & L_1 \\ \overline{A}_1 & \overline{A}_2 \end{bmatrix} \begin{bmatrix} U_1^* & \\ & V_1^* \end{bmatrix}. \tag{8.15}$$

Here, $\overline{A}_1 = A_1 D_1^{-1} U_1$ and $\overline{A}_2 = A_2 D_2^{-1} V_1$; we set $\overline{A} = \begin{bmatrix} \overline{A}_1 & \overline{A}_2 \end{bmatrix}$. We perform another factorization $\begin{bmatrix} I & L_1 \end{bmatrix} = U_2 L_2 V_2^*$ and set $\overline{A} V_2 = \begin{bmatrix} \widetilde{A}_1 & \widetilde{A}_2 \end{bmatrix}$:

$$\begin{aligned}
\begin{bmatrix} H_1 D_1^{-1} U_1 & \\ & I \end{bmatrix} \begin{bmatrix} I & L_1 \\ \overline{A}_1 & \overline{A}_2 \end{bmatrix} \begin{bmatrix} U_1^* & \\ & V_1^* \end{bmatrix} &= \begin{bmatrix} H_1 D_1^{-1} U_1 & \\ & I \end{bmatrix} \begin{bmatrix} U_2 \begin{bmatrix} L_2 & 0 \end{bmatrix} V_2^* \\ \overline{A} \end{bmatrix} \begin{bmatrix} U_1^* & \\ & V_1^* \end{bmatrix} \\
&= \begin{bmatrix} H_1 D_1^{-1} U_1 U_2 & \\ & I \end{bmatrix} \begin{bmatrix} \begin{bmatrix} L_2 & 0 \end{bmatrix} \\ \overline{A} V_2 \end{bmatrix} V_2^* \begin{bmatrix} U_1^* & \\ & V_1^* \end{bmatrix} \\
&= \begin{bmatrix} H_1 D_1^{-1} U_1 U_2 & \\ & I \end{bmatrix} \begin{bmatrix} L_2 & 0 \\ \widetilde{A}_1 & \widetilde{A}_2 \end{bmatrix} V_2^* \begin{bmatrix} U_1^* & \\ & V_1^* \end{bmatrix}.
\end{aligned} \tag{8.16}$$

After a rotation (a small number of Householder reflectors) P to put $\widetilde{A}_2$ into lower triangular form, we compute the final factorization:

$$\begin{aligned}
&\begin{bmatrix} H_1 D_1^{-1} U_1 U_2 & \\ & I \end{bmatrix} \begin{bmatrix} L_2 & 0 \\ \widetilde{A}_1 & \widetilde{A}_2 \end{bmatrix} V_2^* \begin{bmatrix} U_1^* & \\ & V_1^* \end{bmatrix} \\
&= \begin{bmatrix} H_1 D_1^{-1} U_1 U_2 & \\ & I \end{bmatrix} \begin{bmatrix} L_2 & 0 & 0 \\ \widetilde{A}_1 & \widetilde{L} & 0 \end{bmatrix} \begin{bmatrix} I & \\ & P^* \end{bmatrix} V_2^* \begin{bmatrix} U_1^* & \\ & V_1^* \end{bmatrix}. \qquad (8.17)
\end{aligned}$$

We can easily solve for the MSN solution:

$$\begin{aligned}
\begin{bmatrix} L_2 & 0 & 0 \\ \widetilde{A}_1 & \widetilde{L} & 0 \end{bmatrix} y &= \begin{bmatrix} U_2^* U_1^* D_1 H_1^{-1} U \hat{f} \\ \hat{f}_2 \end{bmatrix} \\
a &= D_s^{-1} \Pi \begin{bmatrix} U_1 & \\ & V_1 \end{bmatrix} V_2 \begin{bmatrix} I & \\ & P \end{bmatrix} y. \qquad (8.18)
\end{aligned}$$

All of the above computations can be performed quickly.

8.3 Variable-Coefficient Scalar ODE

When we have the differential equation

$$\begin{aligned}
\alpha(x) u'(x) + \beta(x) u(x) &= f(x) \\
Lu &= g, \qquad (8.19)
\end{aligned}$$

the cost of solving the linear system becomes higher. The good news is that much remains the same; the primary difference comes in the off-diagonal ranks of the H_i matrices. If we have

$$\begin{aligned}
\alpha(x) &= \sum_{k=0}^{r} a_k T_k(x) \\
\beta(x) &= \sum_{k=0}^{r} b_k T_k(x), \qquad (8.20)
\end{aligned}$$

then from the results in Sec. 3.5 show that now H_1 and H_2 will be "banded" matrices with bandwidth $r+1$ (technically, H_2 is banded along the antidiagonal); thus, they will have off-diagonal rank of $r+1$ while H_3 will have off-diagonal rank $2r+2$. The fast algorithms will still apply, save the fact that now the structured solver will take more time. The boundary condition still adds one additional linear constraint to the overall system and is not complicated by the variable α and β.

If the coefficients are not polynomials, then the infinite series can be truncated. In the case of smooth α and β, only a few terms should be required until errors from machine precision take over. Bounds similar to those in Sec. 8.5 for the constant coefficient case should be possible so long as α_0 is sufficiently large to ensure diagonal dominance. Regardless, the fast algorithm still applies with cost proportional to the off-diagonal rank.

8.4 Systems of ODEs

We are now ready to look at systems of ODEs and will start with the constant coefficient case:

$$\begin{aligned} Au'(x) + Bu(x) &= F(x) \\ Lu &= G. \end{aligned} \tag{8.21}$$

Here, we are seeking a solution

$$\begin{aligned} u(x) &= \begin{bmatrix} u_1(x) \\ u_2(x) \\ \vdots \\ u_m(x) \end{bmatrix} \\ u_\ell(x) &= \sum_{k=0}^{2n} a_{\ell;k} T_k(x). \end{aligned} \tag{8.22}$$

with the coefficients arranged like

$$a^* = \begin{bmatrix} a_{1;0} & a_{1;1} & \cdots & a_{1;2n} & a_{2;0} & a_{2;1} & \cdots & a_{2;2n} & \cdots a_{m;0} & a_{m;1} & \cdots & a_{m;2n} \end{bmatrix} \tag{8.23}$$

From this arrangement, we see the results linear system will be

$$[A \otimes VD + B \otimes V]\, a = F, \tag{8.24}$$

where F will store all the values from the righthand side. After applying the tensored IDCT $I \otimes C^{-1}$ gives

$$[A \otimes WD + B \otimes W]\, a = \widehat{F}. \tag{8.25}$$

Applying perfect shuffle matrices [69] P and Q, we can switch the order of the Kronecker product:

$$P\,[A \otimes WD + B \otimes W]\, QQ^{*} a = P\widehat{F}, \tag{8.26}$$

which implies

$$[WD \otimes A + W \otimes B]\, Q^{*} a = P\widehat{F}. \tag{8.27}$$

Multiplying by $U \otimes I$ on the left gives

$$[UWD \otimes A + UW \otimes B]\, Q^{*} a = (U \otimes I)\, P\widehat{F}. \tag{8.28}$$

If Π is the circular downshift permutation matrix, then

$$[UWD \otimes A + UW \otimes B]\, (\Pi \otimes I) = \begin{bmatrix} H_1 & H_2 \end{bmatrix}. \tag{8.29}$$

In this case, we have the following block matrix:

$$H_1 = \begin{bmatrix} A & -\frac{1}{2}B & & & & & \\ \frac{1}{2}B & 2A & -\frac{1}{2}B & & & & \\ & \frac{1}{2}B & 3A & -\frac{1}{2}B & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \frac{1}{2}B & (n-2)A & -\frac{1}{2}B & \\ & & & & \frac{1}{2}B & (n-1)A & 0 \\ & & & & & \frac{1}{2}B & nA \end{bmatrix}. \tag{8.30}$$

This matrix has the exact same form as the constant coefficient example in Eq. (8.10). This time, H_1 now has off-diagonal blocks for rank m, implying H_1 (and H_2) are matrices with SSS and HSS rank m and that H_3 has rank $2m$.

By performing similar operations in the variable-coefficient scalar case, we see that if

$$\begin{aligned} A(x) &= \sum_{k=0}^{r} A_k T_k(x) \\ B(x) &= \sum_{k=0}^{r} B_k T_k(x), \end{aligned} \tag{8.31}$$

then the H_1 and H_2 matrices will have HSS rank $m(r+1)$ and H_3 will have rank $2m(r+1)$.

8.5 Conditioning of H_1 and Related Matrices

Here, we begin with an analysis of the conditioning of the H_1 matrix. In the constant coefficient case from Eq. 8.2, H_1 is defined as follows, which we reproduce from Eqs. (8.10) and (8.11):

$$H_1 = \begin{bmatrix} \alpha & -\frac{\beta}{2} & & & & & \\ \frac{\beta}{2} & 2\alpha & -\frac{\beta}{2} & & & & \\ & \frac{\beta}{2} & 3\alpha & -\frac{\beta}{2} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \frac{\beta}{2} & (n-2)\alpha & -\frac{\beta}{2} & \\ & & & & \frac{\beta}{2} & (n-1)\alpha & 0 \\ & & & & & \frac{\beta}{2} & n\alpha \end{bmatrix} \tag{8.32}$$

$$H_2 = \begin{bmatrix} & & & & & \frac{\beta}{2} & (2n-1)\alpha & -\beta & \beta \\ & & & & \frac{\beta}{2} & (2n-2)\alpha & -\frac{\beta}{2} & & \\ & & & \frac{\beta}{2} & (2n-3)\alpha & -\frac{\beta}{2} & & & \\ & & & ⋰ & & & & & \\ & \frac{\beta}{2} & (n+3)\alpha & -\frac{\beta}{2} & & & & & \\ \frac{\beta}{2} & (n+2)\alpha & -\frac{\beta}{2} & & & & & & \\ (n+1)\alpha & -\frac{\beta}{2} & & & & & & & \\ -\frac{\beta}{2} & & & & & & & & \end{bmatrix}. \tag{8.33}$$

If $\frac{|\beta|}{|\alpha|} < 2$, then H_1 is diagonally dominant in both the rows and columns with

$$\begin{aligned} \min_k \left(|H_{1;kk}| - \sum_{j\neq k} |H_{1;kj}| \right) &\geq |\alpha| - \frac{|\beta|}{2} \\ \min_k \left(|H_{1;kk}| - \sum_{j\neq k} |H_{1;jk}| \right) &\geq |\alpha| - \frac{|\beta|}{2}. \end{aligned} \tag{8.34}$$

From [70], we can then bound the smallest singular value:

$$\sigma_n(H_1) \geq |\alpha| - \frac{|\beta|}{2}. \tag{8.35}$$

Because

$$\begin{aligned} ||H_1||_1 &\leq n\,|\alpha| + |\beta| \\ ||H_1||_\infty &\leq n\,|\alpha| + |\beta|\,, \end{aligned} \tag{8.36}$$

we see

$$||H_1||_2 = \sigma_1(H_1) \leq n\,|\alpha| + |\beta| \tag{8.37}$$

by $||H_1||_2^2 \leq ||H_1||_1\,||H_1||_\infty$. This allows us to bound the condition number:

$$\kappa_2(H_1) = \frac{\sigma_1(H_1)}{\sigma_n(H_1)} \leq \frac{n + \frac{|\beta|}{|\alpha|}}{1 - \frac{1}{2}\frac{|\beta|}{|\alpha|}}. \tag{8.38}$$

If we look at Eq. (8.11), we have the following bounds:

$$\begin{aligned} ||H_2||_1 &\leq 2n\,|\alpha| + |\beta| \\ ||H_2||_\infty &\leq 2n\,|\alpha| + \frac{5}{2}\,|\beta| \\ ||H_2||_2 &\leq 2n\,|\alpha| + \frac{5}{2}\,|\beta|\,. \end{aligned} \tag{8.39}$$

Using the exact form of H_1 and H_2 coupled with the fact $H_3 = H_1^{-1}H_2$, we see

$$\begin{aligned} ||H_3||_2 &\leq \left|\left|H_1^{-1}\right|\right|_2 ||H_2||_2 \\ &\leq \left(\frac{1}{|\alpha| - \frac{|\beta|}{2}}\right)\left(2n\,|\alpha| + \frac{5}{2}\,|\beta|\right) \\ &\leq \frac{2n + \frac{5}{2}\frac{|\beta|}{|\alpha|}}{1 - \frac{1}{2}\frac{|\beta|}{|\alpha|}}. \end{aligned} \tag{8.40}$$

Thus, we see that $||H_3||_2 = O(n)$, so it does not grow too quickly.

We now turn our attention to bounding $D_1 H_3 D_2^{-1}$. Previously, we have noted

$$\begin{aligned} D_1 &= \mathrm{diag}\left[2^s, 3^s, \cdots, (n+1)^s\right] \\ D_2 &= \mathrm{diag}\left[(n+2)^s, (n+2)^s, \cdots, (2n+1)^s, 1^s\right]. \end{aligned} \tag{8.41}$$

Now, we see

$$\begin{aligned} \left|\left[D_1 H_3 D_2^{-1}\right]_{ij}\right| &= \left|\left(\frac{i+1}{n+1+j}\right)^s H_{3;ij}\right| \\ &\leq \left|H_{3;ij}\right|, \quad i,j \in \{1, \cdots, n\}. \end{aligned} \tag{8.42}$$

Importantly, we see that the elements of H_3 bound the elements of $D_1 H_3 D_2^{-1}$ *except in the last column*, for $H_3 \in \mathbb{R}^{n\times(n+1)}$; this is most unfortunate. To proceed further, we let

$$H_3 = \left[\widetilde{H}_3 \quad \tilde{h}\right], \tag{8.43}$$

so that $\tilde{h}$ is just the last column of H_3. We will show $||D_1\tilde{h}||_2$ is bounded, but to do so we will need to specifically look at H_3.

From our previous work we see

$$\tilde{h} = \beta H_1^{-1} e_1. \tag{8.44}$$

The obvious bound

$$\begin{aligned} ||D_1\tilde{h}||_2 &\le ||D_1||_2 \left|\left|H_1^{-1}\right|\right|_2 ||e_1||_2 \\ &\le (n+1)^s \, \frac{\frac{|\beta|}{|\alpha|}}{1-\frac{1}{2}\frac{|\beta|}{|\alpha|}} \end{aligned} \tag{8.45}$$

is too rough, for the bound $||D_1\tilde{h}||_2 = O(n^s)$ is unacceptable, for we expect that we could do better. To do so, we need to solve

$$\begin{bmatrix} \gamma & -\frac{1}{2} & & & & & \\ \frac{1}{2} & 2\gamma & -\frac{1}{2} & & & & \\ & \frac{1}{2} & 3\gamma & -\frac{1}{2} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \frac{1}{2} & (n-2)\gamma & -\frac{1}{2} & \\ & & & & \frac{1}{2} & (n-1)\gamma & 0 \\ & & & & & \frac{1}{2} & n\gamma \end{bmatrix} \tilde{h} = e_1. \tag{8.46}$$

Here, we have set $\gamma = \alpha/\beta$ (we are assuming $\beta \neq 0$). We call the above matrix $\widetilde{H}_1 = \frac{1}{\beta}H_1$.

We employ the standard tridiagonal solver algorithm (one reference is [41, Chapter 9]), although our main concern is about bounding the size of the elements of $\tilde{h}$; therefore, we are only concerned about bounding the size of the entries of ℓ_i, d_i, and u_i. We use the convention

$$\begin{aligned} &\begin{bmatrix} a_1 & c_1 & & & \\ b_1 & a_2 & c_2 & & \\ & & \ddots & & \\ & & b_{n-2} & a_{n-1} & c_{n-1} \\ & & & b_{n-1} & a_n \end{bmatrix} = \\ &\begin{bmatrix} 1 & & & & \\ \ell_1 & 1 & & & \\ & & \ddots & & \\ & & \ell_{n-2} & 1 & \\ & & & \ell_{n-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & u_1 & & & \\ & d_2 & u_2 & & \\ & & \ddots & & \\ & & & d_{n-1} & u_{n-1} \\ & & & & d_n \end{bmatrix} \\ &= \widetilde{L}\widetilde{U}. \end{aligned} \tag{8.47}$$

Clearly, we can factor $\widetilde{H}_1$ recursively as

$$\begin{aligned}
u_k &= -\frac{1}{2}, \quad k \in \{1, \cdots, n-2\} \\
u_{n-1} &= 0 \\
d_1 &= \gamma \\
\ell_1 &= \frac{1}{2\gamma} \\
d_k &= k\gamma - \ell_{k-1} u_{k-1} \\
\ell_k &= \frac{1}{2d_k}.
\end{aligned} \tag{8.48}$$

By assumption, we have $|\gamma| = \frac{|\alpha|}{|\beta|} > 2$. We will prove inductively that

$$\begin{aligned}
\operatorname{sign}(\gamma) &= \operatorname{sign}(d_k) \\
&= \operatorname{sign}(\ell_k) \\
k\,|\gamma| \le |d_k| &\le 2k\,|\gamma| \\
\frac{1}{4k\,|\gamma|} \le |\ell_k| &\le \frac{1}{2k\,|\gamma|}.
\end{aligned} \tag{8.49}$$

The results hold for $k = 1$, and assume they hold for $k \geq 1$. Then we see

$$d_{k+1} = (k+1)\gamma + \frac{1}{2}\ell_k, \tag{8.50}$$

telling us $\operatorname{sign}(d_{k+1}) = \operatorname{sign}(\ell_k) = \operatorname{sign}(\gamma)$, from which it follows

$$(k+1)\,|\gamma| \le |d_{k+1}| \le 2(k+1)\,|\gamma|\,. \tag{8.51}$$

From Eq. (8.48), it is clear $\operatorname{sign}(\ell_k) = \operatorname{sign}(d_k) = \operatorname{sign}(\gamma)$, so that

$$\frac{1}{4(k+1)\,|\gamma|} \le |\ell_{k+1}| \le \frac{1}{2(k+1)\,|\gamma|}. \tag{8.52}$$

We have just proven the inductive step. All of these hold for $k \in \{1, \cdots, n-1\}$. We have $d_n = n\gamma$ because $u_{n-1} = 0$. Therefore,

$$\widetilde{U}^{-1} e_1 = \gamma^{-1} e_1 \tag{8.53}$$

and

$$\begin{aligned}\gamma^{-1}\widetilde{L}^{-1}e_1 &= \tilde{h}\\ \tilde{h}_k &= (-1)^{k-1}\ell_{k-1}\cdots\ell_2\ell_1\gamma^{-1}.\end{aligned} \tag{8.54}$$

This gives us

$$4\left(\frac{1}{4\left|\gamma\right|}\right)^k\frac{1}{(k-1)!} \leq |\tilde{h}_k| \leq 2\left(\frac{1}{2\left|\gamma\right|}\right)^k\frac{1}{(k-1)!}. \tag{8.55}$$

We can now compute

$$\begin{aligned}||D_1\tilde{h}||_2^2 &\leq 4\sum_{k=1}^{n}(k+1)^{2s}\left(\frac{1}{2\left|\gamma\right|}\right)^{2k}\frac{1}{\left[(k-1)!\right]^2}\\ &\leq 4\sum_{k=0}^{n-1}(k+2)^{2s}\left(\frac{1}{2\left|\gamma\right|}\right)^{2k+2}\frac{1}{\left[k!\right]^2}\\ &\leq \frac{1}{\left|\gamma\right|^2}\left\{4^s+\sum_{k=1}^{n-1}(k+2)^{2s}\left(\frac{1}{2\left|\gamma\right|}\right)^{2k}\frac{1}{\left[k!\right]^2}\right\}\\ &\leq \frac{1}{\left|\gamma\right|^2}\left\{4^s+9^s\sum_{k=1}^{\infty}k^{2s}\left(\frac{1}{2\left|\gamma\right|}\right)^{2k}\frac{1}{\left[k!\right]^2}\right\}\\ &\leq \frac{1}{\left|\gamma\right|^2}\left\{4^s+9^sC(s,\gamma)\right\}.\end{aligned} \tag{8.56}$$

Here, we have

$$C(s,\gamma)=\sum_{k=1}^{\infty}k^{2s}\left(\frac{1}{2\left|\gamma\right|}\right)^{2k}\frac{1}{\left[k!\right]^2}, \tag{8.57}$$

so that we must now bound $C(s,\gamma)$. Currently, we have the bound

$$\begin{aligned}C(s,\gamma) &\leq \max_k\left|k^{2s}\left(\frac{1}{2\left|\gamma\right|}\right)^{2k}\right|\sum_{k=1}^{\infty}\frac{1}{\left[k!\right]^2}\\ &\leq e\left[\frac{2s}{\ln\left(4\left|\gamma\right|\right)}\right]^{2s}\left(\frac{1}{2\left|\gamma\right|}\right)^{\left[\frac{4s}{\ln(4|\gamma|^2)}\right]}\\ &\sim s^{2s}.\end{aligned} \tag{8.58}$$

This is a loose bound, and it should be possible to produce better bounds by attempting to compute

$$C(s,\gamma) \leq \max_k \left| \frac{k^{2s}}{[k!]^2} \right| \sum_{k=1}^{\infty} \left(\frac{1}{2\left|\gamma\right|} \right)^{2k} \tag{8.59}$$

We know the geometric series will sum to a constant, but maximizing over the polynomial and factorial is nontrivial.

We set

$$E(s,\gamma) = \frac{1}{|\gamma|} \sqrt{\sum_{k=0}^{\infty} (k+2)^{2s} \left(\frac{1}{2\left|\gamma\right|} \right)^{2k} \frac{1}{[k!]^2}}\,. \tag{8.60}$$

For all n, we have

$$||D_1\tilde{h}||_2 \leq E(s,\gamma). \tag{8.61}$$

By computation, we have

$$E(10,2) \leq 2. \tag{8.62}$$

This shows that we *greatly* overestimated $||D_1\tilde{h}||_2$ and suggests it is well-conditioned; the challenge is determining rigorous bounds.

We also have the lower bound

$$\begin{aligned} ||D_1\tilde{h}||_2^2 &\geq 16 \sum_{k=1}^{n} (k+1)^{2s} \left(\frac{1}{4\left|\gamma\right|} \right)^{2k} \frac{1}{[(k-1)!]^2} \\ &\geq \frac{1}{|\gamma|^2} \sum_{k=0}^{n-1} (k+2)^{2s} \left(\frac{1}{4\left|\gamma\right|} \right)^{2k} \frac{1}{[k)!]^2} \\ &\geq \frac{1}{2\left|\gamma\right|^2} \sum_{k=1}^{\infty} k^{2s} \left(\frac{1}{4\left|\gamma\right|} \right)^{2k} \frac{1}{[k!]^2} \end{aligned} \tag{8.63}$$

where the last inequality holds for sufficientlylarge n. This has the same form as our upper bound. Future work will be devoted to obtaining a tight bound for $||D_1\tilde{h}||_2$.

Before over previous work on bounding $||D_1\tilde{h}||_2$, we previously showed

$$|\widetilde{H}_3|_{ij} \leq |H_{3;ij}|, \quad i,j \in \{1, \cdots, n\}, \tag{8.64}$$

Because

$$\begin{aligned} ||A||_1 &= ||\,|A|\,||_1 \\ ||A||_\infty &= ||\,|A|\,||_\infty \end{aligned} \tag{8.65}$$

holds for all matrices, we have the following bound:

$$\begin{aligned} ||\,|A|\,||_2 &\leq \sqrt{||\,|A|\,||_1\,||\,|A|\,||_\infty} \\ &= \sqrt{||A||_1\,||A||_\infty}. \end{aligned} \tag{8.66}$$

This implies we have this final bound:

$$||\widetilde{H}_3||_2 = O(n). \tag{8.67}$$

With our above bound on $||\tilde{h}||_2$ combined with the fact

$$\left|\left|D_1 H_3 D_2^{-1}\right|\right|_2 \leq \sqrt{||\widetilde{H}_3||_2^2 + ||\tilde{h}||_2^2}, \tag{8.68}$$

we have shown

$$\left|\left|D_1 H_3 D_2^{-1}\right|\right|_2 = O_s(n), \tag{8.69}$$

where the constant depends on s.

We now turn our attention to bounding $\kappa_2(\begin{bmatrix} I & H \end{bmatrix})$, and our goal will be to look at $H = D_1 H_3 D_2^{-1}$. First, we see

$$\begin{aligned} x^* \left(\begin{bmatrix} I & H \end{bmatrix} \begin{bmatrix} I \\ H^* \end{bmatrix} \right) x &= x^*x + x^*HH^*x \\ &\geq 1 \end{aligned} \tag{8.70}$$

when $||x||_2 = 1$. Next, we also have

$$x^* \left(\begin{bmatrix} I & H \end{bmatrix} \begin{bmatrix} I \\ H^* \end{bmatrix} \right) x = x^*x + x^*HH^*x$$
$$\leq 1 + ||H||_2^2 . \tag{8.71}$$

Thus, we now have

$$\kappa_2(\begin{bmatrix} I & H \end{bmatrix}) \leq \sqrt{1 + ||H||_2^2} \tag{8.72}$$

and so

$$\kappa_2(\begin{bmatrix} I & D_1 H_3 D_2^{-1} \end{bmatrix}) = O_s(n). \tag{8.73}$$

This shows $\begin{bmatrix} I & D_1 H_3 D_2^{-1} \end{bmatrix}$ is well-conditioned.

Similar results exist for systems of ODEs for constant coefficient; we omit the details.

8.6 Discussion of ODE Solvers and Extending Fast MSN Methods to PDEs

In this chapter we looked at the potential of using the MSN method to solve ordinary differential equations. Due to the nature of the C-V matrices, there is much structure that can be exploited for a fast solver.

Much of this work, though, could also apply in the situation where we decide to not use MSN and form a square system; as mentioned before, this is used in Chebfun [26], with some of the algorithm details in [5, 25, 74]. While Chebfun uses Chebyshev polynomial extrema instead of Chebyshev polynomial roots, similar results should hold. In particular, similar conditioning results should hold as well as the low-rank off-diagonal blocks. From [67, Appendix A], it appears Chebfun uses slow, dense algorithms for computing higher and higher approximations given an ODE, as it is stated that "the work involved . . . is proportional to the cube of the number of grid points." The difficulty with large systems is specifically mentioned [67, Pages 306–307]; this would be less challenging by taking advantage of the structured system. To be sure, structured solvers require more effort [33], but the speedup is worth the additional cost. One additional feature of Chebfun is its ability to compute eigenfunctions. Future work will investigate if MSN could compute this as well, an interesting question because MSN is inherently nonsquare while eigenvalues only make sense in a square system.

A recent article [61] talks about using randomized sampling to speed up the computation of fast direct solvers for second order ODEs, although the results can be generalized and extended. They note that even though steep gradients exist, it is possible that the off-diagonal blocks still have a low-rank property. This leads to compression with errors that can made made small by setting a sufficiently small tolerance. This suggests that, given the expansion in Eq. (8.31), it may be possible to perform similar compression directly using the coefficients themselves; how this could be done systematically is not clear, though. Using randomized sampling to compute a structured factorization of a matrix is similar to the work in [33], which we discuss in Chapter 9.

Previous work has shown that the MSN method can solve 2D PDEs well [20]; the difficulty arises in storing the matrices. This is especially important in 3D, as the flop cost for slow algorithms is $O(n^9)$ for an n^3 tensor grid with $O(n^6)$ memory units required to hold the matrices themselves; this is impractical if not impossible for any reasonable n. Extending the results here to solving PDEs may give rise to fast, high-order methods for PDEs. In the constant coefficient case, the tensor product nature may allow us to form the normal equations and invert them quickly, similar to the fast LQ factorization discussed here. The variable coefficient will be more challenging, although there is potential to see if the structures here carry over in certain situations in 2D and 3D.

Chapter 9

Stopping Criterion for Randomized Low-Rank Approximations

> The probability professor points to a random student in the third row and says “Get out. I do not teach unlucky students.”
>
> One way to start a probablity course;
> suggested by Elliot Hudgins

There has been work in recent years to understand structured matrices: matrices with off-diagonal blocks that are (or can be approximated as) low-rank. The goal is to develop methods which allow us to compute matrix-vector products and matrix inverses faster than standard algorithms; that is, multiplication in $O(n \log^{\beta} n)$ flops and inversion in $O(n^{\alpha} \log^{\beta} n)$ flops for $\alpha \in [1,2]$ and β small. Another benefit is reduced storage requirements, frequently $O(n \log^{\beta} n)$. The simplest of these are banded matrices, but also include Sequentially Semi-Separable matrices [21], Hierarchically Semi-Separable (HSS) matrices [15, 17], H-matrices [39], and others. Recent work involving randomized HSS construction can be found in [31, 51, 56]. The main contribution of this chapter is related to a stochastic estimate of $||\cdot||_F$ which allows us to accurately measure the low-rank approximation and determine when it is well-approximated; portions of this material first appeared in [33]. In particular, we develop a method to compute a *relative* stopping criterion for an adaptive low-rank approximation.

Throughout this chapter we assume $A \in \mathbb{R}^{m \times n}$ with rank $r \ll \min(m, n)$. For simplicity, we will assume $m \geq n$. We let $N(0, 1)$ refer to the standard normal distribution with mean 0 and variance 1. Finally, some of the notation in this chapter may conflict with those from previous chapters. While other adaptive randomized algorithms have frequently used an absolute tolerance ε, we will focus on allowing both an absolute tolerance ε_{abs} and a relative tolerance ε_{rel}.

9.1 Randomized Low-Rank Approximation

In recent years there has been growing interest in using randomization to speed up conventional numerical linear algebra techniques; one standard reference is [40]. The main idea is that we would like to compute accurate factorizations like pivoted QR or the SVD on low-rank matrices. These factorizations usually have flop counts of $O(mnr)$ and $O(mn^2)$, respectively, but the data transfer required makes these computations expensive, especially when matrices are so large as to not fit in fast memory. Thus, it is impractical to use standard dense algorithms with large communication costs on distributed memory machines. By looking at random samplings of the range, the desire is to build up an approximation of the matrix until reaching a predetermined approximation tolerance. Once a sufficient number of random samples have been computed, we perform a pivoted QR or SVD on this smaller dense matrix, leading to a smaller overall cost. The random samplings are computed using matrix-matrix multiplication, which can be efficiently computed on modern hardware. Although these multiplications may be a significant portion of the overall flop count, the total time spent performing multiplication is small. Thus, the goal is to determine when enough samples have been computed to ensure an

`rand`(m,n)	an $m \times n$ matrix with iid $N(0,1)$ elements
`cols`(A)	number of columns of A
`qr`(A)	unpivoted QR factorization of A returning Q or Q and R
`rrqr`($A, \varepsilon_{\text{abs}}, \varepsilon_{\text{rel}}$)	rank-revealing QR factorization of A such as QRCP with absolute/relative tolerances $\varepsilon_{\text{abs}}/\varepsilon_{\text{rel}}$

Table 9.1: List of helper functions for low-rank approximation.

accurate approximation of the matrix. The standard algorithm for building a low-rank approximation is found in Alg. 5, with the primary arguments being the block size d and `stopping_criterion`, a function which determines when we have a good enough approximation of the range of A; similar blocked algorithms can be found in [52, 75] and are reproduced below. In Line 8 of Alg. 5, we perform iterated Gram-Schmidt orthogonalization for numerical stability, as it helps ensure Q is *numerically* orthogonal to machine precision, especially in the blocked case [9, 64]. This may not always be necessary, for [64] notes this would only be required should the column norms decrease by a significant factor; even so, algorithmically it is easier, though more expensive, to always perform iterated GS. Additionally, although `rand` could produce any kind of random matrices, our analysis requires us to use Gaussian random variables. See Table 9.1 for a list of helper functions for the algorithms in this chapter. We denote the pivoted QR factorization as `rrqr` and it stops when $|R_{k,k}| < \varepsilon_{\text{abs}}$ or $|R_{k,k}| < \varepsilon_{\text{rel}}\,|R_{1,1}|$.

These algorithms frequently use a QB approximation of A. Here Q is an approximate orthonormal basis for the range of A with $B = Q^*A$; that is

$$\begin{aligned} A &\approx Q(Q^*A) \\ &= QB. \end{aligned} \tag{9.1}$$

Knowing when $||A - QB||$ is small is our primary concern, and the particular choice of norm is important. This chapter will investigate when this happens, propose a new method, and compare it with existing stopping criteria. The stopping criterion presented here has focused on developing a *relative* stopping criterion. This is particularly useful as

Algorithm 5 Randomized Block Low-Rank Approximation (General)

```
function RANDOM_LOW_RANK($A$,$d$,stopping_criterion)    ▷ Builds $QB$ approximation
    $Q = [\,]$; $n = \texttt{cols}(A)$
    $k = 0$
    while !stopping_criterion($Q$,$A$,$\varepsilon$) do
        $k = k + 1$
        $\Omega_k = \texttt{rand}(n, d)$    ▷ $d$ is block size
        $S_k = A\Omega_k$
        $\widehat{S}_k = (I - QQ^*)^2 \, S_k$    ▷ 2× GS orthogonalization for numerical stability
        $[Q_k, R_k] = \texttt{qr}(\widehat{S}_k)$
        $Q = \begin{bmatrix} Q & Q_k \end{bmatrix}$
    end while
    $B = Q^* A$
    return $Q$, $B$    ▷ We have the approximation $A \approx QB$
end function
```

it may be more convenient to specify a relative error tolerance over an absolute tolerance. Furthermore, we would like this method to work in the case we do not have explicit access to the matrix. Theorem 4.2 in [72] says that if H is a structured $N \times N$ matrix and $\widetilde{H}$ is H with off-diagonal blocks truncated to tolerance ε_{rel}, then

$$||H - \widetilde{H}||_F \leq C(N, L)\varepsilon_{\text{rel}} \, ||H||_F \, . \tag{9.2}$$

Thus, it makes sense to seek an accurate relative stopping criterion. Once a sufficient number of random samples have been computed, we perform a pivoted QR factorization (such as QR with Column Pivoting, although the Strong RRQR from [37] would be better) on these samples in order to obtain a high-quality Q for a better QB approximation. While this last step may not be necessary and was not included in [52, 75], this would be an easy step to add and was desired in the original setting where this stopping criterion was developed [33].

We recall the Eckhart-Young theorem [36, Theorem 1.1], which gives optimal low-rank approximation bounds in the 2-norm and F-norm:

Theorem 9.1 (Eckhart-Young Theorem)

If A_k is the matrix A chopped to k singular values, then we have the following bounds on approximations of A:

$$\begin{aligned}\min_{\operatorname{rank}(B)\leq k} ||A - B||_2 &= ||A - A_k||_2 \\ &= \sigma_{k+1} \\ \min_{\operatorname{rank}(B)\leq k} ||A - B||_F &= ||A - A_k||_F \\ &= \sqrt{\sum_{j=k+1}^{\min(m,n)} \sigma_j^2}\,. \end{aligned} \tag{9.3}$$

In practice, we hope to be able to determine approximations with near-optimal ranks within a specified tolerance with smaller overall computation time. Thus, we are looking at the fixed-precision low-rank approximation problem.

9.2 Stopping Criteria

The unitary invariance of the 2-norm combined with it being the definition of the operator (matrix) norm makes it desirable to bound $||(I - QQ^*)A||_2$. The main challenge of the 2-norm stems from the computational complexity required to compute it exactly. Because of this, the F-norm is the next norm that we may wish to bound, and it is easy to compute if we have explicit access to the matrix in question; furthermore, it trivially bounds the 2-norm. If A is large or implicitly defined, $||A||_F$ may still be expensive to compute. Additionally, the exact norm is frequently not required so much as an accurate approximation of it. We will be making comparisons with algorithms in [52, 75], so we reproduce their blocked versions here. Alg. 6 is from [52, Figure 2] (the MV Algorithm, named after the authors) and Alg. 7 is from [75, Algorithm 2] (the YGL Algorithm, named after the authors). Algs. 6 and 7 originally used $Q = \texttt{orth}(A)$ to compute the orthonormal basis of a matrix A; this work uses the function `qr` in our notation because it is an unpivoted QR factorization, and our new stopping criterion will require both Q

and R factors of the unpivoted QR factorization. A variation of iterated Gram-Schmidt orthogonalization can be found in Line 5 of Alg. 6 and Line 7 of Alg. 7 to help Q retaining numerical orthogonality. It is explicitly stated in [52, Remark 5] that the fact R is not used in the `qr` routine (original `orth`). From our work here, we see the R factor helps ensure we do not take too many random samples by noting when the R-values become small. Even so, no pivoting is required.

One stopping criterion is based on the following lemma [40]:

Lemma 9.2 (HMT Error Bound; Lemma 4.1 in [40])
Let $B \in \mathbb{R}^{m\times n}$. Fix a positive integer d and $\alpha > 1$. Draw an independent family of standard Gaussian vectors $\{\omega_k\}_{k=1}^{d}$. Then

$$||B||_2 \le \alpha\sqrt{\frac{2}{\pi}} \max_{k=1,\cdots,d} ||B\omega_k||_2 \tag{9.4}$$

with probability $1 - \alpha^{-d}$.

The reference to "HMT" comes from the first letters of the authors' last names in [40] and is used in [40, Alg. 4.2] (Adaptive Randomized Range Finder). Here, α can be viewed as a trade-off parameter: larger values ensure a smaller probability of failure. It turns out, though, this overestimation is significant; this will be shown for a number of matrices in Sec. 9.7. We can find no references to randomized lower bounds of $||B||_2$; this would be helpful in producing an order-of-magnitude estimate or estimates relating to the variance or other moments of the random variable. This method is used in [47, 60, 73] as a stopping criterion for adaptive randomized algorithms for structured matrices (the same situation where this work was originally developed).

The authors in [52] note the drawback of Lem. 9.2 of this over estimation and instead use the explicit bound $||A|| < \varepsilon$ in Alg. 6, where we note A is updated every time after having known information subtracted out during each iteration. The authors said that $||A||_2$ could be used but noted that $||A||_F$ may be preferred for its easy computation.

Algorithm 6 Randomized Block Low-Rank Approximation (MV) [52, Figure 2]

```
function randQB_b_MV($A$,$\varepsilon$,$d$)
    for $k = 1, 2, 3, \cdots$ do
        $\Omega_k = \texttt{rand}(n, d)$
        $Q_k = \texttt{qr}(A\Omega_k)$
        $Q_k = \texttt{qr}(Q_k - \sum_{j=1}^{k-1} Q_j Q_j^* Q_k)$
        $B_k = Q_k^* A$
        $A = A - Q_k B_k$
        if $||A|| < \varepsilon$ then
            stop
        end if
    end for
    $Q = \begin{bmatrix} Q_1 & \cdots & Q_k \end{bmatrix}$
    $B = \begin{bmatrix} B_1^* & \cdots & B_k^* \end{bmatrix}^*$
    return $Q$, $B$
end function
```

Algorithm 7 Randomized Block Low-Rank Approximation (YGL) [75, Algorithm 2]

```
function randQB_b_YGL($A$,$\varepsilon$,$d$)
    $Q = [\,]$; $B = [\,]$
    $E = ||A||_F^2$
    for $k = 1, 2, 3, \cdots$ do
        $\Omega_k = \texttt{rand}(n, d)$
        $Q_k = \texttt{qr}(A\Omega_k - Q(B\Omega_k))$
        $Q_k = \texttt{qr}(Q_k - Q(Q^* Q_k))$
        $B_k = Q_k^* A$
        $Q = \begin{bmatrix} Q & Q_k \end{bmatrix}$
        $B = \begin{bmatrix} B^* & B_k^* \end{bmatrix}^*$
        $E = E - ||B_k||_F^2$
        if $E < \varepsilon^2$ then
            stop
        end if
    end for
    return $Q$, $B$
end function
```

One drawback (noted in [75]) is that this requires access to a dense copy of A. This is problematic with regards to memory because we may want to keep an unmodified version of the original matrix; furthermore, if A is originally sparse, it will immediately become dense after subtracting the first approximation, greatly increasing memory requirements.

This resulted in the development of the stopping criterion in [75]. Instead of storing a dense matrix with the unused information of A to keep track of the error, it is possible to compute the error just based on the difference in F-norm. This is made explicit in the next theorem, which we call the "YGL Error Bound" because of the first letters of the authors' last names. Calling this an error bound is slightly misleading because the error is exact, not bounded, but a better title escapes us at this time.

Theorem 9.3 (YGL Error Bound; Theorem 1.1 in [75])
Let $A \in \mathbb{R}^{m\times n}$ and $Q \in \mathbb{R}^{n\times k}$ be orthogonal. If $B = Q^*A$ then

$$||A - QB||_F^2 = ||A||_F^2 - ||B||_F^2 . \tag{9.5}$$

If one builds up Q in blocks, then we can compute the true F-norm error provided $||A||_F$ is already known. Because this value only needs to be computed once, it may not be too expensive; however, this only works if the matrix is explicitly, not implicitly, defined. Additionally, Thm. 9.3 is only useful for relative tolerances down to $O(\sqrt{\varepsilon_{\mathrm{mach}}})$, where $\varepsilon_{\mathrm{mach}}$ is machine precision, as noted in [75, Theorem 3]. It is possible, using techniques such as compensated summation [41, Chapter 4], that these limitations may be lifted, but this would be difficult in the parallel setting. This may be the first explicit reference to a relative error bound in the literature, though we seek to remove the relative tolerance restrictions.

9.3 Previous Probabilistic Bounds

We now present results from [36], although looser bounds from [40] may be more well-known. The important portions of the bounds will be emphasized here; the original paper should be referenced for the full results and proofs.

Theorem 9.4 (Average 2-Norm and F-Norm Error; Theorem 5.7 in [36])
Let Ω be a Gaussian random matrix with $k+p$ columns for $p \geq 2$, $S = (AA^*)^q A\Omega$, and $Q = \texttt{qr}(S)$. Then

$$\begin{aligned} \mathbb{E}\, ||A - QQ^*A||_2 &\leq \sqrt{\sigma_{k+1}^2 + C(A,k,p,q)} \\ \mathbb{E}\, ||A - QQ^*A||_F &\leq \sqrt{\left(\sum_{j=k+1}^{n} \sigma_j^2\right) + D(A,k,p,q)} \end{aligned} \tag{9.6}$$

Here, C and D are constants which decay exponentially as q increases.

The above theorem states the expected value of the error is close to optimal. Similar results hold for upper tail bounds:

Theorem 9.5 (Probability Tail Bounds in 2-Norm and F-Norm; Theorem 5.8 in [36])
Let the assumptions of Thm. 9.4 hold. Then if $0 < \Delta \ll 1$,

$$\begin{aligned} ||A - QQ^*A||_2 &\leq \sqrt{\sigma_{k+1}^2 + C(A,k,p,q,\Delta)} \\ ||A - QQ^*A||_F &\leq \sqrt{\left(\sum_{j=k+1}^{n} \sigma_j^2\right) + D(A,k,p,q,\Delta)} \end{aligned} \tag{9.7}$$

hold for probability $1 - \Delta$. C and D are constants which decay exponentially as q increases.

Additionally, [36, Theorem 5.6] bounds deviations of the singular values of QQ^*A from the singular values of A.

These are useful results and help explain why randomized methods perform better than expected if one looks at similar bounds in [40]. The downside is that in practice they are not helpful because we must know the singular values.

In these theorems, q refers to the possibility of using power iteration in order to increase the quality of the approximation; this is a well-known technique in randomized numerical linear algebra [40, 52, 75] and helps ensure the larger singular values and corresponding singular vectors are matched more closely [36, 59]. Power iteration is not mentioned in [33] because it is not clear how to use it in randomized HSS construction without greatly increasing the overall computational cost; it is likely this also holds in general for the randomized construction of structured matrices because we do not have explicit access to matrix subblocks. This lead to the desire for an accurate method to determine low-rank approximations which could be used with small relative tolerances (for example, tolerances as small as 10^{-5} in single precision and 10^{-14} in double precision).

9.4 Basic Probability Theory

We assume A has positive singular values $\sigma_1 \geq \cdots \geq \sigma_r > 0$. It follows that we have the SVD

$$\begin{aligned} A &= U\Sigma V^* \\ &= \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix} \\ &= U_1 \Sigma_r V_1^*, \end{aligned} \tag{9.8}$$

where

$$\Sigma_r = \operatorname{diag}\left(\sigma_1, \cdots, \sigma_r\right). \tag{9.9}$$

If $x \in \mathbb{R}^n$ is a Gaussian random vector (that is, $x_i \sim N(0,1)$) and we set $\xi = V^* x$, then by the rotational invariance of $||\cdot||_2$ we have

$$\begin{aligned} ||Ax||_2^2 &= ||\Sigma_r \xi||_2^2 \\ &= \sigma_1^2 \xi_1^2 + \cdots + \sigma_r^2 \xi_r^2. \end{aligned} \tag{9.10}$$

Now, $\xi_i \sim N(0,1)$ because rotations of Gaussian random vectors are also Gaussian random vectors. From here, we can see

$$\mathbb{E}\,||Ax||_2^2 = ||A||_F^2\,. \tag{9.11}$$

This by itself is useful, but, in order to make use of computer architecture, matrix-matrix products are preferred over multiple matrix-vector products. Thus, if $\Omega \in \mathbb{R}^{n\times d}$ with $\Omega_{ij} \sim N(0,1)$ (so that Ω is a Gaussian random matrix), it follows that

$$||A\Omega||_F^2 = ||A\Omega_{:,1}||_2^2 + \cdots + ||A\Omega_{:,d}||_2^2\,. \tag{9.12}$$

Because each $\Omega_{:,i}$ is a Gaussian random vector, we also have

$$\mathbb{E}\,||A\Omega||_F^2 = d\,||A||_F^2\,. \tag{9.13}$$

It is this equality that allows us to accurately compute the F-norm of matrices using random range samples.

Because there is interest in using power iterations to increase the quality of randomized low-rank factorizations [40], we also include these related results. In particular, it is clear from the previous work that

$$\begin{aligned} ||(AA^*)^q\,Ax||_2^2 &= \sigma_1^{4q+2}\xi_1^2 + \cdots + \sigma_r^{4q+2}\xi_r^2 \\ ||(A^*A)^q\,x||_2^2 &= \sigma_1^{4q}\xi_1^2 + \cdots + \sigma_r^{4q}\xi_r^2, \end{aligned} \tag{9.14}$$

so we have

$$\begin{aligned} \mathbb{E}\,||(AA^*)^q\,Ax||_2^2 &= ||A||_{s,4q+2}^{4q+2} \\ \mathbb{E}\,||(A^*A)^q\,x||_2^2 &= ||A||_{s,4q}^{4q} \end{aligned} \tag{9.15}$$

and

$$\begin{aligned}\mathbb{E}\left|\left|(AA^*)^q A\Omega\right|\right|_2^2 &= d\left|\left|A\right|\right|_{s,4q+2}^{4q+2} \\ \mathbb{E}\left|\left|(A^*A)^q \Omega\right|\right|_2^2 &= d\left|\left|A\right|\right|_{s,4q}^{4q}.\end{aligned} \tag{9.16}$$

Here, $||\cdot||_{s,p}$ is the Schatten p-norm defined in Eq. (2.34) and Ω is a Gaussian random matrix with d columns, as before. From the definitions of the Schatten p-norm, it is clear

$$\begin{aligned}\left[\mathbb{E}\left|\left|(AA^*)^q Ax\right|\right|_2^2\right]^{1/4q+2} &\to ||A||_2 \\ \left[\mathbb{E}\left|\left|(A^*A)^q x\right|\right|_2^2\right]^{1/4q} &\to ||A||_2 \\ \left[\mathbb{E}\left|\left|(AA^*)^q A\Omega\right|\right|_2^2\right]^{1/4q+2} &\to ||A||_2 \\ \left[\mathbb{E}\left|\left|(A^*A)^q \Omega\right|\right|_2^2\right]^{1/4q} &\to ||A||_2\end{aligned} \tag{9.17}$$

as $q \to \infty$.

The proof that independent realizations of Eqs. (9.13) and (9.16) produce accurate approximations of the F-norm and Schatten norm will be postponed until Sec. 9.6. We now focus on developing our stopping criterion, which is based on the fact that we can now accurately estimate our error in a matrix norm.

9.5 New Stopping Criterion

We now step through Alg. 5 to see where some problems, previously ignored, may arise; this stems from the fact that we are wanting to be able to have relative error tolerances on the order of 10^{-12} or 10^{-14} in double precision. We will be building our random matrices up in blocks of d random vectors at a time, although in theory the size of the blocks could vary. Assume we have our absolute tolerance ε_{abs} and relative tolerance ε_{rel}. Then our `stopping_criterion` function will return `true` if

$$||\widehat{S}_k||_F < \max(\varepsilon_{\text{abs}}\sqrt{d}, \varepsilon_{\text{rel}}\,||S_k||_F); \tag{9.18}$$

otherwise, `stopping_criterion` returns `false`.

1. Set $\Omega_i = \texttt{rand}(n, d)$ and compute $S_i = A\Omega_i$ for $i \in \{1, 2\}$; set $Q = [\,]$ and $S = \begin{bmatrix} S_1 & S_2 \end{bmatrix}$.

2. Compute $[Q_1, R_1] = \texttt{qr}(S_1)$, giving the initial approximation for the range of A; set $Q = Q_1$.

3. Set $\widehat{S}_2 = (I - QQ^*)^2\, S_2$, so that $\widehat{S}_2$ contains potentially new information about the range of A. If

$$||\widehat{S}_2||_F < \max(\varepsilon_{\text{abs}}\sqrt{d}, \varepsilon_{\text{rel}}||S_2||_F), \tag{9.19}$$

   then stop and perform $Q = \texttt{rrqr}(S, \varepsilon_{\text{abs}}, \varepsilon_{\text{rel}})$.

4. Set $\Omega_3 = \texttt{rand}(n, d)$ and compute $S_3 = A\Omega_3$; set $S = \begin{bmatrix} S & S_3 \end{bmatrix}$.

5. Compute $[Q_2, R_2] = \texttt{qr}(\widehat{S}_2)$, giving new information for the range of A; set $Q = \begin{bmatrix} Q & Q_2 \end{bmatrix}$.

6. Set $\widehat{S}_3 = (I - QQ^*)^2\, S_3$, so that $\widehat{S}_3$ contains potentially new information about the range of A. If

$$||\widehat{S}_3||_F < \max(\varepsilon_{\text{abs}}\sqrt{d}, \varepsilon_{\text{rel}}||S_3||_F), \tag{9.20}$$

   then stop and perform $Q = \texttt{rrqr}(S, \varepsilon_{\text{abs}}, \varepsilon_{\text{rel}})$.

7. Continue this process until convergence ...

Using individual realizations of Eq. (9.13) is important in the stopping criterion, although the relative stopping $||\widehat{S}_k||_F < \varepsilon_{\text{rel}}||S_k||_F$ is new. Even so, in order to ensure we are adding new information to Q in steps 2 and 5, we need to know $\widehat{S}_k$ is full rank. This problem does not show up in the unblocked version because updating Q one vector at a time makes it easy to determine when a vector has small norm. This situation, when $\widehat{S}_k$ is numerically low-rank, is not considered in Algs. 6 and 7 due to how the Q_k blocks are formed, although this is critical to ensure Q has orthonormal columns. As noted previously, performing multiple matrix-vector multiplications is inefficient on modern machines, which is why the blocked case is important. Thus, if $\widehat{S}_k$ is (numerically) rank-deficient, then additional random samples contain no new information above the specified tolerance and we should compute the rank-revealing QR factorization on our random samples. If $\widehat{S}_k$ is determined to be rank-deficient, then $\begin{bmatrix} S_1 & S_2 & \cdots & S_k \end{bmatrix}$ is rank-deficient. This implies the random samples matrix

$$S = \begin{bmatrix} S_1 & S_2 & \cdots & S_k & S_{k+1} \end{bmatrix}. \tag{9.21}$$

is rank-deficient as well and the additional block of random samples S_{k+1} should ensure a better quality RRQR factorization than just using $\begin{bmatrix} S_1 & S_2 & \cdots & S_k \end{bmatrix}$. Although this is not a proof, extensive examples in Sec. 9.7 show that this is expected and matches our intuition.

To determine numerical rank-deficiency, we set

$$\rho = \frac{||S_1||_F}{\sqrt{d}}, \tag{9.22}$$

implying $\rho \approx ||A||_F$, and say $\widehat{S}_k$ is numerically rank-deficient when

$$\min_{j=1,\cdots,d} |\,(R_k)_{j,j}\,| < \max(\varepsilon_{\text{abs}}, \varepsilon_{\text{rel}}\rho). \tag{9.23}$$

The purpose of ρ is to characterize how large the original norms of the random samples should be, as this will help determine when they have fallen below the desired tolerance. Because of this, other potential values of ρ are $|(R_1)_{1,1}|$, $\max_j |(R_1)_{j,j}|$, or $\max_{j,k} |(R_k)_{j,j}|$. If the desire is to minimize communication, then $|(R_1)_{1,1}|$ or $\max_j |(R_1)_{j,j}|$ may be preferred. Because qr is unpivoted, there is no guarantee the diagonals of R will always decrease in value. Even so, extensive tests have shown there is general decay, but more theoretical work that needs to be done. This is one reason for using the additional block of samples S_{k+1} in rrqr.

Taken together, this gives us a new stopping criterion for low-rank approximations, presented in Alg. 8. At this point, we now compare our stopping criterion with the ones previously mentioned.

The MV stopping criterion explicitly assumes an absolute stopping tolerance. The YGL stopping criterion has an absolute stopping tolerance that requires and depends on $||A||_F$, so this could be viewed as a relative stopping criterion with the restriction that $\varepsilon_{\text{rel}} \geq C\sqrt{\varepsilon_{\text{mach}}}$. One important benefit to the MV and YGL stopping criteria is that they are exact: we know the exact error in the F-norm at each step in the process. Our new stopping criterion allows for absolute and relative stopping criteria explicitly; furthermore, the relative tolerance is limited by the ability to accurately compute

$$||\widehat{S}_k||_F < \varepsilon_{\text{rel}}\, ||S_k||_F \quad \text{and} \quad \min_{j=1,\cdots,d} |(R_k)_{j,j}| < \varepsilon_{\text{rel}}\rho. \tag{9.24}$$

Thus, we expect to be able to compute relative tolerances down to $O(\varepsilon_{\text{mach}})$, although we have not determined the exact restrictions. It is likely that we must have $\varepsilon_{\text{rel}} \geq C_{n,d}\varepsilon_{\text{mach}}$, but we will not investigate the nature of $C_{n,d}$ at this time, as it is related to the accuracy of matrix-matrix multiplication [41, Chapter 3], blocked GS [9, 64], and QR factorizations [41, Chapter 19]. The results in Sec. 9.7 and [33] show that we can obtain excellent results using our relative stopping criterion even though our estimates are probabilistic.

9.6 Probability Theory Proofs

To simplify our analysis, we begin by defining a random variable which has the same properties as $||Ax||_F^2$:

$$X \sim \sigma_1^2 \xi_1^2 + \cdots + \sigma_r^2 \xi_r^2. \tag{9.25}$$

Here, $\xi_i \sim N(0,1)$, so we have $\mathbb{E}\,||Ax||_2^2 = ||A||_F^2$. We now average X to arrive at the random variable

$$\overline{X}_d \sim \frac{1}{d}\left[X_1 + \cdots + X_d\right]. \tag{9.26}$$

X_i are independent and identically distributed realizations of X. Obviously, we also have $\mathbb{E}(\overline{X}_d) = ||A||_F^2$ as well as

$$\begin{aligned} \mathbb{V}(\overline{X}_d) &= \frac{1}{d}\mathbb{V}(X) \\ &= \frac{2\,||A||_{s,4}^4}{d}. \end{aligned} \tag{9.27}$$

The above result is well-known about the variance of independent random variables.

We now seek a concentration inequality for $\overline{X}_d$. To do this, we need Chernoff's Inequality [11], which is useful for bounding tail probabilities:

Theorem 9.6 (Chernoff's Inequality; Theorem 3.2.2 in [11])
Given a random variable X, we have

$$\mathbb{P}\left[X \geq a\right] \leq \min_{t>0} e^{-ta}\,\mathbb{E}\left(e^{tX}\right). \tag{9.28}$$

and

$$\mathbb{P}\left[X \leq a\right] \leq \min_{t>0} e^{ta}\,\mathbb{E}\left(e^{-tX}\right). \tag{9.29}$$

Here, $\mathbb{E}\left(e^{tX}\right)$ is the moment generating function for the random variable X. We will use these inequalities to prove the tail probabilities of $\overline{X}_d$ decay exponentially in d, ensuring independent realizations of Eq. (9.13) are close to the expected value. This result and proof were published in [33] but we flesh out some of the details not included due to length considerations.

Theorem 9.7 (Probabilistic Error Bounds)
Given $\overline{X}_d$ as defined in Eq. (9.26) with $r \geq 2$, the following bounds on the tail probabilities hold:

$$\begin{aligned}
&\mathbb{P}\left[\overline{X}_d \geq ||A||_F^2\,\mu\right] \leq \exp\left(-\frac{d\mu}{2}\right) ||A||_F^{dr} \prod_{k=1}^{r} (A_k')^{-d} \qquad \mu > 1 \\
&\mathbb{P}\left[\overline{X}_d \leq ||A||_F^2\,\mu\right] \leq \exp\left(\frac{d\mu}{2}\right) ||A||_F^{dr} \prod_{k=1}^{r} (A_k'')^{-d} \qquad \mu \in [0,1).
\end{aligned} \tag{9.30}$$

Here,

$$\begin{aligned}
||A||_F^2 &= \sigma_1^2 + \cdots + \sigma_r^2 \\
(A_k')^2 &= ||A||_F^2 - \sigma_k^2 \\
(A_k'')^2 &= ||A||_F^2 + \sigma_k^2.
\end{aligned} \tag{9.31}$$

We know $\mathbb{E}\left(\overline{X}_d\right) = ||A||_F^2$, so μ controls multiplicative deviation above or below the expectation value. Furthermore, if

$$\begin{aligned}
\nu_k &= -\ln\left[1 - \frac{\sigma_k^2}{||A||_F^2}\right] \\
\lambda_k &= \ln\left[1 + \frac{\sigma_k^2}{||A||_F^2}\right],
\end{aligned} \tag{9.32}$$

then

$$\mathbb{P}\left[\overline{X}_d \geq ||A||_F^2\,\mu\right] \leq \exp\left[-\frac{d\mu}{2}\left(\mu - \{\nu_1 + \cdots + \nu_r\}\right)\right] \tag{9.33}$$

decays exponentially in d when

$$\mu > 1 + \frac{||A||_2^2}{||A||_F^2 - ||A||_2^2}. \tag{9.34}$$

Similarly,

$$\mathbb{P}\left[\overline{X}_d \leq ||A||_F^2 \,\mu\right] \leq \exp\left[-\frac{d}{2}\left(\{\lambda_1 + \cdots + \lambda_r\} - \mu\right)\right] \tag{9.35}$$

decays exponentially in d when $\mu \in [0, \ln 2)$.

Proof. When $r = 1$ (that is, when A is a rank-one matrix), we can use Thm. 9.6 to compute exponential bounds on tail probabilities. Because these bounds can be computed exactly, we assume $r \geq 2$.

Clearly X is a linear combination of chi-squared distributions, so by properties of the moment generating function we have

$$M_{\overline{X}_d}(t) = \prod_{k=1}^{r}\left(1 - \frac{2\sigma_k^2}{d}t\right)^{-\frac{d}{2}}. \tag{9.36}$$

Attempting to compute

$$\min_{t>0} e^{-at} M_{\overline{X}_d}(t) \tag{9.37}$$

will require factoring the roots of a degree r polynomial in t; a standard result of Galois theory is that this is not possible in general for $r \geq 5$. Instead, we set

$$\bar{t} = \frac{d}{2\,||A||_F^2}. \tag{9.38}$$

Then, if we set $a = \mu\,||A||_F^2$ for $\mu > 1$, we have

$$\begin{aligned}
\mathbb{P}\left[\overline{X}_d \geq \mu\,||A||_F^2\right] &\leq \min_{t>0} \exp\left(-\mu\,||A||_F^2\, t\right) M_{\overline{X}_d}(t) \\
&\leq \exp\left(-\mu\,||A||_F^2\, \bar{t}\right) M_{\overline{X}_d}(\bar{t}) \\
&= \exp\left(-\frac{\mu d}{2}\right) \prod_{k=1}^{r}\left[1 - \frac{\sigma_k^2}{||A||_F^2}\right]^{-\frac{d}{2}} \\
&= \exp\left(-\frac{\mu d}{2}\right) ||A||_F^{rd} \prod_{k=1}^{r} (A'_k)^{-d}.
\end{aligned} \tag{9.39}$$

Here, A'_k is as defined in Eq. (9.31). Similarly, for $\mu \in [0, 1)$ and $a = \mu\,||A||_F^2$, we have

$$\begin{aligned}\mathbb{P}\left[\overline{X}_d \le \mu \left|\left|A\right|\right|_F^2\right] &\le \min_{t>0} \exp\left(\mu \left|\left|A\right|\right|_F^2 t\right) M_{\overline{X}_d}(-t) \\ &\le \exp\left(\mu \left|\left|A\right|\right|_F^2 \bar{t}\right) M_{\overline{X}_d}(-\bar{t}) \\ &= \exp\left(\frac{\mu d}{2}\right) \prod_{k=1}^{r} \left[1 + \frac{\sigma_k^2}{\left|\left|A\right|\right|_F^2}\right]^{-\frac{d}{2}} \\ &= \exp\left(\frac{\mu d}{2}\right) \left|\left|A\right|\right|_F^{rd} \prod_{k=1}^{r} \left(A_k''\right)^{-d}. \end{aligned} \tag{9.40}$$

A_k'' is defined in Eq. (9.31). This proves the desired bounds as stated in the theorem. Stronger tail probability bounds could be determined but we will not pursue the matter here, for to do so may require knowledge of the singular value decay.

We now focus on determining when the tail probabilities decay exponentially in d. Looking at the upper tail probability, we have

$$\begin{aligned}\mathbb{P}\left[\overline{X}_d \ge \mu \left|\left|A\right|\right|_F^2\right] &\le \exp\left(-\frac{\mu d}{2}\right) \left|\left|A\right|\right|_F^{rd} \prod_{k=1}^{r} \left(A_k'\right)^{-d} \\ &= \exp\left(-\frac{d}{2}\left[\mu - \{\nu_1 + \cdots + \nu_r\}\right]\right), \end{aligned} \tag{9.41}$$

where

$$\nu_k = -\ln\left[1 - \frac{\sigma_k^2}{\left|\left|A\right|\right|_F^2}\right]. \tag{9.42}$$

We will have exponential tail probability decay in d if

$$\nu_1 + \cdots + \nu_r < \mu. \tag{9.43}$$

We know $-\ln x$ is a convex function, so $-\ln(1-x)$ is convex on $[0,1)$. For any $\alpha \in (0,1)$, we have

$$\ln\left(\frac{1}{1-x}\right) \le \frac{x}{\alpha} \ln\left(\frac{1}{1-\alpha}\right). \tag{9.44}$$

Because $r \ge 2$, we have $\sigma_1 = \left|\left|A\right|\right|_2 < \left|\left|A\right|\right|_F$ and set $\alpha = \frac{\left|\left|A\right|\right|_2^2}{\left|\left|A\right|\right|_F^2} < 1$. It now follows that

$$\begin{aligned}\nu_1 + \cdots + \nu_r &\leq \frac{1}{\alpha} \ln \left(\frac{1}{1-\alpha} \right) \left[\frac{\sigma_1^2}{||A||_F^2} + \cdots + \frac{\sigma_r^2}{||A||_F^2} \right] \\ &= \frac{1}{\alpha} \ln \left(1 + \frac{\alpha}{1-\alpha} \right) \\ &\leq 1 + \frac{\alpha}{1-\alpha} \\ &= 1 + \frac{||A||_2^2}{||A||_F^2 - ||A||_2^2}, \end{aligned} \tag{9.45}$$

where we used $\ln(1+x) \leq x$ in the last inequality. So long as

$$\mu > 1 + \frac{||A||_2^2}{||A||_F^2 - ||A||_2^2}, \tag{9.46}$$

we have exponential decay in d.

We now look at the lower tail probability. For $\mu \in [0, 1)$, we have

$$\begin{aligned}\mathbb{P}\left[\overline{X}_d \leq ||A||_F^2 \, \mu\right] &\leq \exp\left(\frac{d\mu}{2}\right) ||A||_F^{dr} \prod_{k=1}^{r} (A''_k)^{-d} \\ &= \exp\left[\frac{d}{2}\left\{\mu - (\lambda_1 + \cdots + \lambda_r)\right\}\right], \end{aligned} \tag{9.47}$$

where

$$\lambda_k = \ln\left[1 + \frac{\sigma_k^2}{||A||_F^2}\right]. \tag{9.48}$$

To have exponential decay in probability, we require

$$\lambda_1 + \cdots + \lambda_r > \mu\,. \tag{9.49}$$

Now, we know

$$\ln(1+x) \geq x \ln 2 \quad x \in [0,1]\,, \tag{9.50}$$

which implies

$$\begin{aligned}\lambda_1 + \cdots + \lambda_r &\geq \frac{\sigma_1^2}{||A||_F^2} \ln 2 + \cdots + \frac{\sigma_r^2}{||A||_F^2} \ln 2 \\ &= \ln 2. \end{aligned} \tag{9.51}$$

Therefore, so long as $\mu < \ln 2$, we have exponentially decaying tail probabilities in d.

□

We now analyze the situation for power iteration. Let $\alpha \geq \frac{1}{2}$ and define

$$Y_\alpha = \sigma_1^{2\alpha} \xi_1^2 + \cdots + \sigma_r^{2\alpha} \xi_r^2. \tag{9.52}$$

Additionally, let

$$\overline{Y}_{d,\alpha} = \frac{1}{d} \left[Y_{1,\alpha} + \cdot + Y_{d,\alpha} \right], \tag{9.53}$$

where $Y_{i,\alpha}$ are independent and identically distributed realizations of Y_α. Naturally, $\mathbb{E}(Y_\alpha) = \mathbb{E}(\overline{Y}_{d,\alpha}) = ||A||_{s,2\alpha}^{2\alpha}$ and it clear we have

$$\begin{aligned}\mathbb{V}(\overline{Y}_{d,\alpha}) &= \frac{1}{d} \mathbb{V}(Y_\alpha) \\ &= \frac{2\, ||A||_{s,4\alpha}^{4\alpha}}{d}. \end{aligned} \tag{9.54}$$

We have the analogous theorem:

Theorem 9.8 (Probabilistic Error Bounds for Power Iteration)

Given $\overline{Y}_d$ as defined in Eq. (9.53) with $r \geq 2$ and

$$\begin{aligned}\bar{\nu}_k &= -\ln \left[1 - \frac{\sigma_k^{2\alpha}}{||A||_{s,2\alpha}^{2\alpha}} \right] \\ \bar{\lambda}_k &= \ln \left[1 + \frac{\sigma_k^{2\alpha}}{||A||_{s,2\alpha}^{2\alpha}} \right], \end{aligned} \tag{9.55}$$

then

$$\mathbb{P}\left[\overline{Y}_d \geq ||A||_{s,2\alpha}^{2\alpha}\,\mu\right] \leq \exp\left[-\frac{d\mu}{2}\left(\mu - \{\bar{\nu}_1 + \cdots + \bar{\nu}_r\}\right)\right] \tag{9.56}$$

decays exponentially in d when

$$\mu > 1 + \frac{||A||_{s,\infty}^{2\alpha}}{||A||_{s,2\alpha}^{2\alpha} - ||A||_{2,\infty}^{2\alpha}}. \tag{9.57}$$

Similarly,

$$\mathbb{P}\left[\overline{Y}_d \leq ||A||_{s,2\alpha}^{2\alpha}\,\mu\right] \leq \exp\left[-\frac{d}{2}\left(\{\bar{\lambda}_1 + \cdots + \bar{\lambda}_r\} - \mu\right)\right] \tag{9.58}$$

decays exponentially in d when $\mu \in [0, \ln 2)$.

Proof. The proof follows the same argument as that of Thm. 9.7. We obtain the results of Thm. 9.7 when $\alpha = 1$, as expected. □

The above theorem ensures that independent realizations of Eq. (9.16) closely match the expected value. This will make it possible for us to bound $||\cdot||_{s,p}$ depending on the power as well as allowing for a relative stopping criterion in these norms.

9.7 Stopping Criteria Comparison

In this section we compare the different error bounds and algorithms that have been discussed previously by running a set of tests.

9.7.1 Matrix Types

We will be testing our algorithms on a few different classes of matrices: algebraic decay, exponential decay, S-shaped decay, and a Devil's staircase. These matrices are chosen because they are similar to those in [75]. All these matrices have the form UDV^*, where U and V are random orthogonal matrices and D contains positive diagonal entries which vary depending on the matrix. All matrices have $||A||_2 \approx 1$ and $U, V \in \mathbb{R}^{1000\times 100}$.

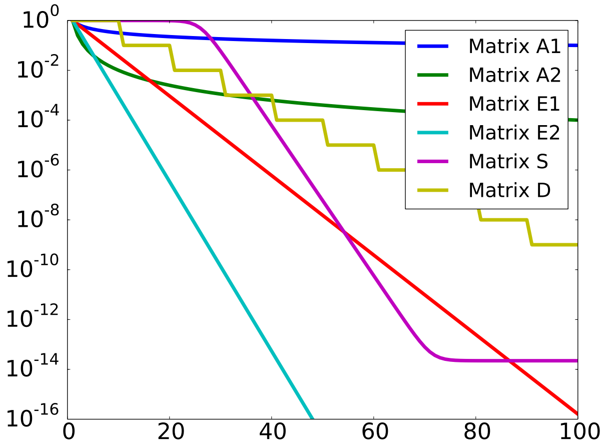


Figure 9.1: A plot of the singular values for the matrices we are investigating.

Specifically, we choose the following for D:

- Algebraic decay: $d_k = k^{-\beta}$ for $\beta \in \{0.5, 2\}$. We will refer to these matrices as Matrix A1 and Matrix A2.

- Exponential decay: $d_k = 2^{-\beta(k-1)/100}$ for $\beta \in \{52, 113\}$. This leads to exponential decay with values ending approximately at 10^{-16} and 10^{-34}. To machine precision using double arithmetic, the second matrix has rank 46. We will refer to these matrices as Matrix E1 and Matrix E2.

- S-shaped decay: $d_k = 100\varepsilon_{\text{mach}} + \left[1 + 2^{k-26}\right]^{-1}$. We will call this Matrix S.

- Devil's Staircase: $d_{10(k-1)+j} = 10^{1-k}$ for $j \in \{1, 2, \cdots, 10\}$ and $k \in \{1, 2, \cdots, 10\}$; that is, we have 10 singular values of value $1 = 10^0$, 10 singular values of value 10^{-1}, continuing until we have 10 singular values of value 10^{-9}. We will call this Matrix D.

A plot of the singular values can be seen in Fig. 9.1.

9.7.2 Norm Approximation

We begin by showing how poorly the HMT error bound from Lem. 9.2 is by using it to estimate $||A||_2$; the results are shown in Table 9.2, and we remember $||A||_2 \approx 1$. We let

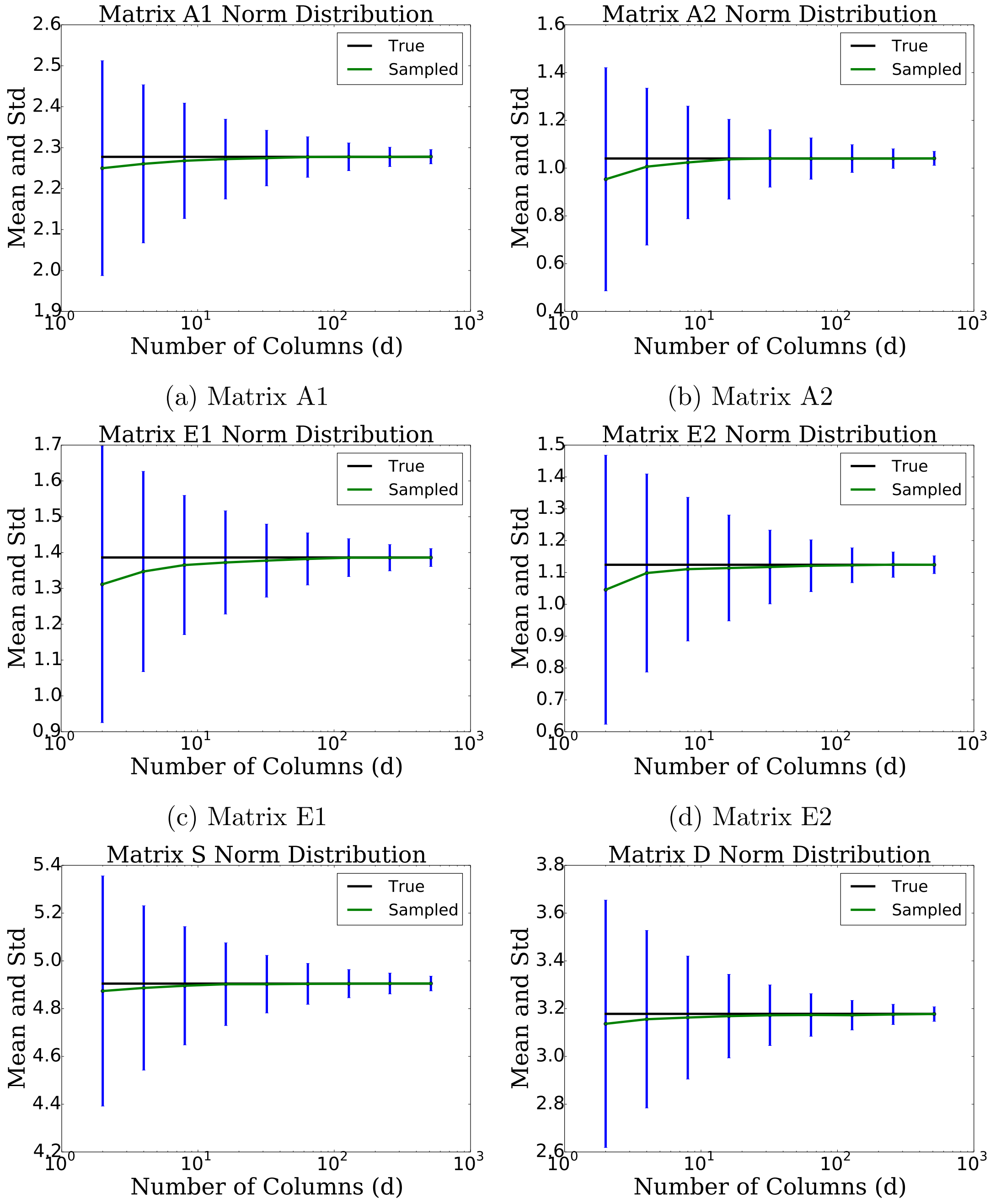
$\alpha \in \{2, 5, 10\}$ and choose the number of samples p so that $\alpha^p \leq 10^{-\ell}$ for various ℓ. To allow for comparisons, we let $\alpha^p \in \{10^{-9}, 10^{-12}, 10^{-15}\}$. It is clear that in *every* case the norm is overestimated, always 3× and frequently 10× larger; furthermore, smaller α always leads to a more accurate estimate. As we can see, we greatly overestimate the norm. We are able to accurately measure the F-norm as seen in Fig. 9.2 using our stochastic estimate; note the logarithmic scale on the number of columns (d). We also include estimates of the squared F-norm in Fig. 9.3 including mean and standard deviations as well as the theoretical standard deviations; the predicted results accurately match expected results. We averaged 10,000 trials to obtain the statistics for these results. In order to obtain the theoretical standard deviations in Fig. 9.3, we took the square root of the variance from Eq. (9.27).

9.7.3 Adaptive Comparison

We now compare some of the adaptive low-rank algorithms; in particular, we use Alg. 5 with the HMT stopping criterion, the YGL algorithm in Alg. 7, and the new randomized low-rank compression algorithm in Alg. 8. Once each algorithm has determined enough random samples have been computed, a pivoted QR factorization is computed on the entire batch and the factorization is truncated below the specified tolerance. This last part (pivoted QR) is not an explicit feature of Algs. 5–7 but is added due to it being necessary in the situation where our new algorithm was developed [33]. These results can be found in Table 9.3; the minimum samples required to meet the specified tolerance (as determined by the SVD) are given in Table 9.4. The averaged number of random samples used is computed as well as the average 2-norm error taken over 1,000 trials. While the YGL and new algorithm bounds are related to the F-norm, the 2-norm is of primary importance. Random samples are computed in blocks of 16 random samples and, due to the nature of our new algorithm, the minimum possible number of samples used in Alg. 8 is 32: 16 samples for initial Q and 16 samples for the error estimate. The

		1E-9			**1E-12**			**1E-15**		
		p	M	S	p	M	S	p	M	S
Matrix A1	**2**	30	5.11	0.53	40	5.23	0.52	50	5.32	0.52
	5	13	11.8	1.4	18	12.2	1.4	22	12.4	1.3
	10	9	22.9	2.8	12	23.5	2.8	15	24.1	2.7
Matrix A2	**2**	30	3.73	0.71	40	3.90	0.69	50	4.03	0.68
	5	13	8.02	1.93	18	8.52	1.86	22	8.85	1.82
	10	9	14.8	4.0	12	15.8	3.9	15	16.5	3.8
Matrix E1	**2**	30	4.11	0.64	40	4.26	0.63	50	4.38	0.62
	5	13	9.14	1.72	18	9.58	1.68	22	9.86	1.66
	10	9	17.2	3.6	12	18.0	3.5	15	18.7	3.4
Matrix E2	**2**	30	3.81	0.70	40	3.98	0.68	50	4.10	0.67
	5	13	8.27	1.90	18	8.78	1.83	22	9.07	1.79
	10	9	15.3	3.94	12	16.3	3.84	15	17.0	3.74
Matrix S	**2**	30	10.1	0.6	40	10.2	0.6	50	10.3	0.6
	5	13	24.1	1.6	18	24.5	1.6	22	24.8	1.5
	10	9	47.0	3.5	12	47.9	3.3	15	48.5	3.2
Matrix D	**2**	30	7.36	0.64	40	7.51	0.61	50	7.61	0.60
	5	13	17.2	1.8	18	17.7	1.7	22	18.0	1.6
	10	9	33.3	3.7	12	34.2	3.6	15	34.8	3.5

Table 9.2: Upper bound of $||A||_2$ based on Lemma 9.2 (HMT) for different failure probabilities (columns) and α (rows). p is the smallest integer for which $\alpha^{-p} \leq 10^{-\ell}$. We performed 10,000 trials and computed the mean (M) and standard deviation (S). The correct values are $||A||_2 \approx 1$.



(a) Matrix A1

(b) Matrix A2

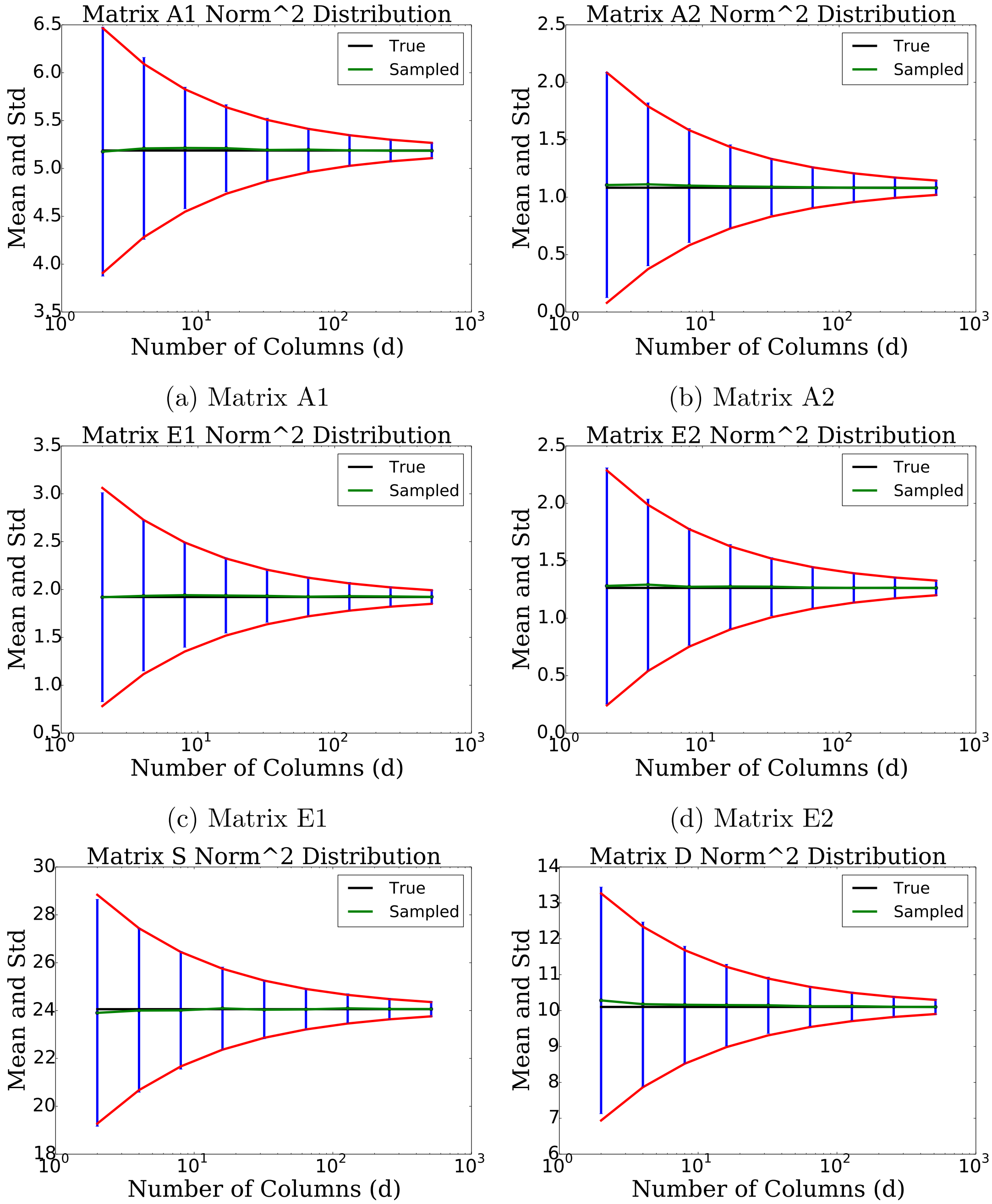
(c) Matrix E1

(d) Matrix E2

(e) Matrix S

(f) Matrix D

Figure 9.2: We performed 10,000 trials and computed the mean (Green) and standard deviation (Blue) of $||A\Omega||_F$ for columns $d \in \{2, 4, 8, \cdots, 512\}$. The true F-norm value is Black.



(a) Matrix A1

(b) Matrix A2

(c) Matrix E1

(d) Matrix E2

(e) Matrix S

(f) Matrix D

Figure 9.3: We performed 10,000 trials and computed the mean (Green) and standard deviation (Blue) of $||A\Omega||_F^2$ for columns $d \in \{2, 4, 8, \cdots, 512\}$. The true squared F-norm value is Black and the theoretical standard deviation bounds from Eq. (9.27) are Red.

YGL algorithm stops if we compute $E \leq 0$.

From the results, we see that the HMT bound always performs the worst in every case; this is expected from it overestimating the 2-norm. Outside of this, the YGL and GEB algorithms are close. Matrix A1 is particularly challenging: it has slow singular value decay and our algorithm performs no power iteration. For this matrix, YGL does better than GEB in every case (they perform equally well with a relative error of 0.1), in part because knowing the error exactly is important. For Matrix A2, YGL does better for 0.1 relative error, with GEB using fewer or equal samples in the other cases. Matrices E1, E2, and S are similar: YGL and GEB are about the same for large relative errors but, due to the inherent error bound restrictions, the new algorithm is better for smaller tolerances. For Matrix D, YGL did a little better than GEB. Overall, the results here show that the YGL stopping criterion may be slightly better than the GEB stopping criterion when for $\varepsilon_{\text{rel}} \geq O(\sqrt{\varepsilon_{\text{mach}}})$, but our stochastic F-norm bound allows us to determine relative errors to smaller tolerances.

We now perform more runs, except here we only look at the new stopping criterion; we set the blocksize to 5 and change the relative tolerances. Results are shown in Table 9.5, and Table 9.6 contains the minimum required ranks. The relative tolerances here are difficult and are meant to test the limits of our stopping criterion. For Matrix E1, we are able to have relative tolerances down to 5E-15 without having any problems with the stopping criterion (that is, every trial determined 100 samples was sufficient for the specified tolerance); some trials failed to satisfy the stopping criterion for 1E-15 and no trials satisfied the stopping criterion for 5E-16. These difficulties are expected because we are getting close to the limitations of machine precision ($\varepsilon_{\text{mach}} \approx 10^{-16}$ for double precision). For Matrix S, we note that we use an average of 107 samples for the tolerance 1E-14, which is more than the 100 samples that should be sufficient ideally. For Matrix D, we used **single precision**. We did not have any problems with relative tolerances down to 5E-6; for the tolerance 1E-6, most trials (738/1000) failed to satisfy the stopping criterion. Again, this is expected because $\varepsilon_{\text{mach}} \approx 10^{-7}$ in single precision. Overall, we see that our new stopping criterion allows us to get close to machine precision, which is not the case when using the HMT or YGL algorithms.

9.7.4 Stopping Criteria Discussion

The results of this section show our new stopping criterion does well, usually being on par or better than other stopping criteria while allowing for small tolerances. One noticeable difficultly, shared by others, is using too many additional samples to perform a low-order approximation ($\varepsilon_{\text{rel}} \gtrsim 0.1$) for matrices with slow decay without using power iteration; part of the difficulty comes from using F-norm bounds when desiring 2-norm accuracy. Outside of this limited range, the error closely matches the specified tolerance, keeping communication-heavy computations to one rank-revealing QR factorization. Additionally, we showed our stopping criterion allows us to approximate matrices to relative tolerances close to machine precision.

	0.75		**0.5**		**0.25**		**0.1**	
Matrix A1	Err	Samp	Err	Samp	Err	Samp	Err	Samp
HMT	0.19	128	0.16	128	0.13	128	3E-15	128
YGL	0.45	20	0.24	59	0.16	96	3E-3	112
GEB	0.22	77	0.18	94	0.15	102	5E-3	112

	1E-1		**1E-2**		**1E-3**		**1E-4**	
Matrix A2	Err	Samp	Err	Samp	Err	Samp	Err	Samp
HMT	4E-2	42	3E-3	92	3E-4	128	2E-15	128
YGL	4E-2	16	5E-3	32	4E-4	96	2E-15	112
GEB	4E-2	32	5E-3	32	4E-4	80	2E-15	112

	1E-3		**1E-6**		**1E-9**		**1E-12**	
Matrix E1	Err	Samp	Err	Samp	Err	Samp	Err	Samp
HMT	4E-4	48	4E-7	76	4E-10	96	4E-13	112
YGL	6E-4	32	7E-7	48	1E-9	64	6E-10	64
GEB	6E-4	32	7E-7	48	1E-9	65	7E-13	94

	1E-3		**1E-6**		**1E-9**		**1E-12**	
Matrix E2	Err	Samp	Err	Samp	Err	Samp	Err	Samp
HMT	3E-4	32	3E-7	48	4E-10	53	3E-13	82*
YGL	6E-4	16	4E-7	32	8E-10	32	1E-10	32
GEB	3E-4	32	4E-7	32	8E-10	32	5E-13	48

	1E-3		**1E-6**		**1E-9**		**1E-12**	
Matrix S	Err	Samp	Err	Samp	Err	Samp	Err	Samp
HMT	4E-4	64	3E-7	80	4E-10	80	3E-13	200+
YGL	5E-4	48	2E-6	48	6E-10	64	2E-11	64
GEB	5E-4	48	9E-7	59	6E-10	64	6E-13	80

	1E-1		**1E-3**		**1E-5**		**1E-7**	
Matrix D	Err	Samp	Err	Samp	Err	Samp	Err	Samp
HMT	3E-2	51	4E-4	80	4E-6	96	4E-8	112
YGL	9E-2	27	6E-4	48	1E-5	64	2E-7	85
GEB	5E-2	32	6E-4	48	9E-6	69	6E-8	96

Table 9.3: QB approximation results for Matrices A1, A2, E1, E2, S, and D. For each absolute error tolerance, we averaged 1,000 trials to determine the average error (Err) and average samples used (Samp) in order to compute a QB approximation once we used either the HMT stopping criterion, the YGL stopping criterion, or the new stopping criterion to determine when we had approximated the range. Random samples were computed in blocks of 16. In the case when we used 200+ samples, we were not able to meet the HMT stopping criterion and used the maximum of 200 random samples. Some of the data here was originally in [33]. The minimum possible ranks can be found in Table 9.4. A "*" means that there were some runs which reached the maximum allowable samples of 200 before compression.

Matrix A1	**0.75**	**0.5**	**0.25**	**0.1**
Min Rank	1	4	16	100
Matrix A2	**1E-1**	**1E-2**	**1E-3**	**1E-4**
Min Rank	3	10	31	100
Matrix E1	**1E-3**	**1E-6**	**1E-9**	**1E-12**
Min Rank	19	38	57	76
Matrix E2	**1E-3**	**1E-6**	**1E-9**	**1E-12**
Min Rank	9	18	27	36
Matrix S	**1E-3**	**1E-6**	**1E-9**	**1E-12**
Min Rank	35	45	55	65
Matrix D	**1E-1**	**1E-3**	**1E-5**	**1E-7**
Min Rank	20	40	60	80

Table 9.4: Here are the absolute minimum ranks required for Matrices A1, A2, E1, E2, S, and D.

	1E-14		**5E-15**		**1E-15**		**5E-16**	
Matrix E1	Err	Samp	Err	Samp	Err	Samp	Err	Samp
GEB	1E-14	97	4E-15	100	6E-16	144*	7E-16	200+
	5E-13		**1E-13**		**4E-14**		**1E-14**	
Matrix S	Err	Samp	Err	Samp	Err	Samp	Err	Samp
GEB	4E-13	75	7E-14	96	5E-14	100	9E-15	107
	5E-5		**1E-5**		**5E-6**		**1E-6**	
Matrix D	Err	Samp	Err	Samp	Err	Samp	Err	Samp
GEB (s)	4E-5	59	9E-6	66	9E-6	71	6E-7	186*

Table 9.5: Tough QB approximation results for Matrices E1, S, and D using the new GEB stopping criterion. For each absolute error tolerance, we averaged 1,000 trials to determine the average error (Err) and average samples used (Samp) in order to compute a QB approximation once we used the new stopping criterion to determine when we had approximated the range. Random samples were computed in blocks of 5. For matrix D, we used **single precision**. In the case when we used 200+ samples, we were not able to meet the HMT stopping criterion and used the maximum of 200 random samples. The minimum possible ranks can be found in Table 9.6. A "*" means that there were some runs which reached the maximum allowable samples of 200 before compression. For matrix E1 with ε_{rel}=1E-15, 7.7% of the trials used over 200 samples (and so failed to stop); for matrix D with ε_{rel}=1F-6, 73.8% of the trials used over 200 samples.

Matrix E1	**1E-14**	**5E-15**	**1E-15**	**5E-16**
Min Rank	88	90	95	96

Matrix S	**5E-13**	**1E-13**	**5E-14**	**1E-14**
Min Rank	66	69	71	100

Matrix D	**5E-5**	**1E-5**	**5E-6**	**1E-6**
Min Rank	50	60	60	70

Table 9.6: Here are the absolute minimum ranks required for Matrices E1, S, and D.

Algorithm 8 Randomized Block Low-Rank Approximation (New)

```
function RANDOM_NEW_LOW_RANK($A$,$d$,$\varepsilon_{\text{abs}}$,$\varepsilon_{\text{rel}}$)    ▷ Builds $QB$ approximation
    $Q = []$; $n = \texttt{cols}(A)$
    $\Omega_i = \texttt{rand}(n, d)$, $S_i = A\Omega_i$ for $i \in \{1, 2\}$    ▷ $d$ is block size
    $S = \begin{bmatrix} S_1 & S_2 \end{bmatrix}$
    $[Q_1, R_1] = \texttt{qr}(S_1)$; $Q = Q_1$
    $\rho = ||S_1||_F / \sqrt{d}$
    if $\min_j |(R_1)_{jj}| < \max(\varepsilon_{\text{abs}}, \varepsilon_{\text{rel}}\rho)$ then
        loop_bool = false    ▷ $S_1$ is numerically rank-deficient
    else
        loop_bool = true
    end if
    $k = 1$
    while loop_bool do
        $k = k + 1$
        $\widehat{S}_k = (I - QQ^*)^2 S_k$    ▷ 2× GS orthogonalization for numerical stability
        if $||\widehat{S}_k||_F < \max(\varepsilon_{\text{abs}}\sqrt{d}, \varepsilon_{\text{rel}}||S_k||_F)$ then
            break    ▷ Range of $A$ is known to specified tolerance
        end if
        $\Omega_{k+1} = \texttt{rand}(n, d)$; $S_{k+1} = A\Omega_{k+1}$
        $S = \begin{bmatrix} S & S_{k+1} \end{bmatrix}$
        $[Q_k, R_k] = \texttt{qr}(\widehat{S}_k)$
        if $\min_j |(R_k)_{jj}| < \max(\varepsilon_{\text{abs}}, \varepsilon_{\text{rel}}\rho)$ then
            break    ▷ $\widehat{S}_k$ is numerically rank-deficient
        end if
        $Q = \begin{bmatrix} Q & Q_k \end{bmatrix}$
    end while
    $Q = \texttt{rrqr}(S, \varepsilon_{\text{abs}}, \varepsilon_{\text{rel}})$
    $B = Q^*A$
    return $Q$, $B$    ▷ We have the approximation $A \approx QB$
end function
```

Chapter 10

Conclusion

10.1 Discussion of Results and Future Directions for MSN

In this dissertation we investigated the structure of Chebyshev-Vandermonde matrices; by doing so, we were able to develop fast algorithms for solving problems in interpolation and ordinary differential equation boundary value problems. We showed examples and determined that the results were on par with many methods as well as, at times, producing more accurate approximations. Under certain circumstances, we were able to prove that our methods converge to the underlying solution.

There are multiple directions where this work could continue. First, we could look into implementing fast algorithms in 2D and 3D in order to see there are more advantages with MSN over standard Chebyshev interpolation for smooth functions or against standard filters for rough functions. Additionally, we could implement the ODE BVP fast solver using fast algorithms for SSS or HSS matrices. From there, we could look into solving 2D and 3D elliptic boundary value problems. In this case, it will be interesting to see if the methods we used in 1D variable coefficients here could be extended to 2D and 3D variable coefficients. The difficulty will come from the fact that in 2D, we will be summing tensor products of three terms.

While interpolation on Chebyshev nodes is fast because of the DCT and IDCT, this could also hold in general, such as interpolating on Legendre nodes using Legendre polynomials as a basis and performing MSN in this basis. On the other hand, there is still structure in the Gram matrix $VD_s^{-2}V^*$, and it may be possible to invert this quickly by converting it into HSS form exactly for certain s values. Doing so would require knowledge of structured matrices and solvers, but should speed up general interpolation and make it useful. It would be interesting to see if this could be used in 2D interpolation.

10.2 Discussion of Results and Future Directions for Randomized Low-Rank Approximations

In Chapter 9 we investigated the blocked form of the low-rank fixed-precision problem in order to develop a stopping criterion useful for relative tolerances down to machine precision. We were able to prove asymptotically that this method would produce an accurate approximation to the Frobenius norm of a matrix, a critical part of the relative stopping criterion. The examples showed our method usually produces a sufficient number of random samples, though not too many, in order to reach the desired 2-norm error.

The stochastic error bound that we developed in the F-norm could be improved. First, it would be useful to determine if we can find a random variable $f(Ax)$ with $\mathbb{E}f(Ax) = ||A||_2^\alpha$ for some power α; that is, we want to find a way to combine random samplings of the range of a matrix in such a way that the expected value of the random variable is (some power of) the matrix 2-norm. Most, if not all, would prefer to bound the 2-norm rather than the F-norm, and yet accurately approximating the 2-norm is a challenge.

In our low-rank approximations, the final computation before returning our low-rank approximation was computing a pivoted QR factorization. A BLAS-3 level QR with Column Pivoting is available in LAPACK [2, 54], yet it is well-known that this can fail for some matrices [42]; this has lead to research into better pivoting strategies [14, 19]. The best appears to be the Strong Rank-Revealing QR factorization from [37], yet there is no known efficient implementation of this algorithm. It would be useful to first develop a BLAS-2, and then BLAS-3, version. The main difficulty comes from the fact that the pivoting strategy is more involved and requires more communication. This required communication is particularly challenging in distributed memory machines and there has been work to trade communication for flops; see [23, 24, 44, 63] for some examples. Even without this, in problems where there is a hierarchical chain of QR factorizations, like those in Randomized HSS constructions [33], better pivoting at the lowest level would lead to smaller ranks overall. An implementation of SRRQR would likely require multiple passes: the first pass could either be a BLAS-3 QR (not pivoted) or BLAS-3 pivoted QR; the second pass could be a blocked version of the SRRQR algorithm acting on adjacent panels; a final BLAS-2 pass could be performed to check for any final pivots. While this would be complicated, it would be useful for the entire computing community as rank-revealing QR factorizations are essential.

Bibliography

[1] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Vol. 55. Courier Corporation, 1965.

[2] Edward Anderson et al. *LAPACK Users' guide*. Vol. 9. Siam, 1999.

[3] Uri M Ascher, Robert MM Mattheij, and Robert D Russell. *Numerical solution of boundary value problems for ordinary differential equations*. Vol. 13. Siam, 1994.

[4] Uri M Ascher and Linda R Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. Vol. 61. Siam, 1998.

[5] Jared L Aurentz and Lloyd N Trefethen. "Block operators and spectral discretizations". In: *SIAM Review* 59.2 (2017), pp. 423–446.

[6] Jeff Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1 (2017), pp. 65–98.

[7] Rajendra Bhatia. *Matrix analysis*. Vol. 169. Springer Science & Business Media, 2013.

[8] David Bindel et al. "On computing Givens rotations reliably and efficiently". In: *ACM Transactions on Mathematical Software (TOMS)* 28.2 (2002), pp. 206–238.

[9] Å. Björck. "Numerics of Gram-Schmidt orthogonalization". In: *Linear Algebra and its Applications* 197-198 (1994), pp. 297–316. ISSN: 0024-3795.

[10] John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[11] P. Brémaud. *Discrete Probability Models and Methods: Probability on Graphs and Trees, Markov Chains and Random Fields, Entropy and Coding*. Probability Theory and Stochastic Modelling. Springer International Publishing, 2017. ISBN: 9783319434766.

[12] Susanne Brenner and Ridgway Scott. *The mathematical theory of finite element methods*. Vol. 15. Springer Science & Business Media, 2007.

[13] L. Brutman. "On the Lebesgue Function for Polynomial Interpolation". In: *SIAM Journal on Numerical Analysis* 15.4 (1978), pp. 694–704. DOI: `10.1137/0715046`. eprint: `https://doi.org/10.1137/0715046`. URL: `https://doi.org/10.1137/0715046`.

[14] Tony F Chan. "Rank revealing QR factorizations". In: *Linear algebra and its applications* 88 (1987), pp. 67–82.

[15] S. Chandrasekaran, M. Gu, and W. Lyons. "A fast adaptive solver for hierarchically semiseparable representations". In: *CALCOLO* 42.3 (Dec. 2005), pp. 171–185. ISSN: 1126-5434.

[16] S. Chandrasekaran, K.R. Jayaraman, and H.N. Mhaskar. "Minimum Sobolev norm interpolation with trigonometric polynomials on the torus". In: *Journal of Computational Physics* 249.Supplement C (2013), pp. 96–112. ISSN: 0021-9991. DOI: `https://doi.org/10.1016/j.jcp.2013.03.041`. URL: `http://www.sciencedirect.com/science/article/pii/S0021999113002192`.

[17] Shiv Chandrasekaran, Ming Gu, and Timothy Pals. "A fast ULV decomposition solver for hierarchically semiseparable representations". In: *SIAM Journal on Matrix Analysis and Applications* 28.3 (2006), pp. 603–622.

[18] Shivkumar Chandrasekaran, CH Gorman, and Hrushikesh Narhar Mhaskar. "Minimum Sobolev norm interpolation of scattered derivative data". In: *Journal of Computational Physics* 365 (2018), pp. 149–172.

[19] Shivkumar Chandrasekaran and Ilse CF Ipsen. “On rank-revealing factorisations”. In: *SIAM Journal on Matrix Analysis and Applications* 15.2 (1994), pp. 592–622.

[20] Shivkumar Chandrasekaran and Hrushikesh Narhar Mhaskar. “A minimum Sobolev norm technique for the numerical discretization of PDEs”. In: *Journal of Computational Physics* 299 (2015), pp. 649–666.

[21] Shiv Chandrasekaran et al. “Some fast algorithms for sequentially semiseparable representations”. In: *SIAM Journal on Matrix Analysis and Applications* 27.2 (2005), pp. 341–364.

[22] Philip J Davis. *Interpolation and approximation.* Courier Corporation, 1975.

[23] James W Demmel et al. “Communication avoiding rank revealing QR factorization with column pivoting”. In: *SIAM Journal on Matrix Analysis and Applications* 36.1 (2015), pp. 55–89.

[24] Simplice Donfack, Laura Grigori, and Alok Kumar Gupta. “Adapting communication-avoiding LU and QR factorizations to multicore architectures”. In: *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS).* IEEE. 2010, pp. 1–10.

[25] Tobin A Driscoll and Nicholas Hale. “Rectangular spectral collocation”. In: *IMA Journal of Numerical Analysis* 36.1 (2015), pp. 108–132.

[26] Tobin A Driscoll, Nicholas Hale, and Lloyd N Trefethen. *Chebfun guide.* 2014.

[27] Harry Dym. *Linear algebra in action.* Vol. 78. American Mathematical Soc., 2013.

[28] P Erdos. “On some convergence properties of the interpolation polynomials”. In: *Annals of Mathematics* (1943), pp. 330–337.

[29] Mariano Gasca and Thomas Sauer. “On the history of multivariate polynomial interpolation”. In: *Numerical Analysis: Historical Developments in the 20th Century.* Elsevier, 2001, pp. 135–147.

[30] Mariano Gasca and Thomas Sauer. "Polynomial interpolation in several variables". In: *Advances in Computational Mathematics* 12.4 (2000), p. 377.

[31] Pieter Ghysels et al. "A robust parallel preconditioner for indefinite systems using hierarchical matrices and randomized sampling". In: *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2017, pp. 897–906.

[32] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Matrix Computations. Johns Hopkins University Press, 2012. ISBN: 9781421408590. URL: `https://books.google.com/books?id=5U-l8U3P-VUC`.

[33] Christopher Gorman et al. "Matrix-free construction of HSS representation using adaptive randomized sampling". In: *CoRR* abs/1810.04125 (2018). arXiv: `1810.04125`. URL: `http://arxiv.org/abs/1810.04125`.

[34] David Gottlieb and Chi-Wang Shu. "On the Gibbs phenomenon and its resolution". In: *SIAM review* 39.4 (1997), pp. 644–668.

[35] Ronald L Graham et al. "Concrete mathematics: a foundation for computer science". In: *Computers in Physics* 3.5 (1989), pp. 106–107.

[36] Ming Gu. "Subspace iteration randomization and singular value problems". In: *SIAM Journal on Scientific Computing* 37.3 (2015), A1139–A1173.

[37] Ming Gu and Stanley C Eisenstat. "Efficient algorithms for computing a strong rank-revealing QR factorization". In: *SIAM Journal on Scientific Computing* 17.4 (1996), pp. 848–869.

[38] MATLAB User's Guide. "The mathworks". In: *Inc., Natick, MA* 5 (1998), p. 333.

[39] Wolfgang Hackbusch. "A Sparse Matrix Arithmetic Based on H-Matrices. Part I: Introduction to H-Matrices". In: *Computing* 62.2 (1999), pp. 89–108.

[40] N. Halko, P. G. Martinsson, and J. A. Tropp. “Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions”. In: *SIAM Rev.* 53.2 (2011), pp. 217–288. ISSN: 0036-1445.

[41] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002, pp. xxx + 680. ISBN: 0-89871-521-0.

[42] William Kahan. “Numerical linear algebra”. In: *Canadian Mathematical Bulletin* 9.5 (1966), pp. 757–801.

[43] Samuel Karlin and John M Karon. “On Hermite-Birkhoff interpolation”. In: *Journal of Approximation Theory* 6.1 (1972), pp. 90–115.

[44] Amal Khabou et al. “LU factorization with panel rank revealing pivoting and its communication avoiding version”. In: *SIAM Journal on Matrix Analysis and Applications* 34.3 (2013), pp. 1401–1429.

[45] Konrad Knopp. *Theory and application of infinite series.* Courier Corporation, 2013.

[46] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems.* Vol. 98. Siam, 2007.

[47] Xiao Liu, Jianlin Xia, and Maarten V De Hoop. “Parallel randomized and matrix-free direct solvers for large structured dense linear systems”. In: *SIAM Journal on Scientific Computing* 38.5 (2016), S508–S538.

[48] Georg Gunther Lorentz and KL Zeller. “Birkhoff interpolation”. In: *SIAM Journal on Numerical Analysis* 8.1 (1971), pp. 43–48.

[49] RA Lorentz. “Multivariate Hermite interpolation by algebraic polynomials: A survey”. In: *Journal of computational and applied mathematics* 122.1-2 (2000), pp. 167–201.

[50] Rudolph A Lorentz. *Multivariate Birkhoff Interpolation*. Springer, 1992.

[51] Per-Gunnar Martinsson. “A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix”. In: *SIAM Journal on Matrix Analysis and Applications* 32.4 (2011), pp. 1251–1274.

[52] Per-Gunnar Martinsson and Sergey Voronin. “A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices”. In: *SIAM Journal on Scientific Computing* 38.5 (2016), S485–S507.

[53] Andrew M Odlyzko and Arnold Schönhage. “Fast algorithms for multiple evaluations of the Riemann zeta function”. In: *Transactions of the American Mathematical Society* 309.2 (1988), pp. 797–809.

[54] Gregorio Quintana-Ortí, Xiaobai Sun, and Christian H Bischof. “A BLAS-3 version of the QR factorization with column pivoting”. In: *SIAM Journal on Scientific Computing* 19.5 (1998), pp. 1486–1494.

[55] Theodore J Rivlin. *An introduction to the approximation of functions*. Courier Corporation, 2003.

[56] François-Henry Rouet et al. “A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization”. In: *ACM Transactions on Mathematical Software (TOMS)* 42.4 (2016), p. 27.

[57] Walter Rudin et al. *Principles of mathematical analysis*. Vol. 3. 4.2. McGraw-hill New York, 1976.

[58] Carl Runge. “Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten”. In: *Zeitschrift für Mathematik und Physik* 46.224-243 (1901), p. 20.

[59] A. Saibaba. “Randomized Subspace Iteration: Analysis of Canonical Angles and Unitarily Invariant Norms”. In: *SIAM Journal on Matrix Analysis and Applications* 40.1 (2019), pp. 23–48. DOI: 10.1137/18M1179432. eprint: https://doi.org/10.1137/18M1179432. URL: https://doi.org/10.1137/18M1179432.

[60] Arvind K Saibaba, Jonghyun Lee, and Peter K Kitanidis. “Randomized algorithms for generalized Hermitian eigenvalue problems with application to computing Karhunen–Loève expansion”. In: *Numerical Linear Algebra with Applications* 23.2 (2016), pp. 314–339.

[61] Jie Shen, Yingwei Wang, and Jianlin Xia. “Fast structured direct spectral methods for differential equations with variable coefficients, I. The one-dimensional case”. In: *SIAM Journal on Scientific Computing* 38.1 (2016), A28–A54.

[62] Simon J Smith. “Lebesgue constants in polynomial interpolation”. In: *Annales Mathematicae et Informaticae*. Vol. 33. 109-123. Eszterházy Károly College, Institute of Mathematics and Computer Science. 2006, pp. 1787–5021.

[63] Edgar Solomonik and James Demmel. “Communication-optimal parallel 2.5 D matrix multiplication and LU factorization algorithms”. In: *European Conference on Parallel Processing*. Springer. 2011, pp. 90–109.

[64] G.W. Stewart. “Block Gram-Schmidt Orthogonalization”. In: *SIAM Journal on Scientific Computing* 31.1 (2008), pp. 761–775.

[65] József Szabados and Vertesi Peter. *Interpolation of functions*. World Scientific, 1990.

[66] Lloyd N Trefethen. *Approximation theory and approximation practice*. Vol. 128. Siam, 2013.

[67] Lloyd N Trefethen, Ásgeir Birkisson, and Tobin A Driscoll. *Exploring ODEs*. Vol. 157. SIAM, 2017.

[68] Lloyd N Trefethen and JAC Weideman. “Two results on polynomial interpolation in equally spaced points”. In: *Journal of Approximation Theory* 65.3 (1991), pp. 247–260.

[69] Charles F Van Loan. “The ubiquitous Kronecker product”. In: *Journal of computational and applied mathematics* 123.1-2 (2000), pp. 85–100.

[70] J.M. Varah. “A lower bound for the smallest singular value of a matrix”. In: *Linear Algebra and its Applications* 11.1 (1975), pp. 3–5. ISSN: 0024-3795. DOI: `https://doi.org/10.1016/0024-3795(75)90112-3`. URL: `http://www.sciencedirect.com/science/article/pii/0024379575901123`.

[71] P Vértesi. “Optimal Lebesgue constant for Lagrange interpolation”. In: *SIAM Journal on Numerical Analysis* 27.5 (1990), pp. 1322–1331.

[72] James Vogel et al. “Superfast divide-and-conquer method and perturbation analysis for structured eigenvalue solutions”. In: *SIAM Journal on Scientific Computing* 38.3 (2016), A1358–A1382.

[73] Yuanzhe Xi, Jianlin Xia, and Raymond Chan. “A fast randomized eigensolver with structured LDL factorization update”. In: *SIAM Journal on Matrix Analysis and Applications* 35.3 (2014), pp. 974–996.

[74] Kuan Xu and Nicholas Hale. “Explicit construction of rectangular differentiation matrices”. In: *IMA Journal of Numerical Analysis* 36.2 (2015), pp. 618–632.

[75] Wenjian Yu, Yu Gu, and Yaohang Li. “Efficient Randomized Algorithms for the Fixed-Precision Low-Rank Matrix Approximation”. In: *SIAM Journal on Matrix Analysis and Applications* 39.3 (2018), pp. 1339–1359.

[76] Antoni Zygmund. *Trigonometric series*. Vol. 1. Cambridge university press, 2002.