Computing with functions in two dimensions

Alex Townsend

St John's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity Term 2014

Abstract

New numerical methods are proposed for computing with smooth scalar and vector valued functions of two variables defined on rectangular domains. Functions are approximated to essentially machine precision by an iterative variant of Gaussian elimination that constructs near-optimal low rank approximations. Operations such as integration, differentiation, and function evaluation are particularly efficient. Explicit convergence rates are shown for the singular values of differentiable and separately analytic functions, and examples are given to demonstrate some paradoxical features of low rank approximation theory.

Analogues of QR, LU, and Cholesky factorizations are introduced for matrices that are continuous in one or both directions, deriving a continuous linear algebra. New notions of triangular structures are proposed and the convergence of the infinite series associated with these factorizations is proved under certain smoothness assumptions.

A robust numerical bivariate rootfinder is developed for computing the common zeros of two smooth functions via a resultant method. Using several specialized techniques the algorithm can accurately find the simple common zeros of two functions with polynomial approximants of high degree ($\geq$ 1,000).

Lastly, low rank ideas are extended to linear partial differential equations (PDEs) with variable coefficients defined on rectangles. When these ideas are used in conjunction with a new one-dimensional spectral method the resulting solver is spectrally accurate and efficient, requiring $\mathcal{O}(n^2)$ operations for rank 1 partial differential operators, $\mathcal{O}(n^3)$ for rank 2, and $\mathcal{O}(n^4)$ for rank ≥ 3 to compute an $n \times n$ matrix of bivariate Chebyshev expansion coefficients for the PDE solution.

The algorithms in this thesis are realized in a software package called Chebfun2, which is an integrated two-dimensional component of Chebfun.

Acknowledgements

My graduate experience could have been very different if it was not for a few key individuals. I wish to thank them now.

I started my academic life with a different supervisor and research topic. One year into my graduate studies my plans needed to change and in a remarkable sequence of events Professor Trefethen adopted me as his DPhil student just 29 hours before attending his own wedding! This exemplifies the dedication that Professor Trefethen has to his students and to academia. He is a fantastic mentor for young mathematicians, and I have and will continue to benefit from his advice. Professor Trefethen's suggestion to write research memoranda is the piece of advice that has had the biggest impact on my research life. Trefethen has also read many drafts of this thesis and his suggestions have been greatly appreciated.

I have shared, discussed, and exchanged many ideas with my collaborators: Nick Hale, Sheehan Olver, Yuji Nakatsukasa, and Vanni Noferini. Though many of these ideas do not appear in this thesis, the hours of discussion and email exchanges have shaped me as a researcher. It has been a pleasure working with and learning from these future stars.

My fiancé has supported me throughout and has listened to many of my practice presentations. As an expert in developmental biology her mathematical intuition is astonishing. There are few like her. I also wish to thank my parents for kindling my early love of mathematics. Before my math questions became too difficult my Dad would spend Sunday mornings answering them. I hope they consider this thesis a small return on their personal and financial investment.

Finally, I would like to thank Anthony Austin and Hrothgar for reading through the thesis and making many useful suggestions.

Contents

Chapter 1

Introduction

This thesis develops a powerful collection of numerical algorithms for computing with scalar and vector valued functions of two variables defined on rectangles, as well as an armory of theoretical tools for understanding them. Every chapter (excluding this one) contains new contributions to numerical analysis and scientific computing.

Chapter 2 develops the observation that established one-dimensional (1D) algorithms can be exploited in two-dimensional (2D) computations with functions, and while this observation is not strictly new it has not been realized with such generality and practicality. At the heart of this is an algorithm based on an iterative variant of Gaussian elimination on functions for constructing low rank function approximations. This algorithm forms the core of a software package for computing with 2D functions called Chebfun2.

Chapter 3 investigates the theory underlying low rank function approximation. We relate the smoothness of a function to the decay rate of its singular values. We go on to define a set of functions that are particularly amenable to low rank approximation and give three examples that reveal the subtle nature of this set. The first example is important in its own right as we prove that a continuous analogue of the Hilbert matrix is numerically of low rank. Finally, we investigate the properties of the singular value decomposition for functions and give three characterizations of the singular values.

Chapter 4 extends standard concepts in numerical linear algebra related to matrix factorizations to functions. In particular, we address SVD, QR, LU, and Cholesky factorizations in turn and ask for each: "What is the analogue for functions?" In the process of answering such questions we define what "triangular" means in this context and determine which algorithms are meaningful for functions. New questions arise

related to compactness and convergence of infinite series that we duly address. These factorizations are useful for revealing certain properties of functions; for instance, the Cholesky factorization of a function can be used as a numerical test for nonnegative definiteness.

Chapter 5 develops a numerical algorithm for the challenging 2D rootfinding problem. We describe a robust algorithm based on a resultant method with Bézout resultant matrices, employing various techniques such as subdivision, local refinement, and regularization. This algorithm is a 2D analogue of a 1D rootfinder based on the eigenvalues of a colleague matrix. We believe this chapter develops one of the most powerful global bivariate rootfinders currently available.

In Chapter 6, we solve partial differential equations (PDEs) defined on rectangles by extending the concept of low rank approximation to partial differential operators. This allows us to automatically solve many PDEs via the solution of a generalized Sylvester matrix equation. Taking advantage of a sparse and well-conditioned 1D spectral method, we develop a PDE solver that is much more accurate and has a lower complexity than standard spectral methods.

Chebyshev polynomials take center stage in our work, making all our algorithms and theorems ready for practical application. In one dimension there is a vast collection of practical algorithms and theorems based on Chebyshev polynomials, and in this thesis we extend many of them to two dimensions.

1.1 Chebyshev polynomials

Chebyshev polynomials are an important family of orthogonal polynomials in numerical analysis and scientific computing [52, 104, 128, 159]. For $j \geq 0$ the Chebyshev polynomial of degree j is denoted by $T_j(x)$ and is given by:

$$T_j(x) = \cos(j \cos^{-1} x), \qquad x \in [-1, 1].$$

Chebyshev polynomials are orthogonal with respect to a weighted inner product, i.e.,

$$\int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}}dx = \begin{cases} \pi, & i = j = 0, \\ \pi/2, & i = j \geq 1, \\ 0, & i \neq j, \end{cases}$$

and as a consequence [122, Sec. 11.4] satisfy a 3-term recurrence relation [114, 18.9(i)],

$$T_{j+1}(x) = 2xT_j(x) - T_{j-1}(x), \quad j \geq 1,$$

with $T_1(x) = x$ and $T_0(x) = 1$. Chebyshev polynomials are a practical basis for representing polynomials on intervals and lead to a collection of numerical algorithms for computing with functions of one real variable (see Section 1.6).

1.2 Chebyshev interpolants and projections

Let n be a positive integer and let $\left\{x_j^{\text{cheb}}\right\}_{0\leq j\leq n}$ be the set of $n+1$ Chebyshev points, which are defined as

$$x_j^{\text{cheb}} = \cos\left(\frac{(n-j)\pi}{n}\right), \qquad 0 \leq j \leq n. \tag{1.1}$$

Given a set of data $\{f_j\}_{0\leq j\leq n}$, there is a unique polynomial p_n^{interp} of degree at most n such that $p_n^{\text{interp}}(x_j^{\text{cheb}}) = f_j$ for $0 \leq j \leq n$. The polynomial p_n^{interp} is referred to as the *Chebyshev interpolant* of f of degree n. If $\{f_j\}_{0\leq j\leq n}$ are data from a continuous function $f : [-1, 1] \to \mathbb{C}$ with $f(x_j^{\text{cheb}}) = f_j$ for $0 \leq j \leq n$, then [159, Thm. 15.1 and 15.2]

$$\left\|f - p_n^{\text{interp}}\right\|_\infty \leq \left(2 + \frac{2}{\pi}\log(n+1)\right)\left\|f - p_n^{\text{best}}\right\|_\infty, \tag{1.2}$$

where p_n^{best} is the best minimax polynomial approximant to f of degree at most n, and $\|\cdot\|_\infty$ is the $L^\infty([-1, 1])$ norm. Chebyshev interpolation is quasi-optimal for polynomial approximation since $\left\|f - p_n^{\text{interp}}\right\|_\infty$ is at most $\mathcal{O}(\log n)$ times larger than $\left\|f - p_n^{\text{best}}\right\|_\infty$. In fact, asymptotically, one cannot do better with polynomial

interpolation since for any set of $n+1$ points on $[-1,1]$ there is a continuous function f such that $\left\|f - p_n^{\text{interp}}\right\|_\infty \geq (1.52125 + \frac{2}{\pi}\log(n+1))\|f - p_n^{\text{best}}\|_\infty$ [29].

The polynomial interpolant, p_n^{interp}, can be represented as a Chebyshev series,

$$p_n^{\text{interp}}(x) = \sum_{j=0}^{n} c_j T_j(x). \tag{1.3}$$

The coefficients $c_0, \ldots, c_n$ can be numerically computed from the data $f_0, \ldots, f_n$ in $\mathcal{O}(n \log n)$ operations by the discrete Chebyshev transform, which is equivalent to the DCT-I (type-I discrete cosine transform) [57].

Another polynomial approximant to f is the *Chebyshev projection*, p_n^{proj}, which is defined by truncating the Chebyshev series of f after $n+1$ terms. If f is Lipschitz continuous, then it has an absolutely and uniformly convergent Chebyshev series [159, Thm. 3.1] given by $f(x) = \sum_{j=0}^{\infty} a_j T_j(x)$, where the coefficients $\{a_j\}_{j\geq 0}$ are defined by the integrals

$$a_0 = \frac{1}{\pi}\int_{-1}^{1} \frac{f(x)T_0(x)}{\sqrt{1-x^2}}dx, \qquad a_j = \frac{2}{\pi}\int_{-1}^{1} \frac{f(x)T_j(x)}{\sqrt{1-x^2}}dx, \quad j \geq 1. \tag{1.4}$$

The Chebyshev expansion for f can be truncated to construct p_n^{proj} as follows:

$$p_n^{\text{proj}}(x) = \sum_{j=0}^{n} a_j T_j(x). \tag{1.5}$$

The approximation errors $\|f - p_n^{\text{proj}}\|_\infty$ and $\|f - p_n^{\text{interp}}\|_\infty$ usually decay at the same asymptotic rate since the coefficients $\{c_j\}_{j\geq 0}$ in (1.3) are related to $\{a_j\}_{j\geq 0}$ in (1.5) by an aliasing formula [159, Thm. 4.2].

As a general rule, Chebyshev projections tend to be more convenient for theoretical work, while Chebyshev interpolants are often faster to compute in practice.

1.3 Convergence results for Chebyshev approximation

The main convergence results for Chebyshev approximation can be summarized by the following two statements:

1. If f is ν times continuously differentiable (and the ν derivative is of bounded total variation), then $\|f - p_n\|_\infty = \mathcal{O}(n^{-\nu})$.

2. If f is analytic on $[-1,1]$ then $\|f - p_n\|_\infty = \mathcal{O}\left(\rho^{-n}\right)$, for some $\rho > 1$.

Here, p_n can be the Chebyshev interpolant of f or its Chebyshev projection.

For the first statement we introduce the concept of *bounded total variation*. A function $f : [-1,1] \to \mathbb{C}$ is of bounded total variation $V_f < \infty$ if

$$V_f = \int_{-1}^{1} |f'(x)| dx < \infty,$$

where the integral is defined in a distributional sense if necessary (for more details see [24, Sec. 5.2] and [159, Chap. 7]).

Theorem 1.1 (Convergence for differentiable functions)**.** *For an integer $\nu \geq 1$, let f and its derivatives through $f^{(\nu-1)}$ be absolutely continuous on $[-1,1]$ and suppose the νth derivative $f^{(\nu)}$ is of bounded total variation V_f. Then, for $n > \nu$,*

$$\left\|f - p_n^{\text{interp}}\right\|_\infty \leq \frac{4V_f}{\pi\nu(n-\nu)^\nu}, \qquad \left\|f - p_n^{\text{proj}}\right\|_\infty \leq \frac{2V_f}{\pi\nu(n-\nu)^\nu}.$$

Proof. See [159, Theorem 7.2]. □

For the second statement we introduce the concept of a *Bernstein ellipse*, as is common usage in approximation theory [159]. (We abuse terminology slightly by using the word "ellipse" to denote a certain region bounded by an ellipse.) The Bernstein ellipse E_ρ with $\rho > 1$ is the open region in the complex plane bounded by the ellipse with foci ± 1 and semiminor and semimajor axis lengths summing to ρ. Figure 1.1 shows E_ρ for $\rho = 1.2, 1.4, 1.6, 1.8, 2.0$.

Theorem 1.2 (Convergence for analytic functions)**.** *Let a function f be analytic on $[-1,1]$ and analytically continuable to the open Bernstein ellipse E_ρ, where it satisfies $|f| \leq M < \infty$. Then, for $n \geq 0$,*

$$\left\|f - p_n^{\text{interp}}\right\|_\infty \leq \frac{4M\rho^{-n}}{\rho - 1}, \qquad \left\|f - p_n^{\text{proj}}\right\|_\infty \leq \frac{2M\rho^{-n}}{\rho - 1}.$$

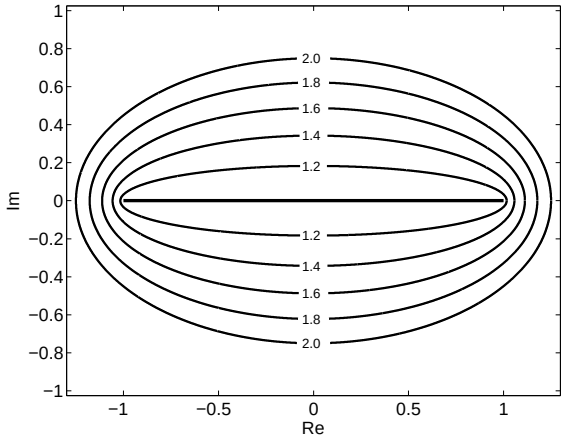


Figure 1.1: Bernstein ellipses E_ρ in the complex plane for $\rho = 1.2, 1.4, 1.6, 1.8, 2.0$. If $f : [-1,1] \to \mathbb{C}$ is analytic on $[-1,1]$ and analytically continuable to a bounded function in E_ρ, then $\|f - p_n^{\text{interp}}\|_\infty = \mathcal{O}\left(\rho^{-n}\right)$ and $\|f - p_n^{\text{proj}}\|_\infty = \mathcal{O}\left(\rho^{-n}\right)$.

Proof. See [159, Theorem 8.2]. □

Theorem 1.1 and Theorem 1.2 suggest that p_n^{interp} and p_n^{proj} have essentially the same asymptotic approximation power. Analogous convergence results hold for Laurent and Fourier series (cf. [162, (2.18)]).

1.4 Why Chebyshev polynomials?

Chebyshev polynomials are intimately connected with the Fourier and Laurent bases. If $x \in [-1,1]$, $\theta = \cos^{-1} x$, and $z = e^{\mathrm{i}\theta}$, then we have $T_j(x) = \text{Re}\left(e^{\mathrm{i}j\theta}\right) = (z^j + z^{-j})/2$ for $j \geq 0$. Therefore, we have the following relations:

$$\underbrace{\sum_{j=0}^{\infty} \alpha_j T_j(x)}_{\text{Chebyshev}} = \underbrace{\sum_{j=0}^{\infty} \alpha_j \text{Re}\left(e^{\mathrm{i}j\theta}\right)}_{\text{even Fourier}} = \underbrace{\alpha_0 + \sum_{j=-\infty, j\neq 0}^{\infty} \frac{\alpha_{|j|}}{2} z^j}_{\text{palindromic Laurent}}, \tag{1.6}$$

where the series are assumed to converge uniformly and absolutely.

As summarized in Table 1.1, a Chebyshev expansion of a function on $[-1,1]$ can be viewed as a Fourier series of an even function on $[-\pi,\pi]$. This in turn can be seen as a palindromic Laurent expansion on the unit circle of a function satisfying $f(z) = f(\overline{z})$, where $\overline{z}$ denotes the complex conjugate of z. Just as the Fourier basis is a natural one for representing periodic functions, Chebyshev polynomials are a natural basis for functions on $[-1,1]$.

Series	Assumptions	Setting	Interpolation points
Chebyshev	none	$x \in [-1, 1]$	Chebyshev
Fourier	$f(\theta) = f(-\theta)$	$\theta \in [-\pi, \pi]$	equispaced
Laurent	$f(z) = f(\overline{z})$	$z \in$ unit circle	roots of unity

Table 1.1: Fourier, Chebyshev, and Laurent series are closely related. Each representation can be converted to the other by a change of variables, and under the same transformation, Chebyshev points, equispaced points, and roots of unity are connected.

One reason the connections in (1.6) are important is they allow the discrete Chebyshev transform, which converts $n+1$ values at Chebyshev points to the Chebyshev coefficients of p_n^{interp} in (1.3), to be computed via the fast Fourier transform (FFT) [57].

1.5 Ultraspherical polynomials

The set of Chebyshev polynomials $\{T_j\}_{j\geq 0}$ are a limiting case of the set of ultraspherical (or Gegenbauer) polynomials [146, Chap. IV]. For a fixed $\lambda > 0$ the ultraspherical polynomials, denoted by $C_j^{(\lambda)}$ for $j \geq 0$, are orthogonal on $[-1, 1]$ with respect to the weight function $(1-x^2)^{\lambda-1/2}$. They satisfy $T_j(x) = \lim_{\lambda\to 0^+} \frac{j}{2\lambda} C_j^{(\lambda)}(x)$ and the following 3-term recurrence relation [114, (18.9.1)]:

$$C_{j+1}^{(\lambda)}(x) = \frac{2(j+\lambda)}{j+1} x C_j^{(\lambda)}(x) - \frac{j+2\lambda-1}{j+1} C_{j-1}^{(\lambda)}(x), \qquad j \geq 1, \tag{1.7}$$

where $C_1^{(\lambda)} = 2\lambda x$ and $C_0^{(\lambda)} = 1$.

Ultraspherical polynomials are of interest because of the following relations for $n \geq 0$ [114, (18.9.19) and (18.9.21)]:

$$\frac{d^k T_n}{dx^k} = \begin{cases} 2^{k-1} n (k-1)!\, C_{n-k}^{(k)}, & n \geq k, \\ 0, & 0 \leq n \leq k-1, \end{cases} \tag{1.8}$$

which means that first, second, and higher order spectral Chebyshev differentiation matrices can be represented by sparse operators. Using (1.8), together with other simple relations, one can construct an efficient spectral method for partial differential equations (see Chapter 6).

Chebfun command	Operation	Algorithm
`feval`	evaluation	Clenshaw's algorithm [37]
`chebpoly`	coefficients	DCT-I transform [57]
`sum`	integration	Clenshaw–Curtis quadrature [38, 164]
`diff`	differentiation	recurrence relation [104, p. 34]
`roots`	rootfinding	eigenvalues of colleague matrix [27, 63]
`max`	maximization	roots of the derivative
`qr`	QR factorization	Householder triangularization [157]

Table 1.2: A selection of Chebfun commands, their corresponding operations, and underlying algorithms. In addition to the references cited, these algorithms are discussed in [159].

1.6 Chebfun

Chebfun is a software system written in object-oriented MATLAB that is based on Chebyshev interpolation and related technology for computing with continuous and piecewise continuous functions of one real variable.

Chebfun approximates a globally smooth function $f : [-1, 1] \to \mathbb{C}$ by an interpolant of degree n, where n is selected adaptively. The function f is sampled on progressively finer Chebyshev grids of size $9, 17, 33, 65, \ldots$, and so on, until the high degree Chebyshev coefficients of p_n^{interp} in (1.3) fall below machine precision relative to $\|f\|_\infty$ [9]. The numerically insignificant trailing coefficients are truncated to leave a polynomial that accurately approximates f. For example, Chebfun approximates e^x on $[-1, 1]$ by a polynomial of degree 14 and during the construction process an interpolant at 17 Chebyshev points is formed before 2 negligible trailing coefficients are discarded.

Once a function has been represented by Chebfun the resulting object is called a *chebfun*, in lower case letters. About 200 operations can be performed on a chebfun `f`, such as `f(x)` (evaluates `f` at a point), `sum(f)` (integrates `f` over its domain), and `roots(f)` (computes the roots of `f`). Many of the commands in MATLAB for vectors have been overloaded, in the object-oriented sense of the term, for chebfuns with a new continuous meaning [9]. (By default a chebfun is the continuous analogue of a column vector.) For instance, if `v` is a vector in MATLAB and `f` is a chebfun, then `max(v)` returns the maximum entry of `v`, whereas `max(f)` returns the global maximum of `f` on its domain. Table 1.2 summarizes a small selection of Chebfun commands and their underlying algorithms.

Chebfun aims to compute operations to within approximately unit roundoff multiplied by the condition number of the problem, though this is not guaranteed for all operations. For example, the condition number of evaluating a differentiable function f at x is $xf'(x)/f(x)$ and ideally, one could find an integer, n, such that the polynomial interpolant p_n^{interp} on $[-1,1]$ satisfies

$$\left\| f - p_n^{\text{interp}} \right\|_\infty \le \left(2 + \frac{2}{\pi} \log(n+1) \right) \frac{\|f'\|_\infty}{\|f\|_\infty} u,$$

where u is unit machine roundoff and the extra $\mathcal{O}(\log n)$ factor comes from interpolation (see (1.2)). In practice, Chebfun overestimates the condition number and in this case $2 + \frac{2}{\pi}\log(n+1)$ is replaced by $n^{2/3}$, which relaxes the error bound slightly to ensure the adaptive procedure for selecting n is more robust.

Chebfun follows a floating-point paradigm where the result of every arithmetic operation is rounded to a *nearby* function [156]. That is, after each operation negligible trailing coefficients in a Chebyshev expansion are removed, leading to a chebfun of approximately minimal degree that represents the result to machine precision. For example, the final chebfun obtained from pointwise multiplication is often of much lower degree than one would expect in exact arithmetic. Chebfun is a collection of stable algorithms designed so that the errors incurred by rounding in this way do not accumulate. In addition, rounding prevents the combinatorial explosion of complexity that can sometimes be seen in symbolic computations [120].

Aside from mathematics, Chebfun is about people and my good friends. At any one time, there are about ten researchers that work on the project and make up the so-called *Chebfun team*. Over the years there have been significant contributions from many individuals and Table 1.3 gives a selection.

1.7 Quasimatrices

An $[a,b] \times n$ *column quasimatrix* $\mathcal{A}$ is a matrix with n columns, where each column is a function of one variable defined on an interval $[a,b] \subset \mathbb{R}$ [9, 42, 140]. Quasimatrices can be seen as a continuous analogue of tall-skinny matrices, where the rows are indexed by a continuous, rather than discrete, variable. Motivated by this, we use the following notation for column and row indexing quasimatrices:

$$\mathcal{A}(y_0,:) = [\, c_1(y_0) \,|\, \cdots \,|\, c_n(y_0) \,] \in \mathbb{C}^{1\times n}, \qquad \mathcal{A}(:,j) = c_j, \quad 1 \le j \le n$$

Member	Significant contribution	References
Zachary Battles	Original developer	[9]
Ásgeir Birkisson	Nonlinear ODEs	[23]
Toby Driscoll	Main Professor (2006–present)	[46]
Pedro Gonnet	Padé approximation	[62]
Stefan Güttel	Padé approximation	[62]
Nick Hale	Project director (2010–2013)	[74, 75]
Mohsin Javed	Delta functions	[88]
Georges Klein	Floater–Hormann interpolation	[92]
Ricardo Pachón	Best approximations	[118]
Rodrigo Platte	Piecewise functions	[117]
Mark Richardson	Functions with singularities	[126]
Nick Trefethen	Lead Professor (2002–present)	[159]
Joris Van Deun	CF approximation	[163]
Kuan Xu	Chebyshev interpolants at 1st kind points	–

Table 1.3: A selection of past and present members of the Chebfun team that have made significant contributions to the project. This table does not include contributions to the musical entertainment at parties.

where $y_0 \in [a, b]$ and $c_1, \ldots, c_n$ are the columns of $\mathcal{A}$ (functions defined on $[a, b]$).

An $n \times [a, b]$ *row quasimatrix* has n rows, where each row is a function defined on $[a, b] \subset \mathbb{R}$. The transpose, denoted by $\mathcal{A}^T$, and conjugate transpose, denoted by $\mathcal{A}^*$, of a row quasimatrix is a column quasimatrix. The term *quasimatrix* is used to refer to a column or row quasimatrix and for convenience we do not indicate the orientation if it is clear from the context.

Quasimatrices can be constructed in Chebfun by horizontally concatenating chebfuns together. From here, one can extend standard matrix notions such as condition number, null space, QR factorization, and singular value decomposition (SVD) to quasimatrices [9, 157]. If `A` is a quasimatrix in Chebfun, then `cond(A)`, `null(A)`, `qr(A)`, and `svd(A)` are continuous analogues of the corresponding MATLAB commands for matrices. For instance, Figure 1.2 outlines the algorithm for computing the SVD of a quasimatrix. Further details are given in Chapter 4.

Mathematically, a quasimatrix can have infinitely many columns ($n = \infty$) and these are used in Chapter 4 to describe the SVD, QR, LU, and Cholesky factorizations of bivariate functions. An $\infty \times [a, b]$ quasimatrix is infinite in both columns and rows, but not square as it has uncountably many rows and only countably many columns.

Algorithm: Singular value decomposition of a quasimatrix

Input: An $[a,b] \times n$ quasimatrix, $\mathcal{A}$.

Output: $\mathcal{A} = \mathcal{U}\Sigma V^*$, where $\mathcal{U}$ is an $[a,b] \times n$ quasimatrix with orthonormal columns, Σ is an $n \times n$ diagonal matrix, and V is a unitary matrix.

1. Compute $\mathcal{A} = \mathcal{Q}R$ (QR of a quasimatrix)
2. Compute $R = U\Sigma V^*$ (SVD of a matrix)
3. Construct $\mathcal{U} = \mathcal{Q}U$ (quasimatrix-matrix product)

Figure 1.2: Pseudocode for computing the SVD of a quasimatrix [9]. The QR factorization of a quasimatrix is described in [157].

1.8 Low rank function approximation

If $A \in \mathbb{C}^{m\times n}$ is a matrix, then the first k singular values and vectors of the SVD can be used to construct a best rank k approximation to A in any unitarily invariant matrix norm [109]. For instance, by the Eckart–Young[1] Theorem [47], if $A = U\Sigma V^*$, where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix with diagonal entries $\sigma_1 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$, then for $k \geq 1$ we have

$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^*,$$

$$\inf_{B_k} \|A - B_k\|_2^2 = \|A - A_k\|_2^2 = \sigma_{k+1}(A)^2,$$

$$\inf_{B_k} \|A - B_k\|_F^2 = \|A - A_k\|_F^2 = \sum_{j=k+1}^{\min(m,n)} \sigma_j(A)^2,$$

where u_j and v_j are the jth columns of U and V, respectively, the infima are taken over $m \times n$ matrices of rank at most k, $\|\cdot\|_2$ is the matrix 2-norm, and $\|\cdot\|_F$ is the matrix Frobenius norm.

Analogously, an L^2-integrable function of two variables $f : [a,b] \times [c,d] \to \mathbb{C}$ can be approximated by low rank functions. A nonzero function is called a *rank* 1 *function* if it is the product of a function in x and a function in y, i.e., $f(x,y) = g(y)h(x)$. (Rank 1 functions are also called *separable functions* [11].) A function is of rank at most k if it is the sum of k rank 1 functions. Mathematically, most functions, such

[1] In fact, it was Schmidt who first proved the Eckart–Young Theorem about 29 years before Eckart and Young did [130, 139].

as $\cos(xy)$, are of infinite rank but numerically they can often be approximated on a bounded domain to machine precision by functions of small finite rank. For example, $\cos(xy)$ can be globally approximated to 16 digits on $[-1,1]^2$ by a function of rank 6. This observation lies at the heart of Chapter 2.

If $f : [a,b] \times [c,d] \to \mathbb{C}$ is an L^2-integrable function, then it has an SVD that can be expressed as an infinite series

$$f(x,y) = \sum_{j=1}^{\infty} \sigma_j u_j(y) \overline{v_j(x)}, \tag{1.9}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$ and $\{u_j\}_{j\geq 1}$ and $\{v_j\}_{j\geq 1}$ are orthonormal sets of functions in the L^2 inner product [130]. In (1.9) the series converges to f in the L^2-norm and the equality sign should be understood to signify convergence in that sense [130]. Extra smoothness assumptions on f are required to guarantee that the series converges absolutely and uniformly to f (see Theorem 3.3).

A best rank k approximant to f in the L^2-norm can be constructed by truncating the infinite series in (1.9) after k terms,

$$\begin{aligned} f_k(x,y) &= \sum_{j=1}^{k} \sigma_j u_j(y) \overline{v_j(x)}, \\ \inf_{g_k} \|f - g_k\|_{L^2}^2 &= \|f - f_k\|_{L^2}^2 = \sum_{j=k+1}^{\infty} \sigma_j^2, \end{aligned} \tag{1.10}$$

where the infimum is taken over L^2-integrable functions of rank k [130, 139, 166]. The smoother the function the faster $\|f - f_k\|_{L^2}^2$ decays to zero as $k \to \infty$, and as we show in Theorem 3.1 and Theorem 3.2, differentiable and analytic functions have algebraically and geometrically accurate low rank approximants, respectively.

In practice the computation of the SVD of a function is expensive and instead in Chapter 2 we use an iterative variant of Gaussian elimination with complete pivoting to construct near-optimal low rank approximants. In Chapter 4 these ideas are extended further to continuous analogues of matrix factorizations.

1.9 Approximation theory for bivariate functions

In one variable a continuous real-valued function defined on a interval has a best minimax polynomial approximation [122, Thm. 1.2] that is unique [122, Thm. 7.6] and satisfies an equioscillation property [122, Thm. 7.2]. In two variables a continuous function $f : [a, b] \times [c, d] \to \mathbb{R}$ also has a best minimax polynomial approximation, but it is not guaranteed to be unique.

Theorem 1.3 (Minimax approximation in two variables)**.** *Let $f : [a, b] \times [c, d] \to \mathbb{R}$ be a continuous function and let m and n be integers. Then, there exists a bivariate polynomial $p_{m,n}^{\text{best}}$ of degree at most m in x and at most n in y such that*

$$\left\| f - p_{m,n}^{\text{best}} \right\|_\infty = \inf_{p \in \mathcal{P}_{m,n}} \|f - p\|_\infty ,$$

where $\mathcal{P}_{m,n}$ denotes the set of bivariate polynomials of degree at most m in x and at most n in y. In contrast to the situation in one variable, $p_{m,n}^{\text{best}}$ need not be unique.

Proof. Existence follows from a continuity and compactness argument [122, Thm. 1.2]. Nonuniqueness is a direct consequence of Haar's Theorem [71, 100]. Further discussion is given in [125, 127]. □

The definition of p_n^{proj} can also be extended for bivariate functions that are sufficiently smooth.

Theorem 1.4 (Uniform convergence of bivariate Chebyshev expansions)**.** *Let $f : [-1, 1]^2 \to \mathbb{C}$ be a continuous function of bounded variation (see* [104, *Def.* 5.2] *for a definition of bounded variation for bivariate functions) with one of its partial derivatives existing and bounded in $[-1, 1]^2$. Then f has a bivariate Chebyshev expansion,*

$$f(x, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{ij} T_i(y) T_j(x), \tag{1.11}$$

where the series converges uniformly to f.

Proof. See [104, Thm. 5.9]. □

This means that bivariate Chebyshev projections $p_{m,n}^{\text{proj}}$ of degree m in x and degree n in y can be defined for functions satisfying the assumptions of Theorem 1.4 by

truncating the series in (1.11), i.e.,

$$p_{m,n}^{\text{proj}}(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} a_{ij}T_i(y)T_j(x). \tag{1.12}$$

Once a Chebyshev projection of a bivariate function has been computed, one can construct a low rank approximation by taking the SVD of the $(n+1)\times(m+1)$ matrix of coefficients in (1.12). However, the main challenge is to construct a low rank approximation to a bivariate function in a computational cost that depends only linearly on $\max(m,n)$. In the next chapter we will present an algorithm based on Gaussian elimination that achieves this.

Chapter 2

An extension of Chebyshev technology to two dimensions*

Chebyshev technology is a well-established tool for computing with functions of one real variable, and in this chapter we describe how these ideas can be extended to two dimensions for scalar and vector valued functions. These types of functions represent scalar functions, vector fields, and surfaces embedded in $\mathbb{R}^3$, and we focus on the situation in which the underlying function is globally smooth and aim for spectrally accurate algorithms. The diverse set of 2D operations that we would like to perform range from integration to vector calculus operations to global optimization.

A compelling paradigm for computing with continuous functions on bounded domains, and the one that we use in this thesis, is to replace them by polynomial approximants (or "proxies") of sufficiently high degree so that the approximation is accurate to machine precision relative to the absolute maximum of the function [28, 161]. Often, operations on functions can then be efficiently calculated, up to an error of machine precision, by numerically computing with a polynomial approximant. Here, we use low rank approximants, i.e., sums of functions of the form $g(y)h(x)$, where the functions of one variable are polynomials that are represented by Chebyshev expansions (see Section 1.2).

Our low rank approximations are efficiently constructed by an iterative variant of Gaussian elimination (GE) with complete pivoting on functions [151]. This involves applying GE to functions rather than matrices and adaptively terminating the number of pivoting steps. To decide if GE has resolved a function we use a mix of 1D and

*This chapter is based on a paper with Nick Trefethen [152]. Trefethen proposed that we work with Geddes–Newton approximations, which were quickly realized to be related to low rank approximations. I developed the algorithms, the Chebfun2 codes, and was the lead author in writing the paper [152].

2D resolution tests (see Section 2.1). An approximant that has been formed by this algorithm is called a *chebfun2*, which is a 2D analogue of a chebfun (see Section 1.6).

Our extension of Chebyshev technology to two dimensions is realized in a software package called Chebfun2 that is publicly available as part of Chebfun [152, 161]. Once a function has been approximated by Chebfun2 there are about 150 possible operations that can be performed with it; a selection is described in this thesis.

Throughout this chapter we restrict our attention to scalar and vector valued functions defined on the unit square, i.e. $[-1,1]^2$, unless stated otherwise. The algorithms and software permit easy treatment of general rectangular domains.

2.1 Gaussian elimination for functions

We start by describing how a chebfun2 approximant is constructed.

Given a continuous function $f : [-1,1]^2 \to \mathbb{C}$, an optimal rank k approximant in the L^2-norm is given by the first k terms of the SVD (see Section 1.8). Alternatively, an algorithm mathematically equivalent to k steps of GE with complete pivoting can be used to construct a near-optimal rank k approximant. In this section we describe a continuous idealization of GE and later detail how it can be made practical (see Section 2.1.1).

First, we define $e_0 = f$ and find $(x_1, y_1) \in [-1,1]^2$ such that[1] $|e_0(x_1, y_1)| = \max(|e_0(x, y)|)$. Then, we construct the rank 1 function

$$f_1(x,y) = \frac{e_0(x_1,y)e_0(x,y_1)}{e_0(x_1,y_1)} = d_1 c_1(y) r_1(x),$$

where $d_1 = 1/e_0(x_1, y_1)$, $c_1(y) = e_0(x_1, y)$, and $r_1(x) = e_0(x, y_1)$, which coincides with f along the two lines $y = y_1$ and $x = x_1$. We calculate the residual $e_1 = f - f_1$ and repeat the same procedure to form the rank 2 function

$$f_2(x,y) = f_1(x,y) + \frac{e_1(x_2,y)e_1(x,y_2)}{e_1(x_2,y_2)} = f_1(x,y) + d_2 c_2(y) r_2(x),$$

where $(x_2, y_2) \in [-1,1]^2$ is such that $|e_1(x_2, y_2)| = \max(|e_1(x, y)|)$. The function f_2 coincides with f along the lines $x = x_1$, $x = x_2$, $y = y_1$, and $y = y_2$. We continue constructing successive approximations $f_1, f_2, \ldots, f_k$, where f_k coincides with f along

[1] In practice we estimate the global absolute maximum by taking the discrete absolute maximum from a sample grid (see Section 2.1.1).

Algorithm: GE with complete pivoting on functions

Input: A function $f = f(x, y)$ on $[-1, 1]^2$ and a tolerance *tol*.
Output: A low rank approximation $f_k(x, y)$ such that $\|f - f_k\|_\infty \leq tol$.
$e_0(x, y) = f(x, y)$, $f_0(x, y) = 0$, $k = 1$
while $\|e_k\|_\infty > tol$
    Find (x_k, y_k) s.t. $|e_{k-1}(x_k, y_k)| = \max(|e_{k-1}(x, y)|), \quad (x, y) \in [-1, 1]^2$
    $e_k(x, y) = e_{k-1}(x, y) - e_{k-1}(x_k, y)e_{k-1}(x, y_k)/e_{k-1}(x_k, y_k)$
    $f_k(x, y) = f_{k-1}(x, y) + e_{k-1}(x_k, y)e_{k-1}(x, y_k)/e_{k-1}(x_k, y_k)$
    $k = k + 1$
end

Figure 2.1: Iterative GE with complete pivoting on functions of two variables. The first k steps can be used to construct a rank k approximation to f. In practice, this continuous idealization must be discretized.

$2k$ lines, until $\|e_k\|_\infty = \|f - f_k\|_\infty$ falls below machine precision relative to $\|f\|_\infty$. Figure 2.1 gives the pseudocode for this algorithm, which is a continuous analogue of matrix GE (cf. [160, Alg. 20.1]).

Despite the many similarities between GE for matrices and for functions, there are also notable differences: (1) The pivoting is done implicitly as the individual columns and rows of a function are not physically permuted, whereas for matrices the pivoting is usually described as a permutation; (2) The canonical pivoting strategy we use is complete pivoting, not partial pivoting; and (3) The underlying LU factorization is no longer just a factorization of a matrix. Further discussion related to these differences is given in Chapter 4.

After k steps of GE we have constructed k successive approximations $f_1, \ldots, f_k$. We call $(x_1, y_1), \ldots, (x_k, y_k)$ the *pivot locations* and $d_1, \ldots, d_k$ the *pivot values*. We also refer to the functions $c_1(y), \ldots, c_k(y)$ and $r_1(x), \ldots, r_k(x)$ as the *pivot columns* and *pivot rows*, respectively.

The kth successive approximant, f_k, coincides with f along k pairs of lines that cross at the pivots selected by GE, as we prove in the next theorem. The theorem and proof appears in [11] and is a continuous analogue of [141, (1.12)].

Theorem 2.1 (Cross approximation of GE)**.** *Let $f : [-1, 1]^2 \to \mathbb{C}$ be a continuous function and f_k the rank k approximant of f computed by k steps of GE that pivoted at $(x_1, y_1), \ldots, (x_k, y_k) \in [-1, 1]^2$. Then, f_k coincides with f along the $2k$ lines $x = x_1, \ldots, x = x_k$ and $y = y_1, \ldots, y = y_k$.*

Proof. After the first step the error $e_1(x, y)$ satisfies

$$e_1(x, y) = f(x, y) - f_1(x, y) = f(x, y) - \frac{f(x_1, y)f(x, y_1)}{f(x_1, y_1)}.$$

Substituting $x = x_1$ gives $e_1(x_1, y) = 0$ and substituting $y = y_1$ gives $e_1(x, y_1) = 0$. Hence, f_1 coincides with f along $x = x_1$ and $y = y_1$.

Suppose that f_{k-1} coincides with f along the lines $x = x_j$ and $y = y_j$ for $1 \leq j \leq k - 1$. After k steps we have

$$e_k(x, y) = f(x, y) - f_k(x, y) = e_{k-1}(x, y) - \frac{e_{k-1}(x_k, y)e_{k-1}(x, y_k)}{e_{k-1}(x_k, y_k)}.$$

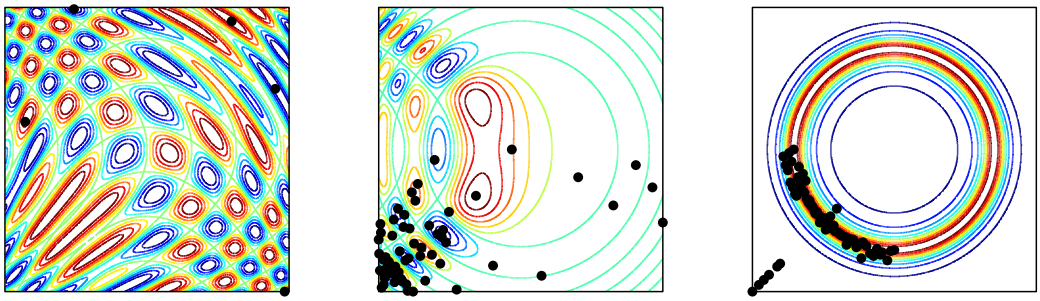
Substituting $x = x_k$ gives $e_k(x_k, y) = 0$ and substituting $y = y_k$ gives $e_k(x, y_k) = 0$. Moreover, since $e_{k-1}(x_j, y) = e_{k-1}(x, y_j) = 0$ for $1 \leq j \leq k - 1$ we have $e_k(x_j, y) = e_{k-1}(x_j, y)$ and $e_k(x, y_j) = e_{k-1}(x, y_j)$ for $1 \leq j \leq k - 1$ and hence, f_k coincides with f along $2k$ lines. The result follows by induction on k. □

Theorem 2.1 implies that GE on functions exactly reproduces functions that are polynomial in the x- or y-variable. For instance, if f is a polynomial of degree m in x, then f_{m+1} coincides with f at $m+1$ points in the x-variable (for every y) and is itself a polynomial of degree $\leq m$ (for every y); hence $f_{m+1} = f$. In particular, a bivariate polynomial of degree m in x and n in y is exactly reproduced after $\min(m, n) + 1$ GE steps.

One may wonder if the sequence of low rank approximants $f_1, f_2, \ldots$ constructed by GE with complete pivoting converges uniformly to f, and in Theorem 4.6 we show that the sequence does converge in that sense under certain assumptions on f.

2.1.1 Algorithmic details

So far we have described GE at a continuous level, but to derive a practical scheme we must work with a discretized version of the algorithm in Figure 2.1. This comes in two stages, with the first designed to find candidate pivot locations and the second to ensure that the pivot columns and rows are resolved.



rank 5 rank 65 rank 33

Figure 2.2: Contour plots for three functions on $[-1,1]^2$ with the pivot locations from stage 1 marked by black dots: (left) $\cos(10(x^2+y)) + \sin(10(x+y^2))$; (center) $\mathrm{Ai}(5(x+y^2))\mathrm{Ai}(-5(x^2+y^2))$; and (right) $1/(1+100(\frac{1}{2}-x^2-y^2)^2)$.

Stage 1: Finding candidate pivot locations

First, we sample f on a 9×9 Chebyshev tensor product grid[2] and perform at most 3 steps of GE. If we find that the sampled matrix can be approximated to machine precision by a rank 1, 2, or 3 matrix, then we move on to stage 2; otherwise, we sample on a 17×17 Chebyshev tensor product grid and perform at most 5 steps of matrix GE with complete pivoting. We proceed to stage 2 if a matrix of rank 5, or less, is sufficient. We continue sampling on nested Chebyshev grids of size 9, 17, 33, 65, and so on, until we discover that the sampled matrix can be approximated to machine precision by a matrix of rank 3, 5, 9, 17, and so on.

Thus, stage 1 approximates a $(2^{j+2}+1) \times (2^{j+2}+1)$ matrix by a matrix of rank at most $2^j + 1$ for $j \geq 1$. Generically, if $f : [-1,1]^2 \to \mathbb{C}$ can be approximated to relative machine precision by a rank k function, then stage 1 samples f on a $(2^{j+2}+1) \times (2^{j+2}+1)$ tensor product grid, where $j = \min(\lceil \log_2(k-1) \rceil, 1)$, and k steps of GE are required. Since

$$2^{\lceil \log_2(k-1) \rceil + 2} + 1 \leq 8k + 1 = \mathcal{O}(k),$$

stage 1 requires $\mathcal{O}(k^3)$ operations. We store the pivot locations used in the k successful steps of GE and go to stage 2. Figure 2.2 shows contour plots of three functions with the pivot locations selected by stage 1.

[2]An $n \times n$ Chebyshev tensor product grid is the set of points $\{(x_i^{\mathrm{cheb}}, x_j^{\mathrm{cheb}})\}_{0 \leq i,j \leq n-1}$, where $\{x_j^{\mathrm{cheb}}\}_{0 \leq j \leq n-1}$ is the set of n Chebyshev points on $[-1,1]$ (see (1.1)).

Stage 2: Resolving the pivot columns and rows

Stage 1 has determined a candidate set of pivot locations required to approximate f, and stage 2 is designed to ensure that the associated pivot columns and rows are sufficiently sampled. For instance, $f(x,y) = x\cos(100y)$ is a rank 1 function, so stage 1 completes after sampling on a 9×9 Chebyshev tensor product grid even though a Chebyshev interpolant of degree 147 is required to resolve the oscillations in the y-direction. For efficiency in this stage we only sample f on a *k-skeleton* of a tensor product grid, i.e., a subset consisting of k columns and rows of the grid, and perform GE on that skeleton. For example, Figure 2.3 shows the 4-skeleton used when approximating Franke's function [53],

$$\begin{aligned} f(x,y) = \frac{3}{4}e^{-\left((9x-2)^2+(9y-2)^2\right)/4} + \frac{3}{4}e^{-\left((9x+1)^2/49-(9y+1)/10\right)} \\ + \frac{1}{2}e^{-\left((9x-7)^2+(9y-3)^2\right)/4} - \frac{1}{5}e^{-\left((9x-4)^2+(9y-7)^2\right)}. \end{aligned} \tag{2.1}$$

After k steps of GE on the skeleton we have sampled the pivot columns and rows at Chebyshev points. Following the procedure used by Chebfun, we convert each pivot column and row to Chebyshev coefficients using the discrete Chebyshev transform to ensure that the coefficients decay to relative machine precision. Figure 2.3 shows the Chebyshev coefficients for the pivot columns used to approximate (2.1). For instance, if the pivot columns are not resolved with 33 points, then we sample f at 65 points along each column and repeat k steps of GE. We continue increasing the sampling along columns and rows until we have resolved them. Since the sets of 9, 17, 33, ..., Chebyshev points are nested, we always pivot at the same locations as determined in stage 1. If the pivot columns require degree $m-1$ Chebyshev interpolants and the pivot rows require degree $n-1$ Chebyshev interpolants, then this stage samples f at, at most,

$$k\left(2^{\lceil \log_2(m) \rceil} + 2^{\lceil \log_2(n) \rceil}\right) \leq 2k(m+n)$$

points. We then perform k steps of GE on the selected rows and columns, requiring $\mathcal{O}\left(k^2(m+n)\right)$ operations.

Once the GE algorithm has terminated we have approximated f by a sum of rank 1 functions,

$$f(x,y) \approx \sum_{j=1}^{k} d_j c_j(y) r_j(x) = \mathcal{C}D\mathcal{R}^T, \tag{2.2}$$

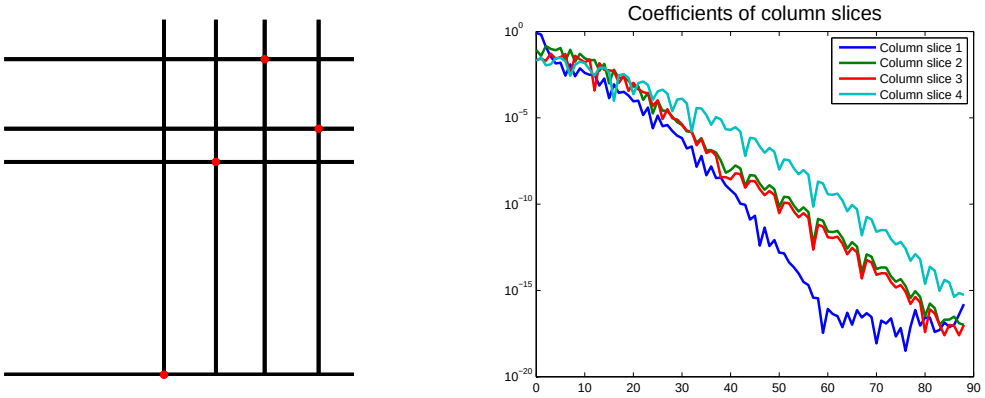


Figure 2.3: Left: The skeleton used in stage 2 of the construction algorithm for approximating Franke's function (2.1). Stage 2 only samples f on the skeleton, i.e., along the black lines. Right: The Chebyshev coefficients of the four pivot columns. The coefficients decay to machine precision, indicating that the pivot columns have been sufficiently sampled.

where $D = \operatorname{diag}(d_1, \ldots, d_k)$, and $\mathcal{C} = [c_1(y) \,|\, \cdots \,|\, c_k(y)]$ and $\mathcal{R} = [r_1(x) \,|\, \cdots \,|\, r_k(x)]$ are $[-1, 1] \times k$ quasimatrices (see Section 1.7). The representation in (2.2) highlights how GE decouples the x and y variables, making it useful for computing some 2D operations using 1D algorithms (see Section 2.2). Given a function f defined on a bounded rectangular domain we call the approximation in (2.2) a *chebfun2 approximant* to f.

Here, we have decided to approximate each column of $\mathcal{C}$ by polynomial approximants of the same degree and likewise, the columns of $\mathcal{R}$ are approximated by polynomials of the same degree. This means that subsequent operations of f can be implemented in a vectorized manner for greater efficiency.

Since in practice we do not use complete pivoting in GE, but instead pick absolute maxima from a sampled grid, one may wonder if this causes convergence or numerical stability issues for our GE algorithm. We observe that it does not and that the algorithm is not sensitive to such small perturbations in the pivot locations; however, we do not currently have a complete stability analysis. We expect that such an analysis is challenging and requires some significantly new ideas.

2.1.2 Near-optimality of Gaussian elimination for functions

We usually observe that GE with complete pivoting computes near-optimal low rank approximants to smooth functions, while being far more efficient than the optimal

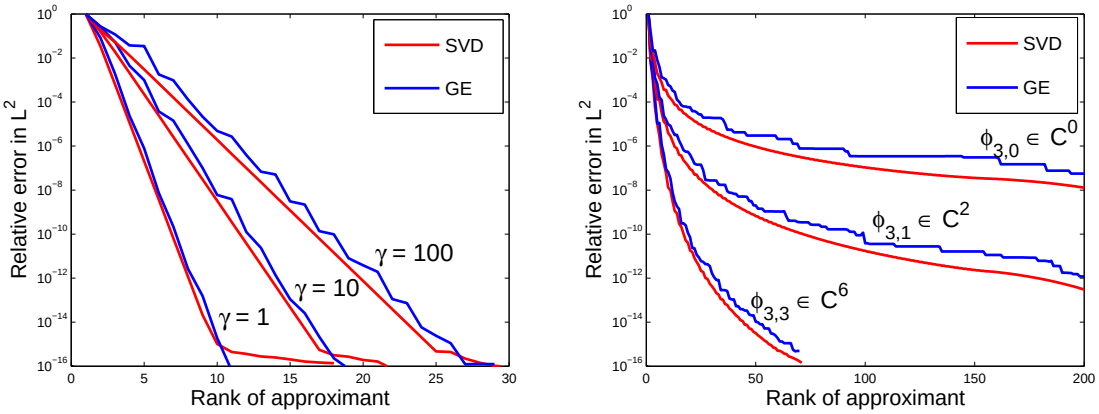


Figure 2.4: A comparison of SVD and GE algorithms shows the near-optimality of the latter for constructing low rank approximations to smooth bivariate functions. Left: 2D Runge functions. Right: Wendland's radial basis functions.

ones computed with the SVD. Here, we compare the L^2 errors of optimal low rank approximants with those constructed by our GE algorithm. In Figure 2.4 (left) we consider the 2D Runge functions given by

$$f(x,y) = \frac{1}{1+\gamma\left(x^2+y^2\right)^2}, \qquad \gamma = 1, 10, 100,$$

which are analytic in both variables, and (right) we consider Wendland's radial basis functions [165],

$$\phi_{3,s}(|x-y|) = \phi_{3,s}(r) = \begin{cases} (1-r)_+^2, & s = 0, \\ (1-r)_+^4\,(4r+1), & s = 1, \\ (1-r)_+^8\,(32r^3+25r^2+8r+1), & s = 3, \end{cases}$$

which have $2s$ continuous derivatives in both variables. Here, $(x)_+$ equals x if $x \geq 0$ and equals 0, otherwise. Theorem 3.1 and 3.2 explain the decay rates for the L^2 errors of the best low rank approximants.

2.1.3 Related literature

Ideas related to GE for functions have been developed by various authors under various names, though the connection with GE is usually not mentioned. We now briefly

summarize some of the ideas of *pseudoskeleton approximation* [66], *Adaptive Cross Approximation* (ACA) [12], *interpolative decompositions* [76], and *Geddes–Newton series* [31]. The moral of the story is that iterative GE has been independently discovered many times as a tool for low rank approximation, often under different names and guises.

2.1.3.1 Pseudoskeletons and Adaptive Cross Approximation

Pseudoskeletons approximate a matrix $A \in \mathbb{C}^{m\times n}$ by a matrix of low rank by computing the CUR decomposition[3] $A \approx CUR$, where $C \in \mathbb{C}^{m\times k}$ and $R \in \mathbb{C}^{k\times n}$ are subsets of the columns and rows of A, respectively, and $U \in \mathbb{C}^{k\times k}$ [66]. Selecting *good* columns and rows of A is of paramount importance, and this can be achieved via maximizing volumes [64], randomized techniques [45, 99], or ACA. ACA constructs a skeleton approximation with columns and rows selected adaptively [11]. The selection of a column and row corresponds to choosing a pivot location in GE, where the pivoting entry is the element belonging to both the column and row. If the first k columns and rows are selected, then for $A_{11} \in \mathbb{C}^{k\times k}$,

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} - \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} A_{11}^{-1} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & S \end{pmatrix}, \tag{2.3}$$

where $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is the Schur complement of A_{22} in A. The relation (2.3) is found in [12, p. 128] and when compared with [141, Thm. 1.4], it is apparent that ACA and GE are mathematically equivalent. This connection remains even when the columns and rows are adaptively selected [141, Theorem 1.8]. The continuous analogue of GE with complete pivoting is equivalent to the continuous analogue of ACA with adaptive column and row selection via complete pivoting. Bebendorf has used ACA to approximate bivariate functions [11] and has extended the scheme to the approximation of multivariate functions [13]. A comparison of approximation schemes that construct low rank approximations to functions is given in [15] and the connection to Gaussian elimination was explicitly given in [12, p. 147].

Theoretically, the analysis of ACA is most advanced for maximum volume pivoting, i.e., picking the pivoting locations to maximize the modulus of the determinant

[3]The pseudoskeleton literature writes $A \approx CGR$. More recently, it has become popular to follow the nomenclature of [45] and replace G by U.

of a certain matrix. Under this pivoting strategy, it has been shown that [65]

$$\|A - A_k\|_{\max} \leq (k+1)\sigma_{k+1}(A),$$

where A_k is the rank k approximant of A after k steps of GE with maximum volume pivoting, $\|\cdot\|_{\max}$ is the maximum absolute entry matrix norm, and $\sigma_{k+1}(A)$ is the $(k+1)$st singular value of A. Generally, maximum volme pivoting is not used in applications because it is computationally expensive. For error analysis of ACA under different pivoting strategies, see [12, 131].

In practice, pseudoskeleton approximation and ACA are used to construct low rank matrices that are derived by sampling kernels from boundary integral equations [12], and these ideas have been used by Hackbusch and others for efficiently representing matrices with hierarchical low rank [72, 73].

2.1.3.2 Interpolative decompositions

Given a matrix $A \in \mathbb{C}^{m\times n}$, an interpolative decomposition of A is an approximate factorization $A \approx A(:, J)X$, where $J = \{j_1, \ldots, j_k\}$ is a k-subset of the columns of A and $X \in \mathbb{C}^{k\times n}$ is a matrix such that $X(:, J) = I_k$. Here, the notation $A(:, J)$ denotes the $m \times k$ matrix obtained by horizontally concatenating the $j_1, \ldots, j_k$ columns of A. It is usually assumed that the columns of A are selected in such a way that the entries of X are bounded by 2 [36, 102]. It is called an interpolative decomposition because $A(:, J)X$ interpolates (coincides with) A along k columns.

More generally, a two-sided interpolative decomposition of A is an approximate factorization $A \approx WA(J', J)X$, where J' and J are k-subsets of the rows and columns of A, respectively, such that $W(:, J') = I_k$ and $X(:, J) = I_k$. Again, the matrices $X \in \mathbb{C}^{k\times n}$ and $W \in \mathbb{C}^{m\times k}$ are usually required to have entries bounded by 2.

Two-sided interpolative decompositions can be written as pseudoskeleton approximations, since

$$WA(J', J)X = (WA(J', J))\, A(J', J)^{-1}\, (A(J', J)X) = CUR,$$

where $C = A(:, J)$, $U = A(J', J)^{-1}$, and $R = A(J', :)$. A major difference is how the rows and columns of A are selected, i.e., the pivoting strategy in GE. Interpolative decompositions are often based on randomized algorithms [76, 103], whereas pseudoskeletons are, at least in principle, based on maximizing a certain determinant.

2.1.3.3 Geddes–Newton series

Though clearly a related idea, the theoretical framework for *Geddes–Newton series* was developed independently of the other methods discussed above [35]. For a function f and a *splitting point* $(a, b) \in [-1, 1]$ such that $f(a, b) \neq 0$, the *splitting operator* $\Upsilon_{(a,b)}$ is defined as

$$\Upsilon_{(a,b)} f(x, y) = \frac{f(x, b) f(a, y)}{f(a, b)}.$$

The splitting operator is now applied to the function,

$$f(x, y) - \Upsilon_{(a,b)} f(x, y) = f(x, y) - \frac{f(x, b) f(a, y)}{f(a, b)},$$

which is mathematically equivalent to one step of GE on f with the splitting point as the pivot location. This is then applied iteratively, and when repeated k times is equivalent to applying k steps of GE on f.

The main application of Geddes–Newton series so far is given in [31], where the authors use them to derive an algorithm for the quadrature of symmetric functions. Their integration algorithm first maps a function to one defined on $[0, 1]^2$ and then decomposes it into symmetric and anti-symmetric parts, ignoring the anti-symmetric part since it integrates to zero. The authors employ a very specialized pivoting strategy designed to preserve symmetry in contrast to our complete pivoting strategy, which we have found to be more robust.

2.2 Quadrature and other tensor product operations

In addition to approximating functions of two variables we would like to evaluate, integrate, and differentiate them. The GE algorithm on functions described in Section 2.1 allows us to construct the approximation

$$f(x, y) \approx \sum_{j=1}^{k} d_j c_j(y) r_j(x), \tag{2.4}$$

where $c_1, \ldots, c_k$ and $r_1, \ldots, r_k$ are Chebyshev interpolants of degree m and n, respectively. The approximant in (2.4) is an example of a *separable model*, in the sense that the x and y variables have been decoupled. This allows us to take advantage

of established 1D Chebyshev technology to carry out substantial computations much faster than one might expect.

For example, consider the computation of the definite double integral of $f(x,y)$ that can be computed by the `sum2` command in Chebfun2. From (2.4), we have

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y)dxdy \approx \sum_{j=1}^{k} d_j \left(\int_{-1}^{1} c_j(y)dy\right)\left(\int_{-1}^{1} r_j(x)dx\right),$$

which has reduced the 2D integral of f to a sum of $2k$ 1D integrals. These 1D integrals can be evaluated by calling the `sum` command in Chebfun, which utilizes Clenshaw–Curtis quadrature [38]. That is,

$$\int_{-1}^{1} r_j(x)dx = a_0^j + \sum_{s=1}^{\lfloor n/2\rfloor} \frac{2a_{2s}^j}{1-4s^2}, \qquad 1 \le j \le k,$$

where a_s^j is the sth Chebyshev expansion coefficient of r_j (see (1.4)). In practice, the operation is even more efficient because the pivot rows $r_1, \ldots, r_k$ are all of the same polynomial degree and thus, Clenshaw–Curtis quadrature can be implemented in a vectorized fashion.

This idea extends to any tensor product operation.

Definition 2.1. *A tensor product operator $\mathcal{L}$ is a linear operator on functions of two variables with the property that if $f(x,y) = g(y)h(x)$ then $\mathcal{L}(f) = \mathcal{L}_y(g)\mathcal{L}_x(h)$, for some operators $\mathcal{L}_y$ and $\mathcal{L}_x$. Thus, if f is of rank k then*

$$\mathcal{L}\left(\sum_{j=1}^{k} d_j c_j(y) r_j(x)\right) = \sum_{j=1}^{k} d_j \mathcal{L}_y(c_j)\mathcal{L}_x(r_j).$$

A tensor product operation can be achieved with $\mathcal{O}(k)$ calls to well-established 1D Chebyshev algorithms since $\mathcal{L}_y$ and $\mathcal{L}_x$ act on functions of one variable. Four important examples of tensor product operations are integration (described above), differentiation, evaluation, and the computation of bivariate Chebyshev coefficients.

Tensor product operations represent the ideal situation where well-established Chebyshev technology can be exploited for computing in two dimensions. Table 2.1 shows a selection of Chebfun2 commands that rely on tensor product operations.

Chebfun2 command	Operation
`sum`, `sum2`	integration
`cumsum`, `cumsum2`	cumulative integration
`prod`, `cumprod`	product integration
`norm`	L^2-norm
`diff`	partial differentiation
`chebpoly2`	2D discrete Chebyshev transform
`f(x,y)`	evaluation
`plot`, `surf`, `contour`	plotting
`flipud`, `fliplr`	reverse direction of coordinates
`mean2`, `std2`	mean, standard deviation

Table 2.1: A selection of scalar Chebfun2 commands that rely on tensor product operations. In each case the result is computed with greater speed than one might expect because the algorithms take advantage of the underlying low rank structure.

2.2.1 Partial differentiation

Partial differentiation is a tensor product operator since it is linear and for $N \geq 0$,

$$\frac{\partial^N}{\partial y^N}\left(\sum_{j=1}^{k} d_j c_j(y) r_j(x)\right) = \sum_{j=1}^{k} d_j \frac{\partial^N c_j}{\partial y^N}(y) r_j(x),$$

with a similar relation for partial differentiation in x. Therefore, derivatives of f can be computed by differentiating its pivot columns or rows using a recurrence relation [104, p. 34]. The fact that differentiation and integration can be done efficiently means that low rank technology is a powerful tool for vector calculus (see Section 2.4).

2.2.2 Function evaluation

Evaluation is a tensor product operation since it is linear and trivially satisfies Definition 2.1. Therefore, point evaluation of a rank k chebfun2 approximant can be carried out by evaluating $2k$ univariate Chebyshev expansion using Clenshaw's algorithm [37]. Evaluation is even more efficient since the pivot columns and rows are of the same polynomial degree and Clenshaw's algorithm can be implemented in a vectorized fashion.

2.2.3 Computation of Chebyshev coefficients

If `f` is a chebfun, then `chebpoly(f)` in Chebfun returns a column vector of the Chebyshev coefficients of `f`. Analogously, if `f` is a chebfun2, then `chebpoly2(f)` returns the matrix of bivariate Chebyshev coefficients of `f` and this matrix can be computed efficiently.

The computation of bivariate Chebyshev coefficients is a tensor product operation since

$$\texttt{chebpoly2}\left(\sum_{j=1}^{k} d_j c_j(y) r_j(x)\right) = \sum_{j=1}^{k} d_j \texttt{chebpoly}(c_j)\texttt{chebpoly}(r_j)^T,$$

and thus the matrix of coefficients in low rank form can be computed in $\mathcal{O}(k(m \log m + n \log n))$ operations, where the `chebpoly` command is computed by the discrete Chebyshev transform.

The inverse of this operation is function evaluation on a Chebyshev tensor product grid. This is also a tensor product operation, where the 1D pivot columns and rows are evaluated using the inverse discrete Chebyshev transform [57].

2.3 Other fundamental operations

Some operations, such as function composition and basic arithmetic operations are not tensor product operations. For these operations it is more challenging to exploit the low rank structure of a chebfun2 approximant.

2.3.1 Composition operations

Function composition operations apply one function pointwise to another to produce a third, and these are generally not tensor product operations. For instance, if `f` is a chebfun2, then we may want an approximant for `cos(f)`, `exp(f)`, or `f.^2`, representing the cosine, exponential, and pointwise square of a function. For such operations we use the GE algorithm, with the resulting function approximated without knowledge of the underlying low rank representation of `f`. That is, we exploit the fact that we can evaluate the function composition pointwise and proceed to use the GE algorithm to construct a new chebfun2 approximation of the resulting function. It is possible that the underlying structure of `f` could somehow be exploited to slightly

improve the efficiency of composition operations. We do not consider this any further here.

2.3.2 Basic arithmetic operations

Basic arithmetic operations are another way in which new functions can be constructed from existing ones, and are generally not tensor product operations. For example, if `f` and `g` are chebfun2 objects, then `f+g`, `f.*g`, and `f./g` represent binary addition, pointwise multiplication, and pointwise division. For pointwise multiplication and division we approximate the resulting function by using the GE algorithm in the same way as for composition operations. For pointwise division an additional test is required to check that the denominator has no roots.

Table 2.2 gives a selection of composition and basic arithmetic operations that use the GE algorithm without exploiting the underlying low rank representation.

Absent from the table is binary addition, for which we know how to exploit the low rank structure and find it more efficient to do so. In principle, binary addition $f + g$ requires no computation, only memory manipulation, since the pivot columns and rows of f and g could just be concatenated together; however, operations in Chebfun aim to approximate the final result with as few degrees of freedom as possible by removing negligible quantities that fall below machine precision. For binary addition the mechanics of this compression step are special.

More precisely, if f and g are represented as $f = \mathcal{C}_f D_f \mathcal{R}_f^T$ and $g = \mathcal{C}_g D_g \mathcal{R}_g^T$, then $h = f + g$ can be written as

$$h = \begin{bmatrix} \mathcal{C}_f & \mathcal{C}_g \end{bmatrix} \begin{pmatrix} D_f & 0 \\ 0 & D_g \end{pmatrix} \begin{bmatrix} \mathcal{R}_f^T \\ \mathcal{R}_g^T \end{bmatrix}. \tag{2.5}$$

This suggests that the rank of h is the sum of the ranks of f and g, but usually it can be approximated by a function of much lower rank. To compress (2.5) we first compute the QR factorizations of the two quasimatrices (see Section 1.7),

$$\begin{bmatrix} \mathcal{C}_f & \mathcal{C}_g \end{bmatrix} = \mathcal{Q}_L R_L, \qquad \begin{bmatrix} \mathcal{R}_f & \mathcal{R}_g \end{bmatrix} = \mathcal{Q}_R R_R,$$

Chebfun2 command	Operation
`.*`, `./`	multiplication, division
`cos`, `sin`, `tan`	trigonometric functions
`cosh`, `sinh`, `tanh`	hyperbolic functions
`exp`	exponential
`power`	integer powers

Table 2.2: A selection of composition and basic arithmetic operations in Chebfun2. In each case we use the GE algorithm to construct a chebfun2 approximant of the result that is accurate to relative machine precision.

and obtain the representation

$$h = \mathcal{Q}_L \underbrace{\left(R_L \begin{pmatrix} D_f & 0 \\ 0 & D_g \end{pmatrix} R_R^T \right)}_{=B} \mathcal{Q}_R^T.$$

Next, we compute the matrix SVD, $B = U\Sigma V^*$, so that

$$h = (\mathcal{Q}_L U)\, \Sigma \left(V^* \mathcal{Q}_R^T \right).$$

This is an approximate SVD for h and can be compressed by removing any negligible diagonal entries in Σ and the corresponding functions in $\mathcal{Q}_L U$ and $V^* \mathcal{Q}_R^T$. A discrete analogue of this algorithm for the addition of low rank matrices is described in [12, p. 17] and [14].

In principle, it is possible to derive similar compression algorithms for composition operations; however, it is only beneficial to do so when the upper bound on the rank of the resulting function is small.

2.4 Vector calculus operations

We can also represent vector valued functions defined on rectangles. Our convention is to use a lower case letter for a scalar function, f, and a capital letter for a vector function, $F = (f_1, f_2)^T$. We call an approximant to a vector valued function with two (or three) components a *chebfun2v*[4]. In this section we assume F has two components

[4]There is a object-oriented class called chebfun2v in Chebfun that represents vector-valued functions [161].

(for three components, see Section 2.5).

No new complications arise for vector valued functions since each component is represented as an independent scalar function approximated by a chebfun2. However, the operations that can be implemented are potentially very useful for applications. Many of the vector calculus operations on a chebfun2v rely on tensor product operations on its chebfun2 components, which in turn rely on 1D calculations with pivot columns and rows represented by chebfuns.

2.4.1 Algebraic operations

The basic non-differential operations of vector calculus are scalar multiplication fG, vector addition $F+G$, dot product $F\cdot G$, and cross product $F\times G$. We explain these operations in turn.

Scalar multiplication is the product of a scalar function with a vector function and algorithmically, it is achieved by two scalar function multiplications, one for each component.

Vector addition of two chebfun2v objects yields another chebfun2v and is computed by adding the two scalar components together using the compression algorithm described in Section 2.3.

The dot product takes two vector functions and returns a scalar function. Algebraically, it is an inner product, i.e., the sum of the products of the two components, and is computed directly from its definition. If the dot product of F and G is zero at (x_0, y_0), then the vectors F and G are orthogonal at (x_0, y_0).

The cross product is well-known as an operation on vector functions with three components, and it can also be defined when there are only two,

$$F\times G = f_1g_2 - f_2g_1, \qquad F = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}, \qquad G = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}.$$

Here, the cross product is a scalar function, which can be represented by a chebfun2. Algorithmically, we compute it directly from the relation $F\times G = f_1g_2 - f_2g_1$.

2.4.2 Differential operations

Vector calculus also involves various differential operators such as the gradient ∇f, curl $\nabla\times F$, divergence $\nabla\cdot F$, and Laplacian $\nabla^2 f$.

Command	Operation
`+`, `-`	addition, subtraction
`dot`	dot product
`cross`	cross product
`gradient`	gradient
`curl`	curl
`divergence`	divergence
`laplacian`	Laplacian
`quiver`	phase portrait

Table 2.3: Vector calculus commands in Chebfun2. In most cases the result is computed with greater speed than one might expect because of the exploitation of the componentwise low rank structure (see Section 2.2).

The gradient of a chebfun2 is a chebfun2v representing $\nabla f = (\partial f/\partial x, \partial f/\partial y)^T$, which is a vector that points in the direction of steepest ascent.

The curl of a vector function with two components is a scalar function defined by

$$\nabla \times F = \frac{\partial f_2}{\partial x} - \frac{\partial f_1}{\partial y}, \qquad F = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}.$$

If F is a vector valued function describing a velocity field of a fluid, then $(\nabla \times F)/2$ is a scalar function equal to the angular speed of a particle in the flow. A particle moving in a gradient field has zero angular speed and hence, $\nabla \times (\nabla f) = 0$.

The divergence of a vector function is defined as

$$\mathrm{div}(F) = \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y}, \qquad F = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}.$$

Divergence measures a vector field's distribution of sources or sinks. The Laplacian is closely related and is the divergence of the gradient, $\nabla^2 f = \nabla \cdot (\nabla f)$.

Table 2.3 summarizes the vector calculus commands available in Chebfun2.

2.4.3 Phase portraits

Aside from vector calculus operations, a vector valued function F can be visualized by a phase portrait.

A phase portrait of a vector field F is a graphical representation of a system of

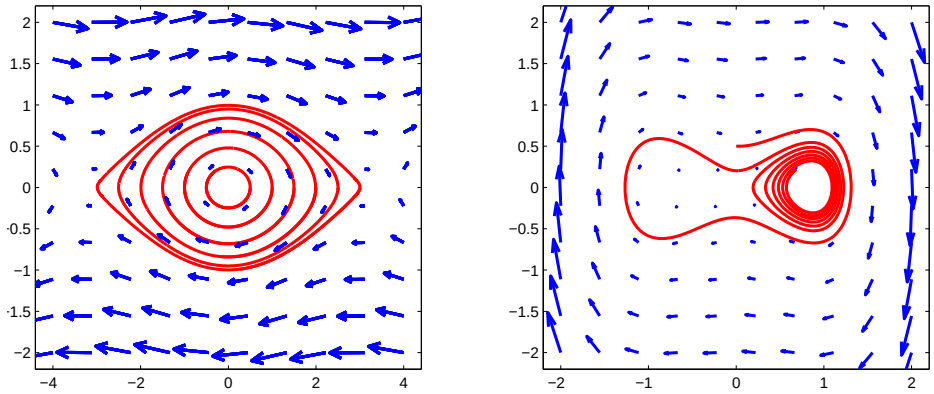


Figure 2.5: Phase portraits and example trajectories for the nonlinear pendulum (left) and the Duffing oscillator (right). Trajectories in the phase portraits (red lines) are computed using the `ode45` command in Chebfun2.

trajectories for the autonomous dynamical system $dx/dt = f_1(x, y)$, $dy/dt = f_2(x, y)$, where f_1 and f_2 are the components of F and each curve represents a different initial condition. In Chebfun2 these phase portraits can be plotted by the `quiver` command for the underlying vector field and the `ode45` command computes a representative trajectory when an initial condition is supplied (see Figure 2.5).

If `F` is a chebfun2v, then `ode45(F,tspan,y0)` solves the autonomous system $dx/dt = f_1(x, y)$, $dy/dt = f_2(x, y)$ with the prescribed time interval, `tspan`, and initial conditions, `y0`. This command returns a complex-valued chebfun encoding the trajectory in the form $x(t) + iy(t)$.

For example, Figure 2.5 (left) shows the phase portrait for the dynamical system $\dot{x} = y$, $\dot{y} = -\sin(x)/4$ representing a nonlinear pendulum, together with some sample trajectories, and Figure 2.5 (right) shows a trajectory in a phase portrait for the Duffing oscillator

$$\dot{x} = y, \qquad \dot{y} = \tfrac{4}{100}y - \tfrac{3}{4}x + x^3,$$

where the time interval is $[0, 50]$ and the initial condition is $x(0) = 0$, $y(0) = \frac{1}{2}$.

2.4.4 Green's Theorem

Green's Theorem expresses a double integral over a connected domain $D \subset \mathbb{R}^2$ with a simple piecewise smooth boundary as a line integral along its boundary, C, which is positively oriented. If F is a vector valued function with components that have

continuous partial derivatives, then by Green's Theorem [105, p. 96]

$$\iint_D \nabla \cdot F dx dy = \oint_C F \cdot \hat{\mathbf{n}}\, dC, \tag{2.6}$$

where $\hat{\mathbf{n}}$ is the outward pointing unit normal vector on the boundary. Therefore, to integrate any scalar function f over a region enclosed by a smooth closed curve we can construct a vector function F satisfying $f = \nabla \cdot F$, and then calculate a line integral instead. Green's Theorem is thus another convenient tool for computing a 2D quantity with 1D technology. The `integral2(f,c)` command in Chebfun2 employs this algorithm, where the boundary of D is encoded as a complex-valued chebfun, `c`.

Figure 2.6 (left) shows a complex-valued chebfun in the shape of a heart defined by $-16\sin(t)^3 + \mathrm{i}(13\cos(t) - 5\cos(2t) - 2\cos(3t) - \cos(4t))$ on $t \in [0, 2\pi]$. The `integral2(f,c)` command in Chebfun2 computes the volume between the surface of $z = f(x, y) = \cos(x) + y/100$ and $z = 0$ over the heart as 10.312461561235859.

2.5 Computing with surfaces embedded in $\mathbb{R}^3$*

Chebfun2 can also represent vector functions with three components, and these types of functions are useful for computing with parametrically defined surfaces (provided the parameters are defined on a rectangular domain). For example, the torus can be defined as

$$F(\theta, \phi) = \begin{pmatrix} (R + r\cos\phi)\cos\theta \\ (R + r\cos\phi)\sin\theta \\ r\sin\phi \end{pmatrix}, \qquad (\theta, \phi) \in [0, 2\pi]^2,$$

where R is the distance from the center of the tube to the center of the torus and r is the radius of the tube.

Some parametrically defined surfaces can be created by rotating a curve around an axis. If f is a function of one variable defined on $[-1, 1]$, then $F(u, v) = (f(u)\cos(v), f(u)\sin(v), u)^T$ with $(u, v) \in [-1, 1] \times [0, 2\pi]$ is the surface of revolution obtained by rotating f about the z-axis. If `f` is a chebfun, then the `cylinder` command in Chebfun returns a chebfun2v that represents the corresponding surface of revolution obtained by rotating `f` about the z-axis.

*The idea of using chebfun2v objects to represent parametric surfaces in Chebfun2 is due to Rodrigo Platte.

All the vector calculus operations described for vector functions of two components can be considered when there are three components, with similar underlying algorithms. Since the vector functions are constant in the third z-variable, standard vector calculus relations can usually be simplified, for example,

$$\nabla \times F = \begin{pmatrix} \partial f_3/\partial y \\ -\partial f_3/\partial x \\ \partial f_2/\partial x - \partial f_1/\partial y \end{pmatrix}, \qquad F(x,y) = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}.$$

One potentially useful quantity of a surface is its normal vector, which is the cross product of the partial derivatives of its parameterization, i.e.,

$$\text{normal}(F) = \frac{\partial F}{\partial x} \times \frac{\partial F}{\partial y}. \tag{2.7}$$

This vector can be scaled to have length 1, making it a unit normal vector. When F represents a surface, the normal vector to F is equal to the cross product of the tangential surface vectors. A challenging algorithmic issue is how to ensure that the normal vector is outward pointing, and currently the `normal` command in Chebfun2 returns a chebfun2v representing the normal vector that can have an inward or outward orientation [161].

Given a smooth vector function F representing a parametric surface that encloses a compact region $V \subset \mathbb{R}^3$, the volume of V can be expressed as a 3D integral, i.e., $\iiint_V dxdydz$. The divergence theorem [105, Sec. 5.1] can be employed to express the volume integral as the following surface integral:

$$\iiint_V dxdydz = \iiint_V (\nabla \cdot G)\, dxdydz = \oiint_F (G \cdot \hat{\mathbf{n}})\, dF,$$

where G is any continuously differentiable vector field in V with $\nabla \cdot G = 1$ and $\hat{\mathbf{n}}$ is the unit normal vector to F. We usually pick $G = (x, y, z)/3$ as this treats each variable equally. Further, suppose that $F = F(\theta, \psi)$. Then since $dF = |\partial F/\partial x \times \partial F/\partial y| d\theta d\psi$ we have

$$\oiint_F (G \cdot \hat{\mathbf{n}})\, dF = \iint_F \left(G \cdot \frac{\frac{\partial F}{\partial x} \times \frac{\partial F}{\partial y}}{\left| \frac{\partial F}{\partial x} \times \frac{\partial F}{\partial y} \right|} \right) \left| \frac{\partial F}{\partial x} \times \frac{\partial F}{\partial y} \right| d\theta d\psi = \iint_F (G \cdot \mathbf{n})\, d\theta d\psi,$$

where $\mathbf{n}$ is the unnormalized normal vector to F equal to the cross product of the

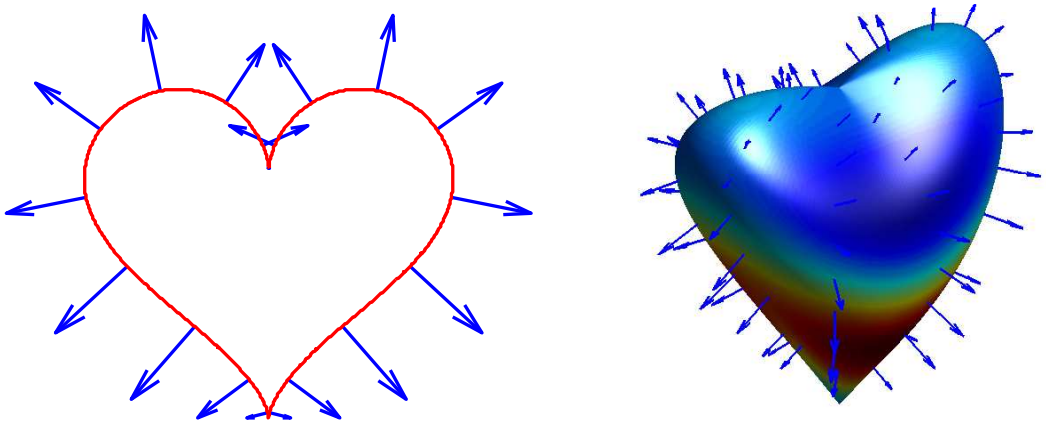


Figure 2.6: Left: Heart-shaped domain and the normal vector to the boundary, which can be used to integrate functions defined on the heart via Green's Theorem (see (2.6)). Right: Heart-shaped surface and the normal vector to the surface, which can be used to integrate functions defined inside the heart via the divergence theorem.

tangential surface vectors. The last integral can be used in practice even when the normal vector determined by (2.7) has zero length somewhere on the surface. This is important, for instance, when working with the sphere defined parametrically since by the Hairy Ball Theorem [108] there is no nonvanishing continuous tangent vector field on a sphere. Hence, regardless of the parameterization of the sphere the resulting normal vector computed by (2.7) has at least one point of zero length.

Figure 2.6 (right) shows a chebfun2v with three components representing a heart-shaped surface, which is parametrically defined as

$$F(\theta,\phi) = \begin{pmatrix} \sin(\pi\phi)\cos(\theta/2) \\ 0.7\sin(\pi\phi)\sin(\theta/2) \\ (\phi-1)(-49+50\phi+30\phi\cos(\theta)+\cos(2\theta))/(\cos(\theta)^2-25) \end{pmatrix},$$

where $(\theta,\phi) \in [0,1]\times[0,4\pi]$. The volume of F can be calculated by picking G so that $\nabla\cdot G = 1$, computing the normal vector to F using the cross product in (2.7), and then integrating the dot product of G with the surface normal vector. Chebfun2 computes the volume of the heart-shaped surface as 2.199114857512854.

Chapter 3

Low rank approximation theory

In this chapter we are interested in the theory of low rank function approximation. How fast does $\inf_{f_k} \|f - f_k\|_\infty$ decay to 0, where the infimum is taken over bounded rank k functions? What sort of functions are numerically of low rank? We discuss these questions and investigate some properties of the SVD of a function.

Throughout this chapter we consider continuous functions defined on $[-1, 1]^2$, but the discussion is analogous for any bounded rectangular domain $[a, b] \times [c, d] \subset \mathbb{R}^2$.

3.1 The rank of a bivariate polynomial

Let $p_{m,n}$ be a bivariate polynomial of degree (m, n), i.e., of degree m in x and degree n in y. Expressing $p_{m,n}$ in the tensor product monomial basis,

$$p_{m,n}(x, y) = \sum_{j=0}^{m} \sum_{i=0}^{n} \alpha_{ij} y^i x^j = \sum_{j=0}^{m} \left(\sum_{i=0}^{n} \alpha_{ij} y^i \right) x^j, \qquad \alpha_{ij} \in \mathbb{C}, \tag{3.1}$$

we see that $p_{m,n}$ is a sum of $m + 1$ functions of rank 1 and hence, is of rank at most $m + 1$. Similarly, by interchanging the summation in (3.1), we find that $p_{m,n}$ is of rank at most $n+1$. We conclude that the rank of $p_{m,n}$ is at most $\min(m, n)+1$. That is,

$$\operatorname{rank}(p_{m,n}) \leq \min(m, n) + 1, \qquad p_{m,n} \in \mathcal{P}_{m,n}, \tag{3.2}$$

where $\mathcal{P}_{m,n}$ is the set of bivariate polynomials of degree (m, n). Therefore, a best rank k approximation to $p_{m,n}$ is exact for $k \geq \min(m, n) + 1$.

Conversely, m and n are not bounded by any function of k. For example, consider the rank 1 function $x^m y^n$ where m and n are arbitrarily large.

3.2 The numerical rank and degree of a function

Inequality (3.2) can be extended to functions that are mathematically of infinite rank, but numerically of finite rank. Given a tolerance $\epsilon > 0$, we say that a function f has a *numerical rank* of k_ϵ if

$$k_\epsilon = \inf \left\{ k \in \mathbb{N} : \inf_{f_k} \|f - f_k\|_\infty \leq \epsilon \|f\|_\infty \right\},$$

where the inner infimum is taken over bounded rank k functions. By the Stone–Weierstrass Theorem [143], if f is continuous then it can be arbitrarily approximated by bivariate polynomials and hence, k_ϵ is well-defined for any $\epsilon > 0$.

For $\epsilon > 0$, we say that a function f has a *numerical degree* of (m_ϵ, n_ϵ) if

$$m_\epsilon = \inf \left\{ m \in \mathbb{N} : \lim_{n\to\infty} \left\|f - p^{\text{best}}_{m,n}\right\|_\infty \leq \epsilon \|f\|_\infty \right\},$$
$$n_\epsilon = \inf \left\{ n \in \mathbb{N} : \lim_{m\to\infty} \left\|f - p^{\text{best}}_{m,n}\right\|_\infty \leq \epsilon \|f\|_\infty \right\},$$

where $p^{\text{best}}_{m,n}$ is a best minimax polynomial approximant to f of degree at most (m, n) (see Theorem 1.3). By the Stone–Weierstrass Theorem again, the numerical degree of a continuous function is well-defined for any $\epsilon > 0$.

Now, fix $\epsilon > 0$ and suppose that f has numerical degree (m_ϵ, n_ϵ). The polynomial $p^{\text{best}}_{m_\epsilon,n}$ is of rank at most $m_\epsilon + 1$ for all n and hence

$$\inf_{f_{m_\epsilon+1}} \|f - f_{m_\epsilon+1}\|_\infty \leq \lim_{n\to\infty} \|f - p^{\text{best}}_{m_\epsilon,n}\|_\infty \leq \epsilon \|f\|_\infty,$$

or equivalently, $k_\epsilon \leq m_\epsilon + 1$. Similarly, we find that $k_\epsilon \leq n_\epsilon + 1$, and we conclude $k_\epsilon \leq \min(m_\epsilon, n_\epsilon) + 1$ (cf. (3.2)).

Conversely, m_ϵ and n_ϵ are not bounded by any function of k_ϵ, as the example $x^m y^n$ again shows.

3.3 Numerically low rank functions

Let $0 < \epsilon \leq 1/4$ and $f : [-1,1]^2 \to \mathbb{C}$ be a continuous function with numerical rank k_ϵ and numerical degree (m_ϵ, n_ϵ). Then, by definition, there exists a bounded rank k_ϵ function f_{k_ϵ} so that $\|f - f_{k_\epsilon}\|_\infty \leq 2\epsilon\|f\|_\infty$ and for sufficiently large m and n we have $\|f - p^{\text{best}}_{m_\epsilon,n}\|_\infty \leq \epsilon\|f\|_\infty$ and $\|f - p^{\text{best}}_{m,n_\epsilon}\|_\infty \leq \epsilon\|f\|_\infty$. Though it is not immediate it turns out that f_{k_ϵ} itself has a numerical degree that is close to (m_ϵ, n_ϵ) since

$$\left\|f_{k_\epsilon} - p^{\text{best}}_{m_\epsilon,n}\right\|_\infty \leq \|f - f_{k_\epsilon}\|_\infty + \left\|f - p^{\text{best}}_{m_\epsilon,n}\right\|_\infty \leq 3\epsilon\|f\|_\infty \leq 6\epsilon\|f_{k_\epsilon}\|_\infty,$$

where the last inequality comes from the reverse triangle inequality,

$$\|f_{k_\epsilon}\|_\infty \geq \|f\|_\infty - \|f - f_{k_\epsilon}\|_\infty \geq (1 - 2\epsilon)\|f\|_\infty \geq \frac{1}{2}\|f\|_\infty.$$

A similar argument shows that $\left\|f_{k_\epsilon} - p^{\text{best}}_{m,n_\epsilon}\right\|_\infty \leq 6\epsilon\|f_{k_\epsilon}\|_\infty$. Thus, we expect that f_{k_ϵ} can be numerically approximated by k_ϵ rank 1 sums, each one involving a polynomial in x and y of degree at most m_ϵ and n_ϵ, respectively. Therefore, roughly, $k_\epsilon\,(m_\epsilon + n_\epsilon)$ parameters are required to store a low rank approximation to f that has an accuracy of $\epsilon\|f\|_\infty$.

In comparison, if $p^{\text{best}}_{m_\epsilon,n_\epsilon}$ is stored as a polynomial of full rank, without taking account of any low rank structure, then about $m_\epsilon n_\epsilon$ parameters are required. We say that a function is *numerically of low rank* if it is more efficient to store the low rank approximation. (The following definition was motivated by an analogue for low rank matrices in [12, Def. 1.3].)

Definition 3.1. *For a fixed $\epsilon > 0$ a continuous function $f : [-1,1]^2 \to \mathbb{C}$ is numerically of low rank if $k_\epsilon\,(m_\epsilon + n_\epsilon) < m_\epsilon n_\epsilon$. If $k_\epsilon \approx \min(m_\epsilon, n_\epsilon)$ then f is numerically of full rank.*

If $m_\epsilon = n_\epsilon$, as is the case when $f(x,y) = f(y,x)$, then f is numerically of low rank when $n_\epsilon/k_\epsilon > 2$ and numerically of full rank when $n_\epsilon/k_\epsilon \approx 1$.

Unfortunately, it is usually theoretically very difficult to determine m_ϵ and n_ϵ precisely due to their connection with best minimax polynomial approximation. Instead, when considering examples we use the near-optimality property of Chebyshev approximation and calculate m^{cheb}_ϵ and n^{cheb}_ϵ, which are the numerical degrees required by

polynomial approximants constructed by Chebyshev projections. That is,

$$m_\epsilon^{\text{cheb}} = \inf\left\{m \in \mathbb{N} : \lim_{n\to\infty} \left\|f - p_{m,n}^{\text{proj}}\right\|_\infty \le \epsilon\|f\|_\infty\right\},$$
$$n_\epsilon^{\text{cheb}} = \inf\left\{n \in \mathbb{N} : \lim_{m\to\infty} \left\|f - p_{m,n}^{\text{proj}}\right\|_\infty \le \epsilon\|f\|_\infty\right\},$$

where $p_{m,n}^{\text{proj}}$ is the degree (m,n) Chebyshev projection of f formed by truncating the bivariate Chebyshev expansion of f (see Theorem 1.4). for us, m_ϵ^{cheb} and n_ϵ^{cheb} are more important than m_ϵ and n_ϵ since $k_\epsilon(m_\epsilon^{\text{cheb}} + n_\epsilon^{\text{cheb}})$ is roughly the number of parameters required to store a chebfun2 approximant of f, where by default $\epsilon = 2^{-52}$.

3.4 Results derived from 1D approximation theory

Let $k \ge 1$ be an integer, $f : [-1,1]^2 \to \mathbb{C}$ be a continuous function, and let $f_y : [-1,1] \to \mathbb{C}$ be defined as $f_y(x) = f(x,y)$ for each $y \in [-1,1]$. The convergence results in this section can be summarized by the following two statements:

1. If f_y is ν times continuously differentiable (and the ν derivative is of bounded variation) uniformly in y, then $\inf_{f_k} \|f - f_k\|_\infty = \mathcal{O}(k^{-\nu})$.

2. If f_y is analytic on $[-1,1]$ uniformly in y, then $\inf_{f_k} \|f - f_k\|_\infty = \mathcal{O}(\rho^{-k})$ for some $\rho > 1$.

Here, the infima are taken over bounded rank k functions and the same statements hold for the singular values σ_k of f. These convergence results are derived from the 1D convergence results in Theorem 1.1 and 1.2 using the inequalities $\inf_{f_{k-1}} \|f - f_{k-1}\|_\infty \le \lim_{n\to\infty} \left\|f - p_{k-2,n}^{\text{best}}\right\|_\infty$ and

$$\sigma_k \le 2 \lim_{n\to\infty} \left\|f - p_{k-2,n}^{\text{best}}\right\|_\infty \le 2 \sup_y \left\|f_y - p_{k-2}^{\text{proj}}(f_y)\right\|_\infty, \quad k \ge 2, \tag{3.3}$$

where $p_{k-2}^{\text{proj}}(f_y)$ is the degree $k-2$ Chebyshev projection of f_y (see Section 1.2).

Theorem 3.1 (Convergence for differentiable functions)**.** *Let $f : [-1,1]^2 \to \mathbb{C}$ be a continuous function, $\nu \ge 1$ an integer, and $V_f < \infty$ a constant. Suppose the functions $f_y(x) = f(x,y)$ satisfy the assumptions of Theorem* 1.1 *with total variation $V_f < \infty$,*

uniformly in y. Then for $k > \nu + 2$ we have

$$\inf_{f_k} \|f - f_k\|_\infty \leq \frac{2V_f}{\pi\nu(k-\nu-1)^\nu} = \mathcal{O}(k^{-\nu}), \qquad \sigma_k \leq \frac{4V_f}{\pi\nu(k-\nu-2)^\nu} = \mathcal{O}(k^{-\nu}),$$

where the infimum is taken over bounded rank k functions.

Proof. For each $y \in [-1,1]$, $f_y(x)$ satisfies the assumptions in Theorem 1.1 so for $k > \nu + 1$ we have

$$f(x,y) = f_y(x) = \sum_{j=0}^{\infty} a_j(y)T_j(x), \qquad \left\| f_y - \sum_{j=0}^{k-1} a_j(y)T_j \right\|_\infty \leq \frac{2V_f}{\pi\nu(k-\nu-1)^\nu}.$$

Moreover, since this holds uniformly in y we have

$$\inf_{f_k} \|f - f_k\|_\infty \leq \sup_{y\in[-1,1]} \left\| f_y - \sum_{j=0}^{k-1} a_j(y)T_j \right\|_\infty \leq \frac{2V_f}{\pi\nu(k-\nu-1)^\nu},$$

which proves the first inequality. For the second we have by (1.10)

$$\sigma_k \leq \left(\sum_{j=k}^{\infty} \sigma_j^2 \right)^{1/2} \leq 2 \sup_{y\in[-1,1]} \left\| f_y - \sum_{j=0}^{k-2} a_j(y)T_j \right\|_\infty \leq \frac{4V_f}{\pi\nu(k-\nu-2)^\nu}.$$

□

The same conclusion holds in Theorem 3.1 when the roles of x and y are swapped. Therefore, if $f(x,y)$ satisfies the assumptions of Theorem 3.1 in one of its variables, then σ_k and $\inf_{f_k} \|f - f_k\|_\infty$ decay like $\mathcal{O}(k^{-\nu})$. Related results appear in [39, 40] under a Hölder continuity condition, in [50, 123, 124] for positive definite kernels, and in [82, 166] for the decay of eigenvalues rather than singular values.

An analogous theorem holds when f_y is analytic on $[-1,1]$.

Theorem 3.2 (Convergence for analytic functions)**.** *Let $f : [-1,1]^2 \to \mathbb{C}$ be a continuous function, $M < \infty$ a constant, and E_ρ an open Bernstein ellipse. Suppose the functions $f_y(x) = f(x,y)$ satisfy the assumptions of Theorem* 1.2 *with M and E_ρ, uniformly in y. Then for $k \geq 2$ we have*

$$\inf_{f_k} \|f - f_k\|_\infty \leq \frac{2M\rho^{-k+1}}{\rho - 1} = \mathcal{O}(\rho^{-k}), \qquad \sigma_k \leq \frac{4M\rho^{-k+2}}{\rho - 1} = \mathcal{O}(\rho^{-k}),$$

where the infimum is taken over bounded rank k functions.

Proof. Use the same reasoning as in the proof of Theorem 3.1, but with the 1D convergence results from Theorem 1.2. □

The same conclusion holds in Theorem 3.2 when the roles of x and y are swapped and therefore, if $f(x, y)$ is a function that is analytic in $[-1, 1]$ in at least one of its variables, then σ_k and $\inf_{f_k} \|f - f_k\|_\infty$ decay geometrically. A closely related proof shows that the eigenvalues of symmetric analytic kernels decay geometrically [94].

Theorems 3.1 and 3.2 relate the smoothness of a function of two variables to the decay of $\inf_{f_k} \|f - f_k\|_\infty$, and are near-optimal in the sense that one can not hope for a better asymptotic rate of convergence with the same assumptions. However, Theorems 3.1 and 3.2 do not guarantee that a function is numerically of low rank, since the same reasoning can be used to bound the numerical degree (see the sequences of inequalities in (3.3)).

3.5 Numerically low rank functions in the wild

After substantial experience, one can predict with reasonable success whether or not a given function is numerically of low rank. The easiest ones to spot are functions already written as sums of rank 1 functions, such as

$$1 + xy^2 + x^2y^2, \quad \cos(x)\sin(y) + e^{10x}e^{10y}, \quad |x|\mathrm{Ai}(10y),$$

though care is required since according to our definition the first is not numerically of low rank (the polynomial degree is also small[1]). Other simple examples are powers and exponentials,

$$(2x + 3y)^4, \quad e^{x^2+y^2}, \quad \log(x + y),$$

which can easily be expressed in low rank form (again, for the same reason, the first is not numerically of low rank). Compositions of trigonometric and exponential functions such as $\cos(10(x + y))$ and $e^{\sin(10(x^2+y^2))}$ are also usually of low rank due to trigonometric identities. These examples can often be combined to build and spot more complicated functions that are numerically of low rank. This is an *ad hoc* process.

[1]For any $\epsilon > 0$, the numerical rank of $1 + xy^2 + x^2y^2$ is 2, the numerical degree is $(2, 2)$, and $8 = k_\epsilon(m_\epsilon + n_\epsilon) > m_\epsilon n_\epsilon = 4$ (see Definition 3.1).

Low rank functions do not necessarily have all their "action" aligned with the coordinate axes (see Section 3.7.1 for an example) and can be highly non-smooth. Therefore, predicting if a function is numerically of low rank without involved calculations is full of potential pitfalls. Often, the only assured way is to numerically approximate the function to determine k_ϵ, m_ϵ, and n_ϵ and verify that $k_\epsilon(m_\epsilon + n_\epsilon) < m_\epsilon n_\epsilon$.

3.6 A mixed Sobolev space containing low rank functions

For any $\epsilon > 0$, the set of functions that are numerically of low rank is not a vector space since it is not closed under addition, but here we exhibit a vector space that contains many good candidates. If a function is somewhat aligned along a coordinate axis, then it is often of low rank and these conditions can be partially captured by a *mixed Sobolev space*. We define the mixed Sobolev space $\mathcal{H}^t_{\text{mix}} = \mathcal{H}^t_{\text{mix}}([-1,1]^2)$ for $t > 1$ by

$$\mathcal{H}^t_{\text{mix}} = \left\{ f(x,y) = \sum_{i,j=0}^{\infty} a_{ij} T_i(y) T_j(x) : \|f\|_{\mathcal{H}^t_{\text{mix}}} = \sum_{i,j=0}^{\infty} (1+i)^t (1+j)^t |a_{ij}| < \infty \right\},$$

which imposes the condition that the matrix of bivariate Chebyshev coefficients, $A = (a_{ij})$, has diagonal entries that have an algebraic decay of $\mathcal{O}(n^{-2t})$, while the individual columns and rows have entries that decay at half that order. The space $\mathcal{H}^t_{\text{mix}}$ is important in hyperbolic cross approximation and is usually stated in terms of the Fourier basis rather than the Chebyshev basis [136].

Let $f \in \mathcal{H}^t_{\text{mix}}$ and suppose that $f(x,y) = \sum_{i,j=0}^{\infty} a_{ij} T_i(y) T_j(x)$ is the Chebyshev expansion of f. Then, the approximation power of the hyperbolic cross approximant

$$p_{I_d} = \sum_{(i,j) \in I_d} a_{ij} T_i(y) T_j(x), \qquad I_d = \{(i,j) \in \mathbb{N}^2 : (1+i)(1+j) \leq 1+d\}, \quad d > 0,$$

is the same as $p^{\text{proj}}_{d,d}$, where $p^{\text{proj}}_{d,d}$ is the degree (d,d) Chebyshev projection of f.

Lemma 3.1. *Let $t > 1$ and $f \in \mathcal{H}^t_{\text{mix}}$. For any $d > 0$ we have*

$$\left\| f - p^{\text{proj}}_{d,d} \right\|_\infty \leq (1+d)^{-t} \|f\|_{\mathcal{H}^t_{\text{mix}}}, \qquad \|f - p_{I_d}\|_\infty \leq (1+d)^{-t} \|f\|_{\mathcal{H}^t_{\text{mix}}}.$$

Proof. Let $f(x,y) = \sum_{i,j=0}^{\infty} a_{ij} T_i(y) T_j(x)$, $I \subset \mathbb{N}^2$ an indexing set, and define $p_I = \sum_{(i,j)\in I} a_{ij} T_i(y) T_j(x)$. Since $\|T_i\|_\infty \leq 1$ we have,

$$\|f - p_I\|_\infty \leq \sum_{(i,j)\notin I} |a_{ij}| \leq \max_{(i,j)\notin I} \left(\frac{1}{(1+i)^t (1+j)^t} \right) \left(\sum_{(i,j)\notin I} (1+i)^t (1+j)^t |a_{ij}| \right).$$

If $I = \{0,\ldots,d\}^2$ then $p_I = p_{d,d}^{\text{proj}}$ and $\|f - p_{d,d}^{\text{proj}}\|_\infty \leq (1+d)^{-t} \|f\|_{\mathcal{H}^t_{\text{mix}}}$. If $I = I_d$ then for $(i,j) \in I_d$ we have $(1+i)(1+j) \leq 1+d$ and hence, $\|f - p_{I_d}\|_\infty \leq (1+d)^{-t} \|f\|_{\mathcal{H}^t_{\text{mix}}}$. □

If $f \in \mathcal{H}^t_{\text{mix}}$ then it is usually more efficient to use hyperbolic cross approximation than Chebyshev projection since $|I_d| = \mathcal{O}(d \log d)$. Similar approximation results exist with the Fourier basis in the sparse grid literature, which usually focus on high dimensions and overcoming the curse of dimensionality [93]. Here, in two dimensions, we note that the approximant p_{I_d} is usually of low rank.

Lemma 3.2. *Let $d \geq 17$. A bivariate polynomial p_{I_d} of exact degree (d,d) with nonzero Chebyshev coefficients in I_d is of low rank.*

Proof. Let k denote the rank of p_{I_d}. Then k is equal to the rank of its matrix of bivariate Chebyshev coefficients. This matrix has at most $(d+1)^{1/2}$ nonzero columns and rows and hence, is of rank at most $2(d+1)^{1/2}$. Since $d \geq 17$ we have $4(d+1)^{1/2} < d$ and hence,

$$2kd \leq 4(d+1)^{1/2} d < d^2.$$

We conclude that p_{I_d} is of low rank (see Definition 3.1). □

Therefore, functions in $\mathcal{H}^t_{\text{mix}}$ with $t > 1$ are well-approximated by low rank polynomials and are candidates for being numerically of low rank themselves. In many ways low rank technology is applicable to a larger range of functions than hyperbolic cross approximation, but is more challenging to extend to higher dimensions. A comparison of low rank approximation and hyperbolic cross approximation is given in [69].

3.7 Three examples

The examples given in Section 3.5 do not capture the rich variety of functions that are numerically of low rank. Here, we give three examples to learn more. The first

b/a	k_ϵ	n_ϵ^{cheb}	$n_\epsilon^{\text{cheb}}/k_\epsilon$
1.1	4	8	2.0
10	12	36	3.0
100	18	110	6.1
1000	25	334	13.4
$b/a \gg 1$	$\mathcal{O}(\log(b/a)\log(\epsilon^{-1}))$	$\mathcal{O}(\sqrt{b/a}\,\log(\epsilon^{-1}))$	$\mathcal{O}(\sqrt{b/a}/\log(b/a))$

Table 3.1: Numerical rank and degree of the symmetric Cauchy function on $[a,b]^2$ with $\epsilon = 2^{-52}$. Despite this function having no obvious low rank structure we find that $k_\epsilon = \mathcal{O}(\log(b/a)\log(\epsilon^{-1}))$ and $n_\epsilon^{\text{cheb}} = \mathcal{O}(\sqrt{b/a}\,\log(\epsilon^{-1}))$ for $b/a \gg 1$.

shows that numerically low rank functions do not necessarily have any obvious low rank structure, the second gives a sum of N linearly independent rank 1 functions that is numerically of rank much less than N, and the last is a full rank function that appears to be of lower rank. These examples are slightly paradoxical and serve as a warning.

3.7.1 The symmetric Cauchy function

The symmetric *Cauchy function* or *Cauchy kernel* is defined as

$$f(x,y) = \frac{1}{x+y}, \qquad (x,y) \in [a,b]^2, \qquad 0 < a < b, \tag{3.4}$$

and since f is symmetric we proceed to show that it is numerically of very low rank, i.e., $n_\epsilon^{\text{cheb}}/k_\epsilon \gg 1$ (see Section 3.3), when b/a is large.

A hint that (3.4) is numerically of low rank comes from taking $[a,b] = [1/2, n-1/2]$ and evaluating the Cauchy function at n equally spaced points. The resulting matrix of samples is the notoriously ill-conditioned $n \times n$ Hilbert matrix [16]. Table 3.1 confirms that the function is numerically of low rank when $b/a \geq 1.1$.

To determine n_ϵ^{cheb} we note that the numerical degree of $f(x,y)$ is more than the numerical degree of $f(x,a) = 1/(x+a)$ on $x \in [a,b]$. The Chebyshev projection of $1/(x+a)$ on $[a,b]$ can be explicitly calculated by first transplanting onto $[-1,1]$ and applying Lemma A.1. We obtain

$$\frac{1}{\frac{b-a}{2}(x+1)+2a} = {\sum_{j=0}^{\infty}}' \frac{4/(b-a)(-1)^j T_j(x)}{\sqrt{A^2-1}\left(A+\sqrt{A^2-1}\right)^j}, \qquad A = \frac{b+3a}{b-a},$$

where the prime indicates that the first term of the summation is halved. Since the coefficients decay geometrically, n_ϵ^{cheb} can be estimated by solving $|\alpha_j| \leq \epsilon/2a$ for j. Thus, we have

$$n_\epsilon^{\text{cheb}} \approx \left\lceil \frac{\log\left(\frac{8a\epsilon^{-1}}{\sqrt{A^2-1}(b-a)}\right)}{\log\left(A+\sqrt{A^2-1}\right)} \right\rceil,$$

which is numerically observed to be satisfied as an equality. Letting $r = b/a$ we have

$$n_\epsilon^{\text{cheb}} \approx \left\lceil \frac{\log\left(\frac{8\epsilon^{-1}}{\sqrt{B^2-1}(r-1)}\right)}{\log\left(B+\sqrt{B^2-1}\right)} \right\rceil, \qquad B = \frac{r+3}{r-1},$$

showing that the numerical degree of f only depends on b/a. Furthermore, for $b/a \gg 1$ we have

$$n_\epsilon^{\text{cheb}} = \mathcal{O}\left(\sqrt{b/a}\log\left(\epsilon^{-1}\right)\right).$$

For the numerical rank, k_ϵ, we have the following result:

Lemma 3.3. *Let $0 < \epsilon < 1$ and $f(x,y) = 1/(x+y)$ on $[a,b]^2$. Then there exists a function $g : [a,b]^2 \to \mathbb{R}$ of rank k_ϵ such that*

$$\|f-g\|_\infty \leq \epsilon \|f\|_\infty, \qquad k_\epsilon = \mathcal{O}\left(\log(b/a)\log\left(\epsilon^{-1}\right)\right).$$

Proof. See Lemma 1 of [68], which proves the rank of the symmetric Cauchy matrix (discrete analogue of this lemma). The proof of Lemma 1 in [68] relies on [68, eq. 16], which is exactly the statement here. □

This shows that the symmetric Cauchy function is numerically of low rank for $b/a \gg 1$ since $n_\epsilon^{\text{cheb}}/k_\epsilon = \mathcal{O}(\sqrt{b/a}/\log(b/a))$. In practice, as shown in Table 3.1, the symmetric Cauchy function is numerically of low rank when $b/a > 1.1$.

The function $1/(x+y)$ on $[a,b]^2$ is an example of a low rank function without any obvious low rank structure or alignment with the coordinate axes.

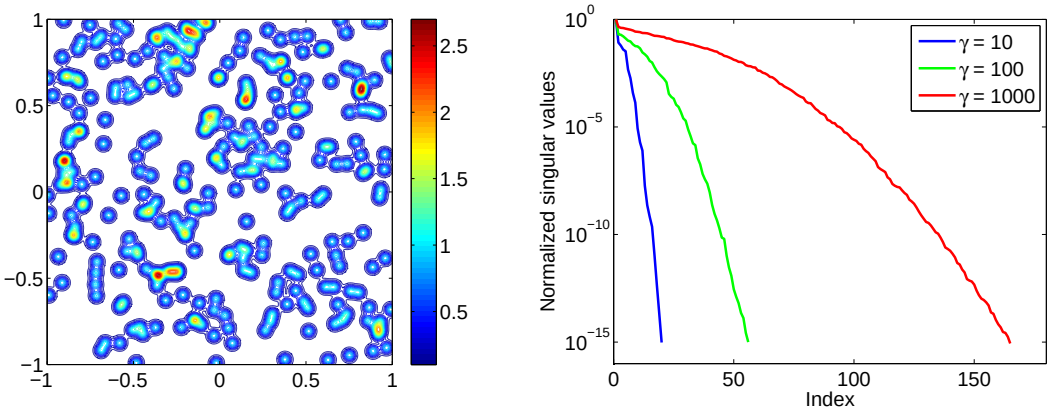


Figure 3.1: Left: Three hundred arbitrarily centered Gaussian bump functions added together, as in (3.5), with $\gamma = 1000$. Right: Super-geometric decay of the normalized singular values (the first singular value is scaled to be 1) for $\gamma = 10, 100, 1000$. Mathematically, (3.5) with $N = 300$ is of rank 300, but it can be approximated to machine precision by functions of rank 21, 59, and 176, respectively.

3.7.2 A sum of Gaussian bumps

For the second example, consider adding N Gaussian bumps together, centered at arbitrary locations $(x_1, y_1), \ldots, (x_N, y_N)$ in $[-1, 1]^2$,

$$f(x, y) = \sum_{j=1}^{N} e^{-\gamma\left((x-x_j)^2+(y-y_j)^2\right)}, \tag{3.5}$$

where $\gamma > 0$ is a positive real parameter. A Gaussian bump is a rank 1 function and hence, mathematically, f is usually of rank N; however, when N is large the numerical rank of f is observed to be much smaller. In Figure 3.1 (left) we display a contour plot for the case $\gamma = 1000$ and $N = 300$ and (right) we show that the singular values of f decay super-geometrically for $\gamma = 10, 100, 1000$.

The numerical degree of f is approximately equal to the maximum numerical degree of its individual terms in (3.5), and thus n_ϵ^{cheb} is approximately equal to the numerical degree of $e^{-\gamma x^2}$. The Chebyshev expansion of $e^{-\gamma x^2}$ can be calculated explicitly, and by Lemma A.2 we have

$$e^{-\gamma x^2} = {\sum_{j=0}^{\infty}}' (-1)^j \frac{2I_j(\gamma/2)}{e^{\gamma/2}} T_{2j}(x),$$

where $I_j(z)$ is the modified Bessel function of the first kind with parameter j and

γ	k_ϵ	n_ϵ^{cheb}	$n_\epsilon^{\text{cheb}}/k_\epsilon$
10	21	49	2.3
100	59	123	2.1
1000	176	371	2.1
10000	523	1151	2.2
$\gamma \gg 1$	$\mathcal{O}(\gamma^{1/2})$	$\mathcal{O}(\gamma^{1/2})$	≈ 2

Table 3.2: Numerical rank and degree of a sum of 1000 Gaussian bumps. The numerical rank and degree are observed to grow like $\mathcal{O}(\gamma^{1/2})$.

the prime indicates that the first term is halved. To find the leading term of n_ϵ^{cheb} as $\gamma \to \infty$ we solve $2I_j(\gamma/2)e^{-\gamma/2} \sim \epsilon$ for j by replacing $I_j(\gamma/2)$ with the asymptotic formula in [1, (9.7.1)], obtaining

$$\frac{2}{\sqrt{\pi\gamma}}\left(1 - \frac{4j^2-1}{4\gamma}\right) \sim \epsilon, \qquad \gamma \gg 1.$$

After rearranging the above expression we conclude that $j^2 \sim \gamma - \sqrt{\pi}\epsilon\gamma^{3/2}/2$, or equivalently, $j \sim \gamma^{1/2} - \sqrt{\pi}\epsilon\gamma/4$. Therefore, n_ϵ^{cheb} of (3.5) satisfies

$$n_\epsilon^{\text{cheb}} \sim 2\gamma^{1/2} - \sqrt{\pi}\epsilon\gamma/2,$$

which is in agreement with the numerical observations in Table 3.2.

For the numerical rank, k_ϵ, we note that $k_\epsilon \le n_\epsilon + 1$ (see Section 3.2) and hence, $k_\epsilon = \mathcal{O}(\gamma^{1/2})$. Therefore, despite the fact that the N rank 1 terms in (3.5) are usually linearly independent, the numerical rank of (3.5) is less than N for any $\epsilon > 0$ when $N \gg \gamma^{1/2}$.

3.7.3 A 2D Fourier-like function

For the final example, we consider the function $f(x,y) = e^{\mathrm{i}M\pi xy}$ defined on $[-1,1]^2$, where $M \ge 1$ is a real parameter. This is an example that is of full numerical rank, i.e., $n_\epsilon/k_\epsilon \approx 1$, but we observe that $n_\epsilon^{\text{cheb}}/k_\epsilon \approx \pi/2$.

First, we determine the (Chebyshev) numerical degree, n_ϵ^{cheb}, of $e^{\mathrm{i}M\pi xy}$. By

Lemma A.3 the bivariate Chebyshev expansion of f is

$$e^{\mathrm{i}M\pi xy} = \sum_{p=0}^{\infty}{}' \sum_{q=0}^{\infty}{}' a_{pq} T_p(y) T_q(y),$$

$$a_{pq} = \begin{cases} 4\mathrm{i}^q J_{(q+p)/2}(M\pi/2) J_{(q-p)/2}(M\pi/2), & \mathrm{mod}(|p-q|,2) = 0, \\ 0, & \text{otherwise}, \end{cases}$$

where the primes indicate that the first terms are halved and $J_\nu(z)$ is the Bessel function with parameter ν. The Bessel function $J_\nu(z)$ oscillates when $z > \nu$ and rapidly decays to 0 when $z < \nu$ and hence, the coefficients a_{pq} rapidly decay to zero when $(p+q)/2 > M\pi/2$ or $(p-q)/2 > M\pi/2$. We conclude that for $M \gg 1$ we have $n_\epsilon^{\mathrm{cheb}} \sim M\pi$.

For the numerical rank, k_ϵ, we have the following lemma.

Lemma 3.4. *Let $\epsilon > 0$. The function $e^{\mathrm{i}M\pi xy}$ has a numerical rank k_ϵ satisfying $k_\epsilon/2M \to c$ for a constant $c \leq 1$ as $M \to \infty$.*

Proof. Let p_{k-1}^{best} and q_{k-1}^{best} be the best minimax polynomial approximations of degree $k-1$ to $\cos(M\pi t)$ and $\sin(M\pi t)$ on $[-1,1]$, respectively, and define $s_{k-1} = p_{k-1}^{\mathrm{best}} + \mathrm{i}q_{k-1}^{\mathrm{best}}$. Note that $s_{k-1}(xy)$ is of rank at most k so that

$$\inf_{f_k} \left\| e^{\mathrm{i}M\pi xy} - f_k \right\|_\infty \leq \left\| e^{\mathrm{i}M\pi xy} - s_k(xy) \right\|_\infty = \left\| e^{\mathrm{i}M\pi t} - s_k(t) \right\|_\infty,$$

where the infimum is taken over bounded rank k functions. Moreover, since $(x,y) \in [-1,1]^2$ implies that $t = xy \in [-1,1]$ we have $\|e^{\mathrm{i}M\pi xy} - s_k(xy)\|_\infty = \|e^{\mathrm{i}M\pi t} - s_k(t)\|_\infty$. Furthermore, we have $e^{\mathrm{i}M\pi t} = \cos(M\pi t) + \mathrm{i}\sin(M\pi t)$ and so

$$\left\| e^{\mathrm{i}M\pi t} - s_k(t) \right\|_\infty \leq \left\| \cos(M\pi t) - p_k^{\mathrm{best}}(t) \right\|_\infty + \left\| \sin(M\pi t) - q_k^{\mathrm{best}}(t) \right\|_\infty.$$

By the equioscillation theorem [122, Thm. 7.4] $p_k^{\mathrm{best}} = 0$ for $k \leq 2\lfloor M \rfloor - 1$ since $\cos(M\pi t)$ equioscillates $2\lfloor M \rfloor + 1$ times in $[-1,1]$. Similarly, $\sin(M\pi t)$ equioscillates $2\lfloor M \rfloor$ times in $[-1,1]$ and hence, $q_k^{\mathrm{best}} = 0$ for $k \leq 2\lfloor M \rfloor - 2$. However, for $k > 2\lfloor M \rfloor - 1$, $\|\cos(M\pi t) - p_k^{\mathrm{best}}(t)\|_\infty$ and $\|\sin(M\pi t) - q_k^{\mathrm{best}}(t)\|_\infty$ decay super-geometrically to zero as $k \to \infty$ and hence, the numerical rank of $e^{\mathrm{i}M\pi xy}$ satisfies $k_\epsilon/2M \to c$ for a constant $c \leq 1$ as $M \to \infty$. □

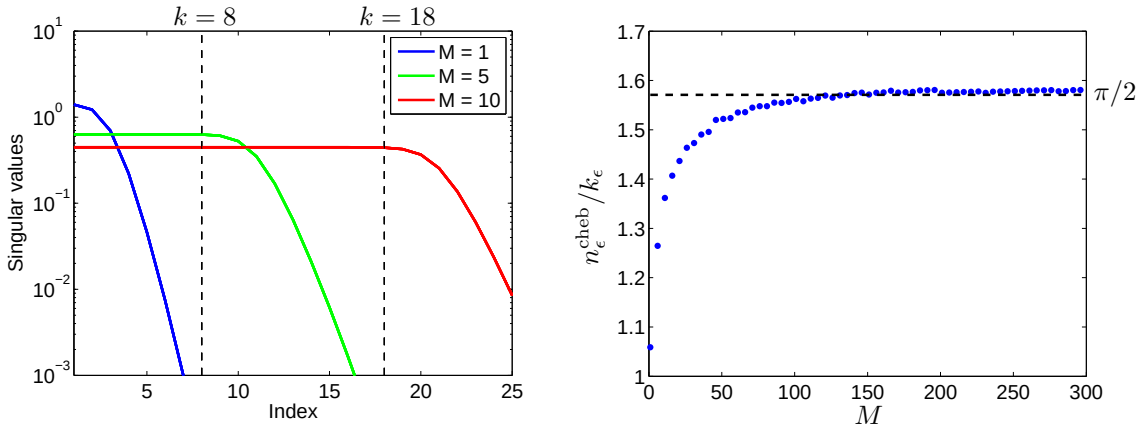


Figure 3.2: Left: The initial decay of the singular values of $e^{\mathrm{i}M\pi xy}$. For $k \le 2\lfloor M \rfloor - 2$ there is no decay of the singular values. For $k > 2\lfloor M \rfloor - 2$ the singular values decay super-geometrically. Right: The ratio $n_\epsilon^{\text{cheb}}/k_\epsilon$. As $M \to \infty$ the ratio tends to a number that is very close to $\pi/2$.

Therefore, for any $\epsilon > 0$ and we have $n_\epsilon^{\text{cheb}}/k_\epsilon \to c$ for a constant $c \ge M\pi/(2M) = \pi/2$. In practice for large M we observe $n_\epsilon^{\text{cheb}}/k_\epsilon \approx \pi/2$.

Figure 3.2 (left) shows the initial decay of the singular values of $e^{\mathrm{i}M\pi xy}$. We observe that the singular values do not decay until $k > 2\lfloor M \rfloor - 2$ after which they decay super-geometrically to zero, confirming the reasoning in Lemma 3.4. In Figure 3.2 (right) we plot $n_\epsilon^{\text{cheb}}/k_\epsilon$ for $M = 1, \ldots, 300$, where ϵ is machine precision, and find that the ratio tends to a number that is very close to $\pi/2$.

The manifestation of the $\pi/2$ factor is a phenomenon that can be explained by the near-optimality of Chebyshev interpolation. The Shannon–Nyquist sampling theorem rate says that to recover an oscillatory function it must be sampled at a minimum of 2 points per wavelength [132]. Chebyshev points on $[-1, 1]$ are clustered at ± 1 and near 0 they are $\pi/2$ times less dense than the same number of equally spaced points on $[-1, 1]$. Therefore, for oscillatory functions Chebyshev interpolation can require about $\pi/2$ times more points than is strictly necessary. This means that $n_\epsilon^{\text{cheb}} \approx (\pi/2)n_\epsilon$ and hence, for this function $n_\epsilon/k_\epsilon \approx 1$ while $n_\epsilon^{\text{cheb}}/k_\epsilon \approx \pi/2$. This phenomenon occurs surprisingly often, even for nonoscillatory functions, due to the fact that the pivot columns and rows in the GE algorithm (see Section 2.1) have many zeros by Theorem 2.1 and hence become highly oscillatory. This phenomenon also arises for many functions if the SVD is used to construct best low rank approximations with respect to the L^2-norm as the singular vectors of a function can be oscillatory [90].

In Chebfun2 some functions become numerically of low rank in practice due to this phenomenon. For instance, for $\cos(M\pi xy)$ it can be shown that $n_\epsilon/k_\epsilon \approx 2$ while

$n_\epsilon^{\text{cheb}}/k_\epsilon \approx \pi > 2$ where $M \gg 1$.

3.8 The singular value decomposition of a function

No discussion about low rank approximation is complete without the SVD, which in our context is the SVD of a bivariate function.

Definition 3.2. *Let $f : [a,b] \times [c,d] \to \mathbb{C}$ be a continuous function. An SVD of f is a series*

$$f(x,y) = \sum_{j=1}^{\infty} \sigma_j u_j(y)\overline{v_j(x)}, \qquad (x,y) \in [a,b] \times [c,d], \tag{3.6}$$

converging in L^2, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$, and $\{u_j\}_{j\geq 1}$ and $\{v_j\}_{j\geq 1}$ are orthonormal sets of functions on $[a,b]$ and $[c,d]$, respectively.

Further assumptions on f are required to guarantee that the series in (3.6) converges in a stronger sense. For example, if f is Lipschitz continuous in both variables, then the series converges absolutely and uniformly to f.

Theorem 3.3 (Singular value decomposition of a function)**.** *Let $f : [a,b] \times [c,d] \to \mathbb{C}$ be Lipschitz continuous with respect to both variables. Then the SVD of f exists and the singular values are unique with $\sigma_j \to 0$ as $j \to \infty$. The singular vectors can be selected to be continuous functions and those corresponding to simple[2] singular values are unique up to complex signs. Moreover, the series in* (3.6) *is absolutely and uniformly convergent.*

Proof. The existence and uniqueness of the SVD is due to Schmidt [130], though he assumed that f was continuous and only showed that the series converges in the L^2-sense. Schmidt also showed that the singular vectors can be selected to be continuous. Hammerstein showed that (3.6) is uniformly convergent under an assumption that is implied by Lipschitz continuity [77], and Smithies showed absolute convergence under an assumption that is implied by Hölder continuity with exponent $> 1/2$ [135]. □

Thus, when f is a Lipschitz continuous function the infinite sum in (3.6) has three additional properties: (1) The series converges pointwise to f (since the series is uniformly convergent), (2) The series can be interchanged with integration (since the series is uniformly convergent), and (3) The series is unconditionally convergent,

[2]We say that a singular value σ_j is simple if $\sigma_i \neq \sigma_j$ for $1 \leq i \neq j$.

so is independent of the order of summation (since the sum is absolutely convergent). These properties will be important in Chapter 4.

The notion of the SVD of a function is a century old and preceded the matrix SVD. An excellent account of the early history is given by Stewart in [139].

3.9 Characterizations of the singular values

There are many different characterizations of the singular values of a function, each one corresponding to a characterization for the singular values of a matrix.

Theorem 3.4 (Characterizations of singular values)**.** *Let $f : [a,b] \times [c,d] \to \mathbb{C}$ be a function satisfying the assumptions in Theorem* 3.3*. The first singular value is:*

1. *The operator norm of f,*

$$\sigma_1 = \sup_{\substack{v \in L^2([a,b]) \\ \|v\|_{L^2}=1}} \left(\int_c^d \left(\int_a^b f(x,y)v(x)dx \right)^2 dy \right)^{1/2}.$$

2. *The maximum absolute value of a variational form,*

$$\sigma_1 = \sup_{\substack{u \in L^2([c,d]), v \in L^2([a,b]) \\ \|u\|_{L^2}=\|v\|_{L^2}=1}} \left| \int_c^d \int_a^b \overline{u(y)} f(x,y) v(x) dx dy \right|.$$

3. *The value of the maximum eigenvalue of a symmetric nonnegative definite kernel,*

$$\sigma_1 = \sup_{\substack{v \in L^2([c,d]) \\ \|v\|_{L^2}=1}} \left| \int_a^b \int_a^b \overline{v(y)} \left(\int_c^d \overline{f(y,s)} f(x,s) ds \right) v(x) dx dy \right|.$$

Proof. Let $v \in L^2([c,d])$ be of unit length $\|v\|_{L^2} = 1$ and let $\sum_{j=1}^{\infty} \sigma_j u_j(y) \overline{v_j(x)}$ be the SVD of $f(x,y)$. We can write $v = v_R + v_N$, where $\int_a^b f(x,\cdot) v_N(x) dx \equiv 0$ and $v_R = \sum_{j=1}^{\infty} \gamma_j v_j$ with[3] $\|\gamma\|_2 \leq 1$. Therefore, since $\{u_j\}_{j \geq 1}$ and $\{v_j\}_{j \geq 1}$ are orthonormal

[3]The singular vectors $\{v\}_{j \geq 1}$ form a complete orthonormal basis for the range of f [130].

sets we have

$$\left\| \int_c^d f(x,\cdot)v(x)dx \right\|^2_{L^2([c,d])} = \left\| \sum_{j=1}^{\infty} \sigma_j \gamma_j u_j \right\|^2_{L^2([c,d])} \leq \sum_{j=1}^{\infty} \sigma_j^2 \gamma_j^2 \leq \sigma_1^2,$$

and the inequalities are satisfied as equalities for $v = v_1$. Similar reasoning leads to the second and third characterization, i.e., bound the supremum and verify that the first singular vector(s) attains the resulting bound. □

The other singular values can be characterized recursively, since for $k \geq 1$

$$\sigma_k(f) = \sigma_1(f - f_{k-1}), \qquad f_{k-1}(x,y) = \sum_{j=1}^{k-1} \sigma_j(f) u_j(y) \overline{v_j(x)},$$

where $\sigma_k(f)$ denotes the kth singular value of f.

3.10 Best low rank function approximation

In 1912 Weyl showed that the first k terms of the SVD of a symmetric function give a best rank k approximant in the L^2-norm [166]. Here, we show that the same result and proof work for nonsymmetric functions (an alternative proof was given by Schmidt [130]). The first step in the proof shows that the singular values of a function are not too sensitive to low rank perturbations.

Lemma 3.5. *Let $f : [a,b] \times [c,d] \to \mathbb{C}$ be a function satisfying the assumptions of Theorem 3.3 and g_k a continuous function of rank at most $k \geq 1$. Then, $\sigma_1(f + g_k) \geq \sigma_{k+1}(f)$.*

Proof. Let $f(x,y) = \sum_{j=1}^{\infty} \sigma_j u_j(y) \overline{v_j(x)}$ be the SVD of f and $g_k(x,y) = \sum_{j=1}^{k} c_j(y) r_j(x)$ for functions c_j and r_j. Since g_k is of rank at most k there exists $\gamma_1, \ldots, \gamma_{k+1}$ such that

$$\int_a^b g_k(x,y)v(x)dx \equiv 0, \qquad v = \sum_{j=1}^{k+1} \gamma_j v_j.$$

Without loss of generality we assume that $\gamma_1^2 + \cdots + \gamma_{k+1}^2 = 1$. Then, by Theorem 3.4

$$\begin{aligned}\sigma_1^2(f+g_k) &\geq \int_a^b \int_a^b \overline{v(y)} \left(\int_c^d \overline{(f(y,s)+g_k(y,s))}(f(x,s)+g_k(x,s))ds \right) v(x)dxdy \\ &= \int_a^b \int_a^b \overline{v(y)} \left(\int_c^d \overline{f(y,s)} f(x,s)ds \right) v(x)dxdy = \sum_{j=1}^{k+1} \gamma_j^2 \sigma_j^2(f) \geq \sigma_{k+1}^2(f).\end{aligned}$$

Taking the square root of both sides gives the result. □

The preceding lemma can also be shown for L^2-integrable functions if certain equalities are understood in the L^2-sense.

The second step in the proof shows how the singular values behave under addition.

Lemma 3.6. *Let $f, g : [a,b] \times [c,d] \to \mathbb{C}$ be functions satisfying the assumptions in Theorem 3.3, and let $h = f + g$. Then, for $i, j \geq 1$ we have $\sigma_{i+j-1}(h) \leq \sigma_i(f) + \sigma_j(g)$.*

Proof. Let $h(x,y) = \sum_{j=1}^{\infty} \sigma_j(h) u_j(y) \overline{v_j(x)}$ be the SVD of h. By Theorem 3.4

$$\sigma_1(h) = \int_c^d \int_a^b \overline{u_1(y)}(f(x,y)+g(x,y))v_1(x)dxdy \leq \sigma_1(f) + \sigma_1(g),$$

which shows the lemma for $i = j = 1$. More generally, let f_{i-1} and g_{j-1} be the first $i-1$ and $j-1$ terms of the SVD of f and g, respectively. Then, $\sigma_1(f - f_{i-1}) = \sigma_i(f)$ and $\sigma_1(g - g_{j-1}) = \sigma_j(g)$. Furthermore, $f_{i-1} + g_{j-1}$ is of rank at most $i+j-2$. Hence, we have by Lemma 3.5

$$\sigma_i(f) + \sigma_j(g) = \sigma_1(f - f_{i-1}) + \sigma_1(g - g_{j-1}) \geq \sigma_1(h - f_{i-1} - g_{j-1}) \geq \sigma_{i+j-1}(h).$$

□

Again, the preceding lemma also holds for L^2-integrable functions.

Finally, the previous two lemmas can be combined to prove that the first k singular values and corresponding vectors of the SVD of a function give a best rank k approximant to f in the L^2-norm.

Theorem 3.5 (Best low rank approximation of a function)**.** *Let $f : [a,b] \times [c,d] \to \mathbb{C}$ be a function satisfying the assumptions in Theorem 3.3. Then, the sum of the first*

k terms of the SVD of f is a best rank k approximant to f in the L^2-norm, i.e.,

$$\|f - f_k\|_{L^2} = \inf_{\substack{g_k \in \mathcal{C}([a,b]\times[c,d]) \\ \operatorname{rank}(g_k) \le k}} \|f - g_k\|_{L^2}, \qquad f_k(x,y) = \sum_{j=1}^{k} \sigma_j(f) u_j(y) \overline{v_j(x)}.$$

Proof. Let g_k be a continuous function on $[a,b] \times [c,d]$ of rank k. By Lemma 3.6 we have $\sigma_j(f - g_k) \ge \sigma_{k+j}(f)$ for $j \ge 1$ and hence,

$$\|f - g_k\|_{L^2}^2 = \sum_{j=1}^{\infty} \sigma_j(f - g_k)^2 \ge \sum_{j=k+1}^{\infty} \sigma_j(f)^2.$$

Equality is attained when g_k is the sum of the first k terms of the SVD of f. □

In linear algebra, it is the connection between the SVD and best rank approximation that has made the SVD such a powerful tool [139].

The SVD of a function can be made into a practical algorithm by discretizing functions on sufficiently fine tensor product grids and computing the matrix SVD [152, Sec. 2]. However, we have found the practical algorithm derived from GE with complete pivoting (see Section 2.1.1) to be far more efficient for constructing near-optimal low rank approximations to smooth functions (see Section 2.1.2).

Chapter 4

Continuous analogues of matrix factorizations*

A fundamental idea in linear algebra is matrix factorization, where matrices are decomposed into a product of, for example, triangular, diagonal, or orthogonal matrices. Such factorizations provide a tool for investigating properties of matrices as well as analyzing and performing matrix computations [67, p. 607], and these ideas are ubiquitous throughout numerical linear algebra [61, 81, 85, 160].

In this chapter we derive continuous analogues of matrix factorizations, where matrices become quasimatrices and cmatrices (see Section 4.1). The factorizations we consider come in three main flavors: (1) two-sided orthogonal ($A = U\Sigma V^*$), (2) one-sided orthogonal ($A = QR$), and (3) non-orthogonal ($A = LU$ and $A = R^*R$). In each case we describe these factorizations for quasimatrices and cmatrices, prove existence and uniqueness theorems, and state basic properties of the factorizations.

4.1 Matrices, quasimatrices, and cmatrices

An $m \times n$ matrix is an array of mn numbers where the entry in row i, column j is denoted by $A(i, j)$. An $[a, b] \times n$ quasimatrix $\mathcal{A}$ is a "matrix" for which the row index is a continuous variable and the n columns are functions defined on $[a, b]$ (see Section 1.7). The row $y \in [a, b]$, column j entry is denoted by $\mathcal{A}(y, j)$. An $[a, b] \times [c, d]$ *cmatrix*, also denoted by $\mathcal{A}$, is a "matrix" for which both the row and column indices are continuous variables. It is a bivariate function with the indexing variables interchanged from the usual x, y ordering to make y the column (vertical) variable

*This chapter is based on a paper with Nick Trefethen [153]. I proposed the key ideas of triangularity, the meaning of the main factorizations, and proved the convergence theorems. Trefethen made the definition of a triangular quasimatrix precise, greatly simplified the proof of Theorem 4.6, and was the lead author of the paper; however, this chapter is quite a bit different from the paper.

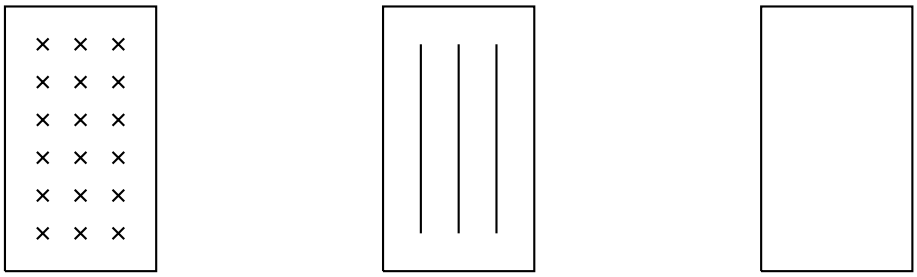


Figure 4.1: A rectangular matrix (left), quasimatrix (center), and a rectangular cmatrix (right). Matrices are drawn as boxes with crosses, quasimatrices as boxes with vertical lines, and cmatrices as empty rectangles. Row quasimatrices are drawn as boxes with horizontal lines and square cmatrices as empty squares.

and x the row (horizontal) variable for consistency with the algebra of matrices. The row $y \in [a, b]$, column $x \in [c, d]$ entry is denoted by $\mathcal{A}(y, x)$, and likewise $\mathcal{A}(\cdot, x)$ and $\mathcal{A}(y, \cdot)$ are the xth column and yth row of $\mathcal{A}$, respectively (continuous analogues of column and row vectors).

Figure 4.1 shows a rectangular matrix, a quasimatrix, and a rectangular cmatrix. A *rectangular cmatrix* is defined on $[a, b] \times [c, d]$, and is *square*[1] if $a = c$ and $b = d$.

Throughout this chapter we consider *continuous quasimatrices*, i.e., quasimatrices with continuous columns and *continuous cmatrices*, which are cmatrices that are continuous in both variables.

4.2 Matrix factorizations as rank one sums

Every matrix can be written as a sum of rank 1 matrices, and a decomposition of a matrix A into a matrix-matrix product, $A = BC$, provides such an expression. For example, if $A = BC$ with $B \in \mathbb{C}^{m \times k}$ and $C \in \mathbb{C}^{k \times n}$, then

$$A = \sum_{j=1}^{k} b_j c_j^*,$$

where b_j and c_j^* are the jth column of B and jth row of C, respectively. In this way, any matrix factorization can be interpreted as a finite sum of rank 0 or 1 matrices, and if $A \in \mathbb{C}^{m \times n}$ with $m \geq n$, then the SVD, QR, and LU factorizations are the

[1]Note that an $[a, b] \times [c, d]$ cmatrix with $b - a = d - c$ is not necessarily a square cmatrix.

following sums involving rank 0 or 1 terms, respectively:

$$A = \sum_{j=1}^{n} \sigma_j u_j v_j^*, \qquad A = \sum_{j=1}^{n} q_j r_j^*, \qquad A = \sum_{j=1}^{n} \ell_j u_j^*. \tag{4.1}$$

One way to generalize matrix factorizations to quasimatrices and cmatrices is to replace the column or row vectors in (4.1) with functions. If $\mathcal{A}$ is an $[a,b] \times n$ quasimatrix, then the SVD, QR, and LU factorizations become

$$\mathcal{A} = \sum_{j=1}^{n} \sigma_j u_j(y) v_j^*, \qquad \mathcal{A} = \sum_{j=1}^{n} q_j(y) r_j^*, \qquad \mathcal{A} = \sum_{j=1}^{n} \ell_j(y) u_j^*,$$

where each column vector is now a function of y (the vertical variable). Moreover, if $\mathcal{A}$ is an $[a,b] \times [c,d]$ cmatrix then (4.1) becomes formally the infinite series

$$\mathcal{A} = \sum_{j=1}^{\infty} \sigma_j u_j(y) v_j^*(x), \qquad \mathcal{A} = \sum_{j=1}^{\infty} q_j(y) r_j^*(x), \qquad \mathcal{A} = \sum_{j=1}^{\infty} \ell_j(y) u_j^*(x), \tag{4.2}$$

where column vectors are now functions of y and row vectors are functions of x.

Every factorization can be viewed either as a decomposition, e.g. $A = LU$, or as a sum involving rank 0 or 1 terms. In the matrix (and quasimatrix) setting these two viewpoints are mathematically equivalent, with the decomposition interpretation far more popular and convenient. However, for cmatrices they are at least formally different due to convergence issues, since each series in (4.2) contains potentially infinitely many nonzero terms. To be able to write the convenient cmatrix decompositions $\mathcal{A} = \mathcal{U}\Sigma\mathcal{V}^*$, $\mathcal{A} = \mathcal{Q}\mathcal{R}$, and $\mathcal{A} = \mathcal{L}\mathcal{U}$ we require that the series in (4.2) be unconditionally convergent, since in the decomposition interpretation the underlying rank 1 terms could be summed up in any order. Therefore, for cmatrices we will seek to find assumptions on $\mathcal{A}$ that ensure the series in (4.2) are absolutely convergent[2] so that the two viewpoints are equivalent. In addition, we will require the series to converge pointwise to $\mathcal{A}$ and to ensure this we demand that the series converges uniformly to $\mathcal{A}$.

[2]An absolutely convergent series is unconditionally convergent.

Object	Factorization	Pivoting strategy	Chebfun command
quasimatrix	SVD	None	`svd`
	QR	None	`qr`
	LU	Partial	`lu`
cmatrix	SVD	None	`svd`
	QR	Column	`qr`
	LU	Complete	`lu`
	Cholesky	Diagonal	`chol`

Table 4.1: Summary of the pivoting strategies we employ, and the corresponding Chebfun commands. The quasimatrix Cholesky factorization does not exist because a quasimatrix cannot be square (see Section 1.7).

4.3 The role of pivoting in continuous linear algebra

Algorithms for matrix factorizations can be stated and employed without pivoting, and are often simple to describe and analyze. Pivoting is usually regarded as an optional extra for matrices, making the algorithm perform more favorably, for example, QR with column pivoting for least squares problems [59]. An exception to this is GE, where partial pivoting is commonly used to prevent the algorithm failing on matrices with singular or near-singular leading principal minors.

For quasimatrices and cmatrices the situation is quite different as pivoting is sometimes essential to even formulate the factorizations and to describe the corresponding algorithms. To see why, consider GE without pivoting on an $[a, b] \times n$ continuous quasimatrix $\mathcal{A}$. The first step of GE causes no problems as we can conveniently regard the first row as $\mathcal{A}(a, \cdot)$ and use it to introduce zeros in column 1, but what is the second row of $\mathcal{A}$? There is no natural "second" row,[3] so GE without pivoting on a quasimatrix is meaningless.

To overcome this fundamental issue we demand that GE pivots in the vertical variable to select that "second" (and "third") row. For example, GE with partial pivoting (row pivoting) on an $[a, b] \times n$ quasimatrix can be defined without ambiguity (see Section 4.7).

Similarly, to define GE on a cmatrix a pivoting strategy needs to select both the next row and column. We use complete pivoting for this, where at each step the pivot is the location of the maximum absolute value of the cmatrix (see Section 4.10).

[3]Fundamentally, this is because of the order structure of $\mathbb{R}$. The reals have no successor function under the standard ordering.

For cmatrices pivoting is also a sorting mechanism. All the cmatrix factorizations are potentially infinite series and the pivoting strategy needs to approximately order the rank 1 terms, with the most significant first, to ensure the series is convergent in any sense. Table 4.1 summarizes the pivoting strategies that we use for each factorization.

4.4 Psychologically triangular matrices and quasimatrices

For a matrix A, the most familiar way of describing partial pivoting in GE is as an interchange of certain rows that leads to the factorization $PA = LU$, where P is a permutation matrix, L is a unit lower-triangular matrix and U is an upper-triangular matrix [160, p.159–160]. However, an equivalent formulation is to regard pivoting as a *selection* of rows without physically interchanging them. This leads to a factorization without a permutation matrix, $A = LU$, where L is now a *psychologically lower-triangular* matrix.[4]

Definition 4.1. *An $m \times n$, $m \geq n$, matrix L is psychologically lower-triangular if there is a permutation $\{i_1, \ldots, i_n\}$ of $\{1, \ldots, n\}$ such that column j has zero entries at rows $i_1, \ldots, i_{j-1}$. A matrix is psychologically upper-triangular if its transpose is psychologically lower-triangular.*

Figure 4.2 shows a lower-triangular matrix (left) and an example of a psychologically lower-triangular matrix (center). Psychologically lower-triangular matrices are constructed by the `lu` command in MATLAB when executed with two output arguments, and the specific permutation of $\{1, \ldots, n\}$ is returned when an optional flag is supplied.

For quasimatrices and cmatrices we use pivoting as a way of selecting certain rows and columns without interchange and hence, our factorizations involve psychologically triangular quasimatrices (a continuous analogue of Definition 4.1).

Definition 4.2. *An $[a, b] \times n$ quasimatrix is lower-triangular (we drop the term "psychologically") if there is a set of distinct values $y_1, \ldots, y_n \in [a, b]$ such that column j has zeros at $y_1, \ldots, y_{j-1}$. Similarly, an $n \times [a, b]$ quasimatrix is upper-triangular if there is a set of distinct values $x_1, \ldots, x_n \in [a, b]$ such that row i has zeros at $x_1, \ldots, x_{i-1}$ (or, equivalently, if its transpose is lower-triangular).*

[4] The phrase "psychologically triangular" is used almost exclusively in the documentation for the `lu` command in MATLAB and was probably coined by Nick Trefethen years ago.

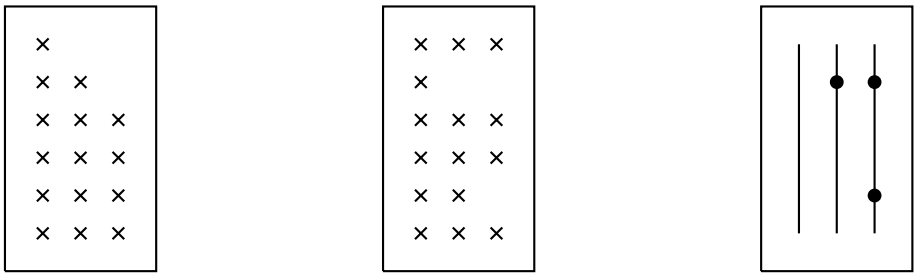


Figure 4.2: A lower-triangular matrix (left), a psychologically lower-triangular matrix (center), and a lower-triangular quasimatrix (right). On the right, the numbers in brackets indicate the numbers of nested zeros in each column and the black dots are possible locations of those zeros. For subsequent diagrams involving triangular quasimatrices we do not draw the black dots.

Triangular quasimatrices arise in continuous analogues of GE, the Cholesky algorithm, and Gram–Schmidt orthogonalization on cmatrices, and there are analogues of forward and back substitution for solving linear systems that involve triangular quasimatrices [153, Sec. 8]. Figure 4.2 (right) shows a lower-triangular quasimatrix, where the numbers in parentheses indicate the numbers of nested zeros in the columns.

Many of the standard definitions for triangular matrices generalize to triangular quasimatrices.

Definition 4.3. *If $\mathcal{L}$ is an $[a,b] \times n$ lower-triangular quasimatrix with respect to $y_1, \ldots, y_n \in [a,b]$, then the diagonal of $\mathcal{L}$ is the set of values $\ell_1(y_1), \ldots, \ell_n(y_n)$, where ℓ_j is the jth column of $\mathcal{L}$. If the diagonal values are all 1, $\mathcal{L}$ is unit lower-triangular. If each diagonal entry is (strictly) maximal, so that $|\ell_j(y)| \le |\ell_j(y_j)|$ (resp. $|\ell_j(y)| < |\ell_j(y_j)|$) for all j and $y \in [a,b]$, $y \ne y_j$, then we say that $\mathcal{L}$ is (strictly) diagonally maximal. If $\mathcal{L}$ is (strictly) diagonally maximal and its diagonal values are real and nonnegative, it is (strictly) diagonally real maximal. Analogous definitions also hold for $n \times [a,b]$ upper-triangular quasimatrices.*

4.5 The SVD of a quasimatrix

The first factorization we consider is the SVD of a quasimatrix, which appeared in [10, Sec. 3.5] and is defined as follows. (Orthogonality of functions is defined with respect to the standard L^2 inner product.)

Definition 4.4. *Let $\mathcal{A}$ be an $[a,b] \times n$ continuous quasimatrix. An SVD of $\mathcal{A}$ is a factorization $\mathcal{A} = \mathcal{U}\Sigma V^*$, where $\mathcal{U}$ is an $[a,b] \times n$ continuous quasimatrix with orthonormal columns, Σ is an $n \times n$ diagonal matrix with entries $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$, and V is an $n \times n$ unitary matrix.*

This definition can be extended to quasimatrices with columns in $L^2([a,b])$ if the equality in $\mathcal{A} = \mathcal{U}\Sigma V^*$ is understood in the L^2 sense. Basic properties of the quasimatrix SVD are discussed in [9, Sec. 3.5] under the assumption that $\mathcal{A}$ is of full rank; however, the full rank assumption is unnecessary.

Theorem 4.1 (SVD of a quasimatrix)**.** *Every $[a,b] \times n$ continuous quasimatrix $\mathcal{A}$ has an SVD, $\mathcal{A} = \mathcal{U}\Sigma V^*$. The singular values are uniquely defined. The singular vectors corresponding to simple singular values are unique up to complex signs. The rank r of $\mathcal{A}$ is the number of nonzero singular values. The first r columns of $\mathcal{U}$ form an orthonormal basis for the range of $\mathcal{A}$ when regarded as a map from $\mathbb{C}^n$ to $\mathcal{C}([a,b])$, and the last $n - r + 1$ columns of V form an orthonormal basis for the null space.*

Proof. If $\mathcal{A}$ is of full rank then existence[5] was shown in [9, Thm 3.5.1]. If $\mathcal{A}$ is not of full rank, Theorem 4.4 shows that $\mathcal{A}$ has an LU factorization $\mathcal{A} = \mathcal{L}_A U_A$, where $\mathcal{L}_A$ is a unit lower-triangular continuous quasimatrix and U_A is an upper-triangular matrix. Since $\mathcal{L}_A$ is unit lower-triangular it is a quasimatrix of full rank and by [9, Thm 3.5.1] $\mathcal{L}_A$ has an SVD, $\mathcal{L}_A = \mathcal{U}_L \Sigma_L V_L^*$. The existence of the SVD for $\mathcal{A}$ follows since $\mathcal{A} = \mathcal{U}_L(\Sigma_L V_L^* U_A) = (\mathcal{U}_L U)\Sigma V^*$, where $\Sigma_L V_L^* U_A = U\Sigma V^*$ is an SVD of $\Sigma_L V_L^* U_A$. Thus, we have constructed an SVD of $\mathcal{A}$. For uniqueness suppose that $\mathcal{A} = \mathcal{U}_1\Sigma_1 V_1^* = \mathcal{U}_2\Sigma_2 V_2^*$. Then, $A^*A = V_1\Sigma_1^*\Sigma_1 V_1^* = V_2\Sigma_2^*\Sigma_2 V_2^*$ and, by uniqueness of the matrix SVD, $\Sigma_1 = \Sigma_2$ and the singular vectors in V_1 and V_2 that correspond to simple singular values are unique up to complex signs. Moreover, since $\Sigma_1 = \Sigma_2$ is a diagonal matrix, the columns of $\mathcal{U}_1$ and $\mathcal{U}_2$ that correspond to simple singular values are also unique up to complex signs [160, Thm. 4.1].

The rank of $\mathcal{A}$ is equal to the rank of Σ (the number of nonzero singular values) as $\mathcal{U}$ and V are of full rank. The range of $\mathcal{A}$ is the space spanned by the columns of $\mathcal{U}$ corresponding to nonzero singular values, and the null space is the space spanned by the columns of V corresponding to zero singular values. □

[5] Though this property is not stated explicitly, the proof in [9, Thm 3.5.1] also shows that the columns of $\mathcal{U}$ can be chosen to be continuous.

The first k terms of the SVD of an $[a,b] \times n$ quasimatrix $\mathcal{A}$ give a best rank k approximant with respect to the quasimatrix norms $\|\cdot\|_2$ and $\|\cdot\|_F$, where

$$\|\mathcal{A}\|_2 = \sup_{v \in \mathbb{C}^n \setminus 0} \frac{\|\mathcal{A}v\|_{L^2([a,b])}}{\|v\|_2}, \qquad \|\mathcal{A}\|_F^2 = \sum_{j=1}^{n} \|\mathcal{A}(\cdot, j)\|_{L^2([a,b])}^2 . \tag{4.3}$$

We summarize this statement as a theorem.

Theorem 4.2 (Best rank quasimatrix approximation)**.** *Let $\mathcal{A}$ be an $[a,b] \times n$ continuous quasimatrix and $\|\cdot\|_F$ the quasimatrix norm in* (4.3)*. Then, the first k terms of the SVD for $\mathcal{A}$ give a best rank k approximant to $\mathcal{A}$ with respect to the quasimatrix norms $\|\cdot\|_2$ and $\|\cdot\|_F$.*

Proof. The reasoning is analogous to that in Section 3.10, but for quasimatrices instead of functions. □

The ability to take the SVD of a quasimatrix has been available in Chebfun since the very beginning [10] via the `svd` command, and the underlying algorithm is based on the QR factorization (as described in Section 1.7). From this factorization, we can readily define notions such as the Moore–Penrose pseudoinverse $\mathcal{A}^+ = V\Sigma^+\mathcal{U}^*$ and the condition number $\kappa(\mathcal{A}) = \kappa(\Sigma)$.

4.6 The QR factorization of a quasimatrix

The QR factorization of an $[a,b] \times n$ quasimatrix is analogous to the QR of a tall-skinny matrix with no surprises.

Definition 4.5. *Let $\mathcal{A}$ be an $[a,b] \times n$ continuous quasimatrix. A QR factorization of $\mathcal{A}$ is a decomposition $\mathcal{A} = \mathcal{Q}R$, where $\mathcal{Q}$ is an $[a,b] \times n$ continuous quasimatrix with orthonormal columns and R is an $n \times n$ upper-triangular matrix.*

The QR factorization of a quasimatrix was mentioned in [160, pp. 52–54], and appeared around the same time in [140]. It was one of the original capabilities of Chebfun [9, 10]. The algorithm originally relied on Gram–Schmidt orthogonalization and was later replaced by a more stable algorithm based on a continuous analogue of Householder triangularization [157].

The basic properties of the factorization are summarized by the following theorem.

Theorem 4.3 (QR factorization of a quasimatrix)**.** *Every $[a,b]\times n$ continuous quasimatrix $\mathcal{A}$ has a QR factorization. If the columns of $\mathcal{A}$ are linearly independent continuous functions and the diagonal entries of R are required to be nonnegative and real, then the QR factorization is unique. For each k with $1 \le k \le n$, the space spanned by the columns $a_1(y),\dots,a_k(y)$ of $\mathcal{A}$ is a subspace of the span of the columns $q_1(y),\dots,q_k(y)$ of $\mathcal{Q}$.*

Proof. If $\mathcal{A}$ is of full rank then existence and uniqueness were shown in [9, Sec. 3.4] (an alternative proof is given in [140, p. 38]). If $\mathcal{A}$ is not of full rank, Theorem 4.4 shows that $\mathcal{A}$ has an LU factorization $\mathcal{A} = \mathcal{L}_A\mathcal{U}_A$, where $\mathcal{L}_A$ is a unit lower-triangular continuous quasimatrix and U_A is an upper-triangular matrix. Since $\mathcal{L}_A$ is of full rank, $\mathcal{L}_A$ has a QR factorization, $\mathcal{L}_A = \mathcal{Q}_A R_A$ [9, Sec. 3.4]. The existence of the quasimatrix QR factorization follows since $\mathcal{A} = \mathcal{Q}_A(R_A U_A)$ is a QR factorization of $\mathcal{A}$.

For the last statement, since $\mathcal{A} = \sum_{j=1}^{n} q_j(y) r_j^*$ and R is an upper-triangular matrix the columns $a_1(y),\dots,a_k(y)$ of $\mathcal{A}$ can be expressed as linear combinations of $q_1(y),\dots,q_k(y)$. □

If the columns of $\mathcal{A}$ are linearly dependent then one can find an arbitrary continuous function that is orthogonal to the preceding columns of $\mathcal{Q}$, making the factorization nonunique.

One can also define the QR factorization of an $n\times[a,b]$ continuous quasimatrix, involving an $n\times n$ unitary matrix Q and an $n\times[a,b]$ upper-triangular continuous quasimatrix $\mathcal{R}$. For the reasons discussed in Section 4.3 this factorization requires a pivoting strategy in the x-variable.

4.7 The LU factorization of a quasimatrix

We now come to our first new factorization: the LU factorization of a quasimatrix.

Definition 4.6. *Let $\mathcal{A}$ be an $[a,b]\times n$ continuous quasimatrix. An LU factorization of $\mathcal{A}$ is a factorization $\mathcal{A} = \mathcal{L}U$, where U is an upper-triangular $n\times n$ matrix and $\mathcal{L}$ is an $[a,b]\times n$ unit lower-triangular continuous quasimatrix.*

Figure 4.3 shows the factorization, where the numbers in parentheses indicate the numbers of nested zeros in each column. (This diagram does not show that $\mathcal{L}$ is unit lower-triangular.)

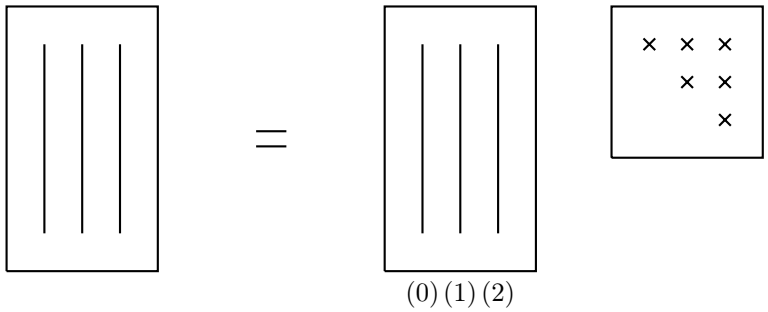


Figure 4.3: The LU factorization of an $[a,b] \times n$ quasimatrix: $\mathcal{A} = \mathcal{L}U$, where $\mathcal{L}$ is an $[a,b] \times n$ unit lower-triangular continuous quasimatrix and U is an $n \times n$ upper-triangular matrix. The numbers in parentheses indicate the number of nested zeros in each column.

An algorithm for computing the LU factorization involves pivoting in the vertical direction (see Section 4.3), and we use GE with partial pivoting. Let $\mathcal{A}$ be an $[a,b] \times n$ continuous quasimatrix. GE with partial pivoting on $\mathcal{A}$ is a direct algorithm requiring n steps. Define $\mathcal{E}_0 = \mathcal{A}$. At step $k = 1$, find a value $y_1 \in [a,b]$ for which $|\mathcal{E}_0(\cdot, 1)|$ is maximal and define $\ell_1 = \mathcal{E}_0(\cdot,1)/\mathcal{E}_0(y_1,1)$, $u_1^* = \mathcal{E}_0(y_1,\cdot)$, and $\mathcal{E}_1 = \mathcal{E}_0 - \ell_1(y)u_1^*$. (If $\mathcal{E}_0(y_1,1) = 0$, ℓ_1 can be any function in $\mathcal{C}([a,b])$ with $|\ell_1(y)| \le 1$ for all y and $\ell_1(y_1) = 1$.) The new quasimatrix $\mathcal{E}_1$ is zero in row y_1 and column 1. At step $k = 2$, find a value $y_2 \in [a,b]$ for which $|\mathcal{E}_1(\cdot,2)|$ is maximal and define $\ell_2 = \mathcal{E}_1(\cdot,2)/\mathcal{E}_1(y_2,2)$, $u_2^* = \mathcal{E}_1(y_2,\cdot)$, and $\mathcal{E}_2 = \mathcal{E}_1 - \ell_2(y)u_2^*$. (If $\mathcal{E}_1(y_2,2) = 0$, ℓ_2 can be any function in $\mathcal{C}([a,b])$ with $|\ell_2(y)| \le 1$ for all y, $\ell_2(y_1) = 0$, and $\ell_2(y_2) = 1$.) The quasimatrix $\mathcal{E}_2$ is zero in rows y_1 and y_2 and columns 1 and 2. Continuing in this fashion, after n steps we have the zero quasimatrix $\mathcal{E}_n$, and hence $\mathcal{A} = \sum_{j=1}^{n} \ell_j(y)u_j^*$ or, equivalently, $\mathcal{A} = \mathcal{L}U$. Due to the pivoting strategy $\mathcal{L}$ is diagonally maximal in addition to being a unit lower-triangular continuous quasimatrix (see Section 4.3).

The basic properties of this factorization are summarized in the following theorem.

Theorem 4.4 (LU factorization of a quasimatrix)**.** *Every $[a,b] \times n$ continuous quasimatrix $\mathcal{A}$ has an LU factorization $\mathcal{A} = \mathcal{L}U$. If $\mathcal{A}$ is of full rank and $\mathcal{L}$ is strictly diagonally maximal, then the factorization is unique. The rank of $\mathcal{A}$ is equal to the rank of U. For each $1 \le k \le n$, the space spanned by the columns $a_1(y), \ldots, a_k(y)$ of $\mathcal{A}$ is a subspace of the span of the columns $\ell_1(y), \ldots, \ell_k(y)$ of $\mathcal{L}$.*

Proof. Existence is evident from the GE algorithm. For uniqueness, we proceed by induction. For $n = 1$, if there are two factorizations then we have $\ell_1^{(1)} u_{11}^{(1)} = \ell_1^{(2)} u_{11}^{(2)}$, and since $\mathcal{A}$ is of full rank $u_{11}^{(1)} \neq 0$ and $\ell_1^{(1)} = (u_{11}^{(2)}/u_{11}^{(1)})\ell_1^{(2)}$. In addition, $\ell_1^{(1)}$ and $\ell_1^{(2)}$ are strictly diagonally maximal so take the value of 1 at the same point, $y_1^{(1)} = y_2^{(2)}$,

so $u_{11}^{(1)} = u_{11}^{(2)}$. Moreover, $\ell_1^{(1)} = \ell_1^{(2)}$ as $\ell_1^{(1)}$ and $\ell_1^{(2)}$ are continuous. For general n and a full rank $\mathcal{A}$, we suppose that the first $n-1$ columns of $\mathcal{A}$ have a unique LU factorization. Then, we have the following expression for the last column of $\mathcal{A}$:

$$u_{1n}^{(1)}\ell_1 + \cdots + u_{n-1n}^{(1)}\ell_{n-1} + u_{nn}^{(1)}\ell_n^{(1)} = u_{1n}^{(2)}\ell_1 + \cdots + u_{n-1n}^{(2)}\ell_{n-1} + u_{nn}^{(2)}\ell_n^{(2)}.$$

By the nested set of zeros in the first $n-1$ columns we conclude $u_{jn}^{(1)} = u_{jn}^{(2)}$ for $1 \le j \le n-1$, and hence $u_{nn}^{(1)}\ell_n^{(1)} = u_{nn}^{(2)}\ell_n^{(2)}$. Finally, $u_{nn}^{(1)} = u_{nn}^{(2)}$ and $\ell_n^{(1)} = \ell_n^{(2)}$ by the same reasoning as in the $n = 1$ case. The result follows by induction on n.

The unit lower-triangular quasimatrix $\mathcal{L}$ is of full rank (its diagonal entries are nonzero) and hence, $\operatorname{rank}(\mathcal{A}) = \operatorname{rank}(U)$. For the last statement, since we have $\mathcal{A} = \sum_{j=1}^{n} \ell_j(y)u_j^*$ and U is an upper-triangular matrix, the columns $a_1(y), \ldots, a_k(y)$ of $\mathcal{A}$ can be expressed as a linear combination of $\ell_1(y), \ldots, \ell_k(y)$. □

The `lu` command in Chebfun computes an LU factorization of a quasimatrix by using GE with partial pivoting; however, an LU factorization can be constructed using other row pivoting strategies in GE. This leads to lower-triangular quasimatrices that are not diagonally maximal. Furthermore, one could introduce column pivoting, in which case U would become a psychologically upper-triangular matrix (see Section 4.3).

One can also define the LU factorization of an $n \times [a,b]$ quasimatrix $\mathcal{A}$, requiring column pivoting rather than row pivoting. This yields $\mathcal{A} = L\mathcal{U}$ where L is an $n \times n$ unit lower-triangular matrix and $\mathcal{U}$ is an $n \times [a,b]$ upper-triangular continuous quasimatrix. If partial pivoting (in the column direction) is used then $\mathcal{U}$ is diagonally maximal.

4.8 The SVD of a cmatrix

We now come to our first cmatrix factorization and accordingly, we must now pay attention to the convergence of the infinite series in (4.2). We have already seen the SVD in Chapter 3, described as a factorization of functions instead of cmatrices. The following cmatrix definition is analogous to the SVD of a continuous function (see Definition 3.2).

Definition 4.7. *Let $\mathcal{A}$ be an $[a,b] \times [c,d]$ continuous cmatrix. An SVD of $\mathcal{A}$ is an*

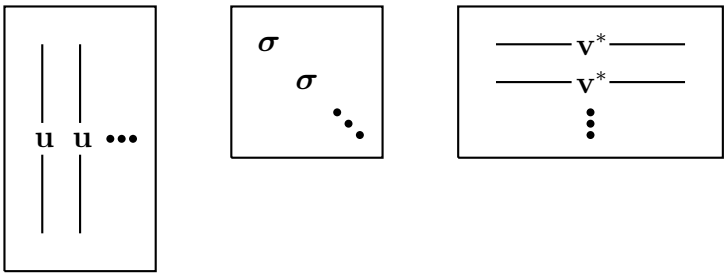


Figure 4.4: Another way to visualize the SVD of a Lipschitz continuous cmatrix. If $\mathcal{A}$ is Lipschitz continuous, then the infinite sum in (3.6) is absolutely convergent and therefore, can be written as a product of quasimatrices and an infinite diagonal matrix (see Corollary 4.1).

infinite series,

$$\mathcal{A} = \sum_{j=1}^{\infty} \sigma_j u_j(y) v_j^*(x), \qquad (x, y) \in [a, b] \times [c, d], \tag{4.4}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$*, and* $\{u_j\}_{j\geq 1}$ *and* $\{v_j\}_{j\geq 1}$ *are orthonormal sets of continuous functions on* $[a, b]$ *and* $[c, d]$*, respectively. The equality in* (4.4) *should be understood to signify that the series converges to* $\mathcal{A}$ *pointwise.*

Theorem 3.3 states that if $\mathcal{A}$ is Lipschitz continuous in both of its variables, then the SVD series is uniformly and absolutely convergent to $\mathcal{A}$. In this situation the series is unconditionally convergent so we can write it as $\mathcal{A} = \mathcal{U}\Sigma\mathcal{V}^*$ (see Section 4.3). Figure 4.4 shows the SVD of a cmatrix as a decomposition.

Corollary 4.1. *If* $\mathcal{A}$ *is an* $[a, b]\times[c, d]$ *cmatrix that is Lipschitz continuous with respect to both its variables, then an SVD of* $\mathcal{A}$ *can be written as* $\mathcal{A} = \mathcal{U}\Sigma\mathcal{V}^*$*, where* $\mathcal{U}$ *is an* $[a, b] \times \infty$ *continuous quasimatrix and* $\mathcal{V}$ *is an* $\infty \times [c, d]$ *continuous quasimatrix, both with orthonormal columns, and* Σ *is an infinite diagonal matrix with real nonnegative non-increasing entries.*

Proof. This is equivalent to the statement in Theorem 3.3. □

The underlying mathematics of the SVD of a cmatrix goes back to work by Schmidt in 1907 [130] and Weyl in 1912 [166]. An account of the history can be found in [142].

The `svd` command in Chebfun2 is a valuable feature for appreciating the near-optimality of the iterative GE algorithm (see Section 2.1.2). Figure 4.5 shows the algorithm we employ in the `svd` command in Chebfun2, which first calculates the LU factorization of a cmatrix (see Section 4.10) before computing a QR factorization of a

Pseudocode: SVD of a cmatrix

Input: An $[a,b]\times[c,d]$ cmatrix $\mathcal{A}$.

Output: Quasimatrices with orthonormal columns $\mathcal{U}$ and $\mathcal{V}$, and an infinite diagonal matrix Σ such that $\mathcal{A}=\mathcal{U}\Sigma\mathcal{V}^*$.

1. Cmatrix LU factorization: $\mathcal{A}=\mathcal{L}_A\mathcal{U}_A$ (see Section 4.10)
2. Quasimatrix QR factorization: $\mathcal{L}_A=\mathcal{Q}_L R_L$ and $\mathcal{U}_A^*=\mathcal{Q}_U R_U$
3. Matrix-matrix product: $B=R_L R_U^*$
4. Matrix SVD: $B=U_B\Sigma_B V_B^*$
5. Construct: $\mathcal{U}_A=\mathcal{Q}_L U_B$, $\mathcal{V}_A=\mathcal{Q}_R V_B$, and $\Sigma_A=\Sigma_B$

Figure 4.5: Pseudocode for computing the SVD of a cmatrix. An analogue of this algorithm for matrices can be found in [12, pp. 15–16]. Alternatively, the cmatrix QR factorization can be used in the first step and then the algorithm is a cmatrix analogue of the algorithm in [34]. The standard matrix SVD algorithm by Golub and Reinsch [60] involves bidiagonalization and a continuous analogue is more subtle. This algorithm is theoretically justified if $\mathcal{A}$ satisfies the assumptions in Theorem 4.6.

lower-triangular quasimatrix. (In theory this algorithm is only justified for cmatrices that satisfy the assumptions in Theorem 4.6, but in practice it works when there is much less smoothness. We expect that Theorem 4.6 can be proven under weaker assumptions.)

We have already covered the main features of the SVD, described for functions rather than cmatrices. Theorems 3.1 and 3.2 related the smoothness of $\mathcal{A}$ to the decay rates of its singular values, and the basic properties of the cmatrix SVD were given in Theorem 3.3.

If $\mathcal{A}$ is a nonnegative definite Hermitian cmatrix (see Table 4.2), then continuity is enough (Lipschitz continuity is not necessary) to ensure that the SVD series in (4.2) is absolutely and uniformly convergent. This is Mercer's Theorem [107].

4.9 The QR factorization of a cmatrix

The QR factorization of a cmatrix is also an infinite series.

Definition 4.8. *Let $\mathcal{A}$ be an $[a,b]\times[c,d]$ continuous cmatrix. A QR factorization of $\mathcal{A}$ is an infinite series,*

$$\mathcal{A}=\sum_{j=1}^{\infty} q_j(y)r_j^*(x), \tag{4.5}$$

Algorithm: Gram–Schmidt with column pivoting on cmatrices

Input: An $[a,b]\times[c,d]$ continuous cmatrix $\mathcal{A}$.
Output: A QR series $\mathcal{A}=\sum_{j=1}^{\infty} q_j(y)r_j^*(x)$.
$\mathcal{E}_0(y,x)=\mathcal{A}(y,x)$, $\|\cdot\|=\|\cdot\|_{L^2([a,b])}$
for $j=1,2,\ldots$
$x_j=\arg\max_{x\in[c,d]}(\|\mathcal{E}_{j-1}(\cdot,x)\|)$
$\mathcal{E}_j(y,x)=\mathcal{E}_{j-1}(y,x)-\frac{\mathcal{E}_{j-1}(y,x_j)}{\|\mathcal{E}_{j-1}(\cdot,x_j)\|^2}\int_a^b \overline{\mathcal{E}_{j-1}(s,x_j)}\mathcal{E}_{j-1}(s,x)ds$
$q_j(y)=\pm\mathcal{E}_{j-1}(y,x_j)/\|\mathcal{E}_{j-1}(\cdot,x_j)\|$
$r_j^*(x)=\pm\int_a^b \overline{\mathcal{E}_{j-1}(s,x_j)}\mathcal{E}_{j-1}(s,x)/\|\mathcal{E}_{j-1}(\cdot,x_j)\|ds$
end

Figure 4.6: Gram–Schmidt orthogonalization with column pivoting on cmatrices. In practice the algorithm is terminated after k steps if $\max(\|\mathcal{E}_{k+1}(\cdot,x)\|)$ is less than a prescribed tolerance leading to a rank k approximation of $\mathcal{A}$.

where $\mathcal{Q}$ is a continuous quasimatrix with orthonormal columns $q_1,q_2,\ldots$, and $\mathcal{R}$ is an upper-triangular continuous quasimatrix with rows $r_1^*,r_2^*,\ldots$. The equality in (4.5) *should be understood to signify that the series converges pointwise to $\mathcal{A}$.*

For this factorization, pivoting in the horizontal variable is essential and can be achieved by a continuous analogue of Gram–Schmidt orthogonalization with column pivoting. Figure 4.6 gives the pseudocode for the algorithm, and to make this continuous idealization practical it must be discretized, involving adaptivity. Alternatively, one could discretize the function on a sufficiently large tensor grid and perform a weighted (with the Gauss–Legendre weights) QR with column pivoting on the resulting matrix. If one interpolated the resulting matrix factors to form quasimatrices, then this would construct a QR factorization of a cmatrix.

In the `qr` command in Chebfun2 we employ a less direct algorithm that happens to be easier to implement. We first compute the LU factorization of the cmatrix $\mathcal{A}=\mathcal{L}\mathcal{U}$ (see Section 4.10) before taking the QR factorization of the lower-triangular quasimatrix $\mathcal{L}=\mathcal{Q}R$ (see Section 4.6). This means we can write $\mathcal{A}=\mathcal{Q}(R\mathcal{U})$, which is a QR factorization of $\mathcal{A}$. This is easier to implement because we have already invested a significant amount of time into Gaussian elimination on cmatrices and designed specialized adaptivity and resolution testing (see Section 2.1.1).

To justify writing the series in (4.5) as $\mathcal{A}=\mathcal{Q}\mathcal{R}$ we require that the series is unconditionally convergent to $\mathcal{A}$ (the product of quasimatrices does not impose the order in which the rank 1 terms are summed). To guarantee this we prove that

the series in (4.5) converges absolutely to $\mathcal{A}$, when $\mathcal{A}$ satisfies certain smoothness assumptions.

Theorem 4.5 (Convergence of QR)**.** *Let $\mathcal{A}$ be an $[a,b]\times[c,d]$ cmatrix that is Lipschitz continuous in the y-variable (uniformly in x) and such that, uniformly in $y \in [a,b]$, the row $\mathcal{A}(y,\cdot)$ is analytic and analytically continuable in the x-variable to a function bounded in absolute value by a fixed constant in a region containing the Bernstein ellipse with foci c and d, where the semiminor and semimajor axis lengths sum to $\sqrt{2}\rho(d-c)$ with $\rho > 1$. Then, its QR series (with column pivoting) converges uniformly, absolutely, and geometrically with rate $\rho^{2/3}$ to $\mathcal{A}$.*

Proof. Let $\mathcal{A}$ be Lipschitz continuous in the y-variable with Lipschitz constant $C < \infty$ (uniformly in x). The Gram–Schmidt process is such that $\mathcal{E}_j$ in Figure 4.6 is Lipschitz continuous in the y-variable with a Lipschitz constant that at worst doubles at each step,

$$\|\mathcal{E}_j(y_1,\cdot) - \mathcal{E}_j(y_2,\cdot)\|_{L^\infty([c,d])} \le 2^j C|y_1 - y_2|, \qquad y_1, y_2 \in [a,b].$$

Since $\mathcal{A}$ is continuous on $[a,b]\times[c,d]$, so is $\mathcal{E}_j$, and hence we can pick $(t_j, s_j) \in [a,b]\times[c,d]$ such that $|\mathcal{E}_j(t_j,s_j)| = \|\mathcal{E}_j\|_\infty$. By Lemma B.1 with $p = 2$ we have

$$\|\mathcal{E}_j\|_\infty \le \max\left(2(2^jC)^{1/3}\|\mathcal{E}_j(\cdot,s_j)\|_{L^2}^{2/3}, \frac{2}{\sqrt{b-a}}\|\mathcal{E}_j(\cdot,s_j)\|_{L^2}\right).$$

Moreover, $\|\mathcal{E}_j(\cdot,s_j)\|_{L^2} \le \sup_x \|\mathcal{E}_j(\cdot,x)\|_{L^2} = |r_{j+1}(x_{j+1})|$, where the last equality holds since $|r_{j+1}(x_{j+1})| = \|\mathcal{E}_j(\cdot,x_{j+1})\|_{L^2}$ and by column pivoting $\|\mathcal{E}_j(\cdot,x_{j+1})\|_{L^2} = \sup_x \|\mathcal{E}_j(\cdot,x)\|_{L^2}$. As a further consequence of column pivoting we have $|r_{j+1}(x_{j+1})| \le |r_j(x_j)|$ and hence,

$$\|\mathcal{E}_j\|_\infty \le \max\left(2(2^jC)^{1/3}|r_j(x_j)|^{2/3}, \frac{2}{\sqrt{b-a}}|r_j(x_j)|\right). \tag{4.6}$$

Now, consider the $j\times j$ submatrix $\tilde{R}_j$ satisfying $(\tilde{R}_j)_{st} = r_s(x_t)$ for $1 \le s,t \le j$. The matrix $\tilde{R}_j$ is upper-triangular with the absolute value of its diagonal entries bounded below by $|r_j(x_j)|$. If $|r_j(x_j)| = 0$ then $\sup_x \|\mathcal{E}_{j-1}(\cdot,x)\|_{L^2} = 0$ and the QR factorization of $\mathcal{A}$ contains only finitely many terms so the convergence results follow. If $|r_j(x_j)| \neq 0$ then $\tilde{R}_j$ is invertible and by the same reasoning as in [153, Thm. 9.2] we have,

$$\|\tilde{R}_j^{-1}\|_2 \le 2^j/|r_j(x_j)|.$$

Now, let $\tilde{\mathcal{A}}_j$ be the $x_1, \ldots, x_j$ columns of $\mathcal{A}$ and $\tilde{\mathcal{Q}}_j$ be the first j columns of $\mathcal{Q}$, respectively. Then, $\tilde{\mathcal{A}}_j = \tilde{\mathcal{Q}}_j \tilde{R}_j$ and the minimum singular value of $\tilde{\mathcal{A}}_j$ is equal to the minimum singular value of $\tilde{R}_j$. Thus,

$$|r_j(x_j)| \leq 2^j / \|\tilde{R}_j^{-1}\|_2 \leq 2^j \sigma_j(\tilde{R}_j) = 2^j \sigma_j(\tilde{\mathcal{A}}_j),$$

and combining with (4.6), we get

$$\|\mathcal{E}_j\|_\infty \leq \max\left(2^{j+1} C^{1/3} (\sigma_j(\tilde{\mathcal{A}}_j))^{2/3}, \frac{2^{j+1}}{\sqrt{b-a}} \sigma_j(\tilde{\mathcal{A}}_j)\right). \tag{4.7}$$

By the best rank property of the quasimatrix SVD (see Theorem 4.2) we have

$$\sigma_j(\tilde{\mathcal{A}}_j) = \inf_{\mathcal{B}_{j-1}} \left\| \tilde{\mathcal{A}}_j - \mathcal{B}_{j-1} \right\|_2,$$

where the infimum is taken over $[a,b] \times j$ quasimatrices $\mathcal{B}_{j-1}$ of rank at most $j-1$. Moreover, for any $[a,b] \times j$ continuous quasimatrix $\mathcal{F}$ and $v \in \mathbb{C}^j$ with $\|v\|_2 = 1$ we have

$$\|\mathcal{F}\|_2^2 \leq \|\mathcal{F}v\|_2^2 = \int_a^b \left| \sum_{i=1}^{j} v_i \mathcal{F}(y,i) \right|^2 dy \leq j(b-a) \sup_{y \in [a,b]} \sup_{1 \leq i \leq j} |\mathcal{F}(y,i)|^2.$$

Thus, for any $[a,b] \times j$ quasimatrix $\mathcal{B}_{j-1}$ of rank at most $j-1$ we have

$$\sigma_j(\tilde{\mathcal{A}}_j) \leq \sqrt{j(b-a)} \sup_{y \in [a,b]} \sup_{1 \leq i \leq j} \left| \tilde{\mathcal{A}}_j(y,i) - \mathcal{B}_{j-1}(y,i) \right|.$$

Our analyticity assumption on $\mathcal{A}$ implies that it can be approximated to a uniform accuracy of $\mathcal{O}((2\sqrt{2}(\rho+\epsilon))^{-j})$ for some $\epsilon > 0$ by functions that are polynomials of degree $j-2$ in the x-variable (see Theorem 3.2). When a quasimatrix is formed by taking the $x_1, \ldots, x_j$ columns from such a function the resulting quasimatrix is of rank at most $j-1$. Combining this observation with (4.7) shows that $\|\mathcal{E}_j\|_\infty = \mathcal{O}(\rho^{-2j/3}) = \mathcal{O}(\rho_*^{-j})$ as $j \to \infty$, where $\rho_* = \rho^{2/3} > 1$. That is, the convergence is uniform and geometrically fast.

Finally, we consider the absolute convergence of the QR series. By definition we

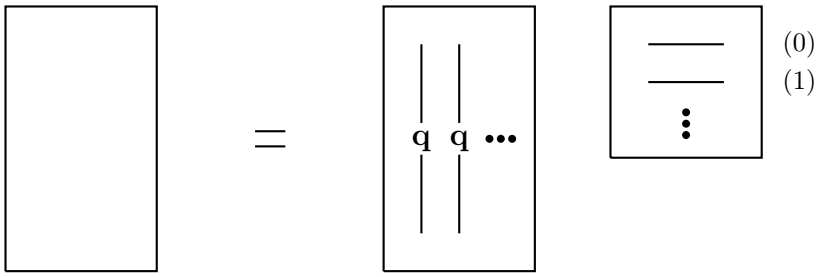


Figure 4.7: The QR factorization of an $[a,b]\times[c,d]$ continuous cmatrix: $\mathcal{A}=\mathcal{Q}\mathcal{R}$, where $\mathcal{Q}$ is an $[a,b]\times\infty$ continuous quasimatrix with orthonormal columns, and $\mathcal{R}$ is an $\infty\times[c,d]$ upper-triangular continuous quasmatrix. The q's indicate orthonormal columns.

have

$$\|q_j\|_\infty \le \frac{\|\mathcal{E}_{j-1}(\cdot,x_j)\|_\infty}{\|\mathcal{E}_{j-1}(\cdot,x_j)\|_{L^2}},$$

and by the Cauchy–Schwarz inequality

$$\|r_j\|_\infty \le \|\mathcal{E}_{j-1}(\cdot,x_j)\|_{L^2}.$$

Therefore, we have absolute convergence

$$\sum_{j=1}^{\infty}|q_j(y)||r_j(x)| \le \sum_{j=1}^{\infty}\|\mathcal{E}_{j-1}(\cdot,x_j)\|_\infty \le \sum_{j=1}^{\infty}\|\mathcal{E}_{j-1}\|_\infty \le \sum_{j=1}^{\infty}\mathcal{O}(\rho_*^{-j}) < \infty.$$

□

Theorem 4.5 shows that Gram–Schmidt orthogonalization with column pivoting can construct a QR factorization of a cmatrix. Figure 4.7 shows the QR factorization of a cmatrix, $\mathcal{A}=\mathcal{Q}\mathcal{R}$, which we know exists for cmatrices satisfying the assumptions of Theorem 4.5.

4.10 The LU factorization of a cmatrix

The LU factorization of a cmatrix involves both column and row pivoting and is an important factorization for Chebfun2 as it is the main factorization used for approximating functions (see Chapter 2). It is also related to adaptive cross approximation, the CUR decomposition, interpolative decompositions, and Geddes–Newton approximation (see Section 2.1.3). The factorization can be defined as a series.

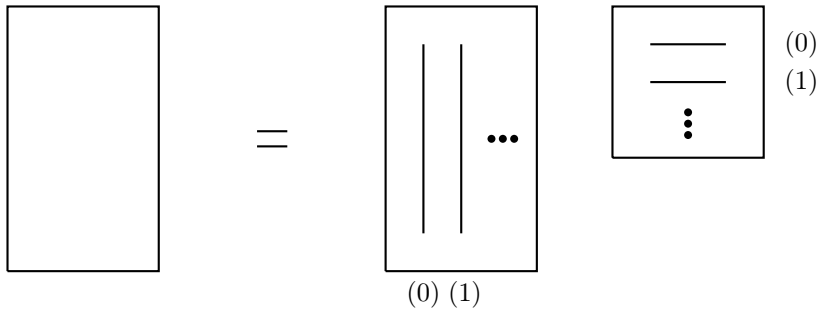


Figure 4.8: The LU factorization of an $[a,b] \times [c,d]$ cmatrix: $\mathcal{A} = \mathcal{L}\mathcal{U}$, where $\mathcal{L}$ is an $[a,b] \times \infty$ unit lower-triangular continuous quasimatrix and $\mathcal{U}$ is an $\infty \times [c,d]$ upper-triangular continuous quasmatrix.

Definition 4.9. *Let $\mathcal{A}$ be an $[a,b] \times [c,d]$ continuous cmatrix. An LU factorization of $\mathcal{A}$ is an infinite series,*

$$\mathcal{A}(y,x) = \sum_{j=1}^{\infty} \ell_j(y) u_j^*(x), \tag{4.8}$$

where $\ell_1, \ell_2, \ldots$ and $u_1^*, u_2^*, \ldots$ are continuous, the $[a,b] \times \infty$ quasimatrix with columns $\ell_1, \ell_2, \ldots$ is unit lower-triangular, and the $\infty \times [a,b]$ quasimatrix with rows $u_1^*, u_2^*, \ldots$ is upper-triangular. The equality in (4.8) *should be understood to signify that the series converges pointwise to $\mathcal{A}$.*

In order to be able to write (4.8) as $\mathcal{A} = \mathcal{L}\mathcal{U}$ we must prove that the series converges absolutely and uniformly to $\mathcal{A}$, and under appropriate assumptions Figure 4.8 shows the decomposition. The nested set of zeros now occurs in the columns of $\mathcal{L}$ and in the rows of $\mathcal{U}$, as indicated by the numbers in parentheses. One would imagine that relatively weak smoothness conditions on $\mathcal{A}$ are required (such as Lipschitz continuity) to be able to write $\mathcal{A} = \mathcal{L}\mathcal{U}$, but so far we have only been able to prove this under assumptions that are far from weak. In practice we observe convergence under much milder assumptions.

Theorem 4.6 (Convergence of LU)**.** *Let $\mathcal{A}$ be an $[a,b] \times [c,d]$ continuous cmatrix. Suppose that the column $A(\cdot, x)$ is analytically continuable to an analytic function bounded in absolute value by a fixed constant in the closed region consisting of complex numbers at a distance of at most $2\rho(b-a)$ from $[a,b]$ for some $\rho > 1$, uniformly in $x \in [a,b]$. Then, the series constructed by Gaussian elimination converges absolutely, uniformly, and geometrically with rate ρ to $\mathcal{A}$.*

Proof. Fix $x \in [c,d]$ and let e_k denote the error function at step k,

$$e_k = \mathcal{A}(\cdot, x) - \sum_{j=1}^{k} \ell_j(\,\cdot\,) u_j^*(x),$$

a function of $y \in [a,b]$. Furthermore, let K be the closed region consisting of complex numbers at a distance of at most $2\rho(b-a)$ from $[a,b]$ for some $\rho > 1$. By assumptions in the theorem, $\mathcal{A}(\cdot, x)$ is analytic in K and by the elimination process so is e_k, with the bound on the magnitude of e_k that is at worst doubling at each step,

$$|e_k(z)| \le 2^k M, \qquad z \in K, \tag{4.9}$$

where $M < \infty$ is a constant such that for any $x \in [c,d]$ we have $\sup_{z\in K} |\mathcal{A}(z,x)| \le M$.

By Theorem 2.1, e_k has at least k zeros $y_1, \ldots, y_k$ in $[a,b]$. Let $p_k(y)$ be the polynomial $(y-y_1)\cdots(y-y_k)$. Then e_k/p_k is analytic in K and hence, satisfies the maximum modulus principle within K. For any $y \in [a,b]$, this implies

$$|e_k(y)| \le |p_k(y)| \sup_{z\in\partial K} \frac{|e_k(z)|}{|p_k(z)|} \le 2^k M \sup_{z\in\partial K} \frac{|p_k(y)|}{|p_k(z)|}.$$

In this quotient of polynomials, each of the k factors in the denominator is at least 2ρ times bigger in modulus than the corresponding factor in the numerator. We conclude that $|e_k(y)| \le \rho^{-k} M$ for $y \in [a,b]$. Since this error estimate applies independently of x and y, it establishes uniform convergence. It also implies that the next term $\ell_{k+1} u_{k+1}^*$ in the series is bounded in absolute value by $\rho^{-k} M$, which gives absolute convergence since $\sum_{k=0}^{\infty} \rho^{-k} M < \infty$. □

The theorem is equally applicable to the convergence of adaptive cross approximation and Geddes–Newton approximation in two dimensions (see Section 2.1.3). An analogous theorem holds when the analyticity assumptions on $\mathcal{A}$ are imposed on the rows, $\mathcal{A}(y,\cdot)$ for $y \in [a,b]$, rather than the columns.

The region of analyticity in Theorem 4.6 appears in 1D approximation theory, where it is called a "stadium" (of radius $2\rho(b-a)$) [159, p. 82]. If a function $f : [a,b] \to \mathbb{C}$ is analytic in a stadium of radius $\rho(b-a)$ with $\rho > 1$, then a sequence $\{p_n\}_{n\ge 1}$ of polynomials such that p_n interpolates f at n distinct points in $[a,b]$ converges uniformly and geometrically to f as $n \to \infty$, irrespective of how the interpolation nodes are distributed in $[a,b]$ [54, p. 63]. Here, double the radius, $2\rho(b-a)$, is required

to compensate for the potential doubling in (4.9). When $\mathcal{A}$ satisfies the assumptions in Theorem 4.6 we can write the LU series as a decomposition, $\mathcal{A} = \mathcal{L}\mathcal{U}$.

Corollary 4.2. *Let $\mathcal{A}$ be an $[a,b] \times [c,d]$ cmatrix satisfying the assumptions of Theorem 4.6. Then $\mathcal{A} = \mathcal{L}\mathcal{U}$, where $\mathcal{L}$ is an $[a,b] \times \infty$ unit lower-triangular continuous quasimatrix and $\mathcal{U}$ is an $[c,d] \times \infty$ upper-triangular continuous quasimatrix.*

Proof. By Theorem 4.6 the series in (4.8) is absolutely and uniformly convergent. In particular, it is unconditionally and pointwise convergent. □

The algorithm we use for the cmatrix LU factorization is described in Section 2.1. In Chapter 2 the iterative GE algorithm constructed a CDR factorization (2.2), but this can be easily converted into an LU factorization by sharing out scaling factors in the diagonal matrix, D, appropriately. Every function in Chebfun2 starts from an LU factorization of finite rank that is truncated so the series is accurate to about machine precision. Further operations, such as the integration, differentiation, and evaluation, exploit the resulting low rank structure (see Chapter 2). The remarkable ability of GE with complete pivoting to construct near-optimal low rank approximations to cmatrices is a central observation underlying the adaptive approximation algorithm of Chebfun2 (see Section 2.1.2).

4.11 The Cholesky factorization of a cmatrix

Our last factorization is a continuous analogue of the Cholesky factorization.

Definition 4.10. *Let $\mathcal{A}$ be an $[a,b] \times [a,b]$ continuous cmatrix. A Cholesky factorization of $\mathcal{A}$ is an infinite series,*

$$\mathcal{A}(y,x) = \sum_{j=1}^{\infty} r_j(y) r_j^*(x), \tag{4.10}$$

where $r_1^*, r_2^*, \ldots$ are continuous and the $\infty \times [a,b]$ quasimatrix $\mathcal{R}$ with rows $r_1^*, r_2^*, \ldots$ is upper-triangular. The equality in (4.10) should be understood to signify that the series is pointwise convergent to $\mathcal{A}$.

Unlike the situation with the other factorizations not every sufficiently smooth cmatrix can have a Cholesky factorization since if the series in (4.10) is absolutely and uniformly convergent, then $\mathcal{A}$ must be Hermitian, i.e., $\mathcal{A}(y,x) = \overline{\mathcal{A}(x,y)}$, and *nonnegative definite*.

Definition 4.11. *An $[a,b] \times [a,b]$ continuous cmatrix is nonnegative definite if for all continuous functions $v \in \mathcal{C}([a,b])$,*

$$v^* \mathcal{A} v = \int_a^b \int_a^b \overline{v(y)} \mathcal{A}(y,x) v(x) dx dy \geq 0.$$

We would like to be able to prove the stronger result: a Hermitian cmatrix has a Cholesky factorization if and only if it is nonnegative definite. We establish this in Theorem 4.7 below under a fairly restrictive set of hypotheses on the smoothness of $\mathcal{A}$, though we suspect this statement holds under more modest assumptions. Before proving Theorem 4.7 we need to derive several results about Hermitian and nonnegative definite cmatrices.

Lemma 4.1. *Let $\mathcal{A}$ be a continuous Hermitian $[a,b] \times [a,b]$ cmatrix. Then, the diagonal entries of $\mathcal{A}$ are real. In addition, if $\mathcal{A}$ is nonnegative definite then the diagonal entries are nonnegative and for any $x, y \in [a,b]$ we have $|\mathcal{A}(y,x)|^2 \leq \mathcal{A}(y,y)\mathcal{A}(x,x)$. Moreover, an absolute maximum of $\mathcal{A}$ occurs on the diagonal.*

Proof. If $\mathcal{A}$ is Hermitian then for $x \in [a,b]$ we have $\mathcal{A}(x,x) = \overline{\mathcal{A}(x,x)}$ so the diagonal entries are real. If $\mathcal{A}$ is nonnegative definite, then by [107, Sec. 5] we have $\mathcal{A}(x,x) \geq 0$. Moreover, by [107, eq. (6)] (with $n = 2$, $s_1 = x$, and $s_2 = y$) the 2×2 matrix

$$B = \begin{bmatrix} \mathcal{A}(y,y) & \mathcal{A}(y,x) \\ \mathcal{A}(x,y) & \mathcal{A}(x,x) \end{bmatrix} = \begin{bmatrix} \mathcal{A}(y,y) & \mathcal{A}(y,x) \\ \overline{\mathcal{A}(y,x)} & \mathcal{A}(x,x) \end{bmatrix}$$

is nonnegative definite and hence $\det(B) = \mathcal{A}(y,y)\mathcal{A}(x,x) - |\mathcal{A}(y,x)|^2 \geq 0$. Therefore, we have $|\mathcal{A}(y,x)|^2 \leq \sup_{s \in [a,b]} |\mathcal{A}(s,s)|^2$ for any $x, y \in [a,b]$ and hence, an absolute maximum of $\mathcal{A}$ occurs on the diagonal. □

The lower- and upper-triangular quasimatrix factors in a Cholesky factorization of cmatrix are conjugate transposes of each other, and thus they must be triangular with respect to the same indexing sequence (see Definition 4.2). Therefore, to select a pivot the Cholesky algorithm searches along the diagonal. The pivoting strategy we use is *diagonal pivoting*, where the location of an absolute maximum diagonal entry of $\mathcal{A}$ is taken as the next pivot.

The Cholesky algorithm with diagonal pivoting can be described as follows: Let $\mathcal{A}$ be a Hermitian nonnegative definite $[a,b] \times [a,b]$ cmatrix, and set $\mathcal{E}_0 = \mathcal{A}$. At step $k = 1$, select $x_1 \in [a,b]$ for which $\mathcal{E}_0(x_1,x_1) = \sup_{s \in [a,b]} \mathcal{E}_0(s,s)$. (The diagonal entries

are necessarily real and nonnegative by Lemma 4.1.) Let γ_1 be the nonnegative real square root of $\mathcal{E}_0(x_1, x_1)$ and define $r_1 = \mathcal{E}_0(\cdot, x_1)/\gamma_1$ and $\mathcal{E}_1 = \mathcal{E}_0 - r_1 r_1^*$. (If $\mathcal{E}_0(x_1, x_1) = 0$, $\mathcal{A}$ is the zero cmatrix and we take r_1 to be the zero function in $\mathcal{C}([a, b])$.) The cmatrix $\mathcal{E}_1$ is zero in row x_1 and column x_1, and $\mathcal{E}_1$ is Hermitian and nonnegative definite itself (by the Schur Complement Lemma [141, Thm. 2.6]). At step $k = 2$, select an $x_2 \in [a, b]$ for which $\mathcal{E}_1(x_2, x_2) = \sup_{s\in[a,b]} \mathcal{E}_1(s, s)$. Let γ_2 be the nonnegative real square root of $\mathcal{E}_1(x_2, x_2)$ and define $r_2 = \mathcal{E}_1(\cdot, x_2)/\gamma_2$ and $\mathcal{E}_2 = \mathcal{E}_1 - r_2 r_2^*$. The cmatrix $\mathcal{E}_2$ is zero in rows and columns x_1 and x_2, and $\mathcal{E}_2$ is Hermitian and nonnegative definite. We continue in this fashion, generating the Cholesky factorization in (4.10) term-by-term.

We are able to show that the resulting Cholesky series is absolutely and uniformly convergent when $\mathcal{A}$ is sufficiently smooth.

Theorem 4.7 (Convergence of Cholesky)**.** *Let $\mathcal{A}$ be an $[a, b] \times [a, b]$ continuous Hermitian nonnegative definite cmatrix. Suppose the function $\mathcal{A}(\cdot, x)$ can be extended to an analytic function bounded in absolute value by a fixed constant in the closed region contained in the Bernstein ellipse with foci a and b with semiaxes lengths summing to $2\rho(b - a)$ for some $\rho > 1$, uniformly in $y \in [a, b]$. Then, the series in* (4.10) *constructed by the Cholesky algorithm converges absolutely, uniformly, and geometrically at rate ρ to $\mathcal{A}$.*

Proof. Let the first $k + 1$ pivots of the Cholesky algorithm be $x_1, \ldots, x_{k+1} \in [a, b]$. Since we are using diagonal pivoting and an absolute maximum occurs on the diagonal (Lemma 4.1) we have

$$\|\mathcal{E}_k\|_\infty = r_{k+1}(x_{k+1})^2 \leq r_k(x_k)^2. \tag{4.11}$$

Moreover, $\sum_{j=1}^{k} r_j r_j^*$ interpolates $\mathcal{A}$ along the lines $x = x_j$ and $y = x_j$ for $1 \leq j \leq k$ and therefore, there is an underlying $k \times k$ matrix Cholesky factorization:

$$A_k = R_k^* R_k, \qquad (A_k)_{ij} = \mathcal{A}(x_i, x_j), \qquad (R_k)_{ij} = \overline{r_j(x_i)}, \qquad 1 \leq i, j \leq k.$$

From [80] we have the bound $\left\|R_k^{-1}\right\|_2^2 \leq 4^k / r_k(x_k)^2$ and hence,

$$\sigma_k(A_k)^{-1} = \left\|A_k^{-1}\right\|_2 = \left\|R_k^{-1}\right\|_2^2 \leq 4^k / r_k(x_k)^2.$$

After rearranging we obtain $r_k(x_k)^2 \leq 4^k \sigma_k(A)$. Thus, by combining with (4.11) we have

$$\|\mathcal{E}_k\|_\infty \leq 4^k \sigma_k(A_k) = 4^k \inf \|A_k - C_{k-1}\|_2 ,$$

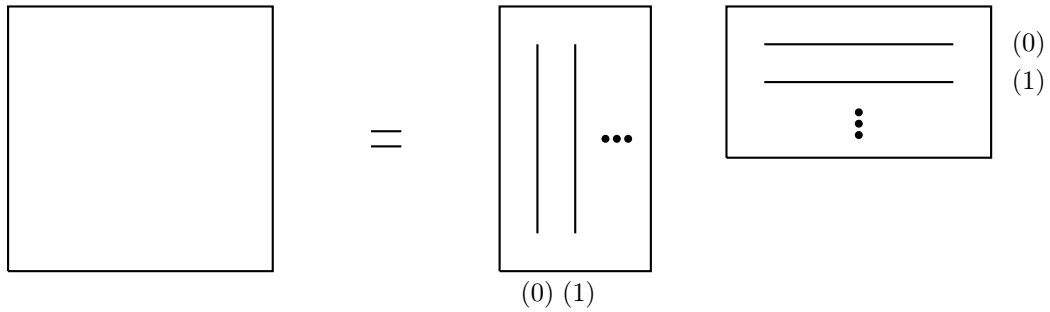


Figure 4.9: The Cholesky factorization of an $[a,b] \times [a,b]$ cmatrix: $\mathcal{A} = \mathcal{R}^*\mathcal{R}$, where $\mathcal{R}$ is an $\infty \times [a,b]$ upper-triangular continuous quasimatrix.

where the infimum is taken over $k \times k$ matrices of rank $k-1$. By switching from the 2-norm to the maximum entry norm we have

$$\|\mathcal{E}_k\|_\infty \le k4^k \inf \max_{i,j} |(A_k - C_{k-1})_{ij}| .$$

Now, by Theorem 3.2 and our analyticity assumptions, $\mathcal{A}$ can be approximated to an accuracy of $\mathcal{O}((4\rho)^{-k})$ by polynomials of degree $k-2$ with respect to one of the variables. Evaluating this polynomial approximation on the $k \times k$ tensor product grid $\{(x_i, x_j) : 1 \le i, j \le k\}$ leads to a matrix C_{k-1} that is of rank at most $k-1$ (see Section 3.1) and satisfies $|(A_k - C_{k-1})_{ij}| = \mathcal{O}((4\rho)^{-k})$ with $\rho > 1$, which completes the proof. □

Theorem 4.7 is a continuous analogue of a convergence result for the pivoted Cholesky algorithm for matrices [78]. When the Cholesky series in (4.10) is absolutely and uniformly convergent we are able to write the decomposition $\mathcal{A} = \mathcal{R}^*\mathcal{R}$, as shown in Figure 4.9.

The `chol` command in Chebfun2 does not use the Cholesky algorithm, but instead starts from the cmatrix LU factorization already computed when a chebfun2 is first realized. The Cholesky factors are obtained by appropriately rescaling the lower- and upper-triangular quasimatrix factors.

4.11.1 Test for nonnegative definite functions

The Cholesky algorithm is often used to test numerically if a matrix is positive definite. This simple and useful test can be generalized to cmatrices.

Clearly, if a Hermitian cmatrix $\mathcal{A}$ can be written as $\mathcal{A} = \mathcal{R}^*\mathcal{R}$, then for any continuous function $v \in \mathcal{C}([a,b])$,

$$v^*\mathcal{A}v = v^*\mathcal{R}^*\mathcal{R}v = (\mathcal{R}v)^*(\mathcal{R}v) \geq 0$$

and hence, $\mathcal{A}$ is nonnegative definite. We also have a partial converse.

Theorem 4.8 (Test for nonnegative definite functions)**.** *Let $\mathcal{A}$ be an $[a,b] \times [a,b]$ Hermitian nonnegative definite cmatrix satisfying the assumptions in Theorem* 4.7*. Then $\mathcal{A}$ has a Cholesky factorization $\mathcal{A} = \mathcal{R}^*\mathcal{R}$.*

Proof. The Cholesky algorithm completes without breaking down if $\mathcal{A}$ is Hermitian and nonnegative definite. By Theorem 4.7 the series converges absolutely and uniformly and hence $\mathcal{A} = \mathcal{R}^*\mathcal{R}$. □

We use the Cholesky algorithm as a numerical test for nonnegative definite functions even when they are not smooth enough to satisfy the assumptions in Theorem 4.7. First, we approximate the function by a chebfun2 and then check if all the GE pivots are nonnegative and lie on the diagonal,[6] i.e., the line $y = x$. For example, the inverse multiquadric function with $\epsilon > 0$

$$\mathcal{A}(y,x) = 1/(1 + \epsilon(x^2 + y^2)) \tag{4.12}$$

is nonnegative definite on $[-1,1]^2$ (or any other square domain), and this can be numerically verified by checking that all its GE pivots are nonnegative and lie on the diagonal. Figure 4.10 (left) shows the pivot locations for the inverse multiquadric.

As another example, we can numerically verify that $\mathcal{A}^*\mathcal{A}$ is a nonnegative cmatrix. For instance, take $\mathcal{A}(y,x) = \cos(10xy) + y + x^2 + \sin(10xy)$ on $[-1,1]^2$ and calculate $\mathcal{B} = \mathcal{A}^*\mathcal{A}$, i.e.,

$$\mathcal{B}(y,x) = \int_{-1}^{1} \overline{\mathcal{A}(s,y)}\mathcal{A}(s,x)ds.$$

Then, $\mathcal{B}$ is a nonnegative definite cmatrix. Figure 4.10 (right) shows that the pivot locations are on the diagonal, and in addition the pivot values are nonnegative.

[6] If there is a tie between an off-diagonal and a diagonal absolute maximum, the absolute maximum occurring on the diagonal is taken.

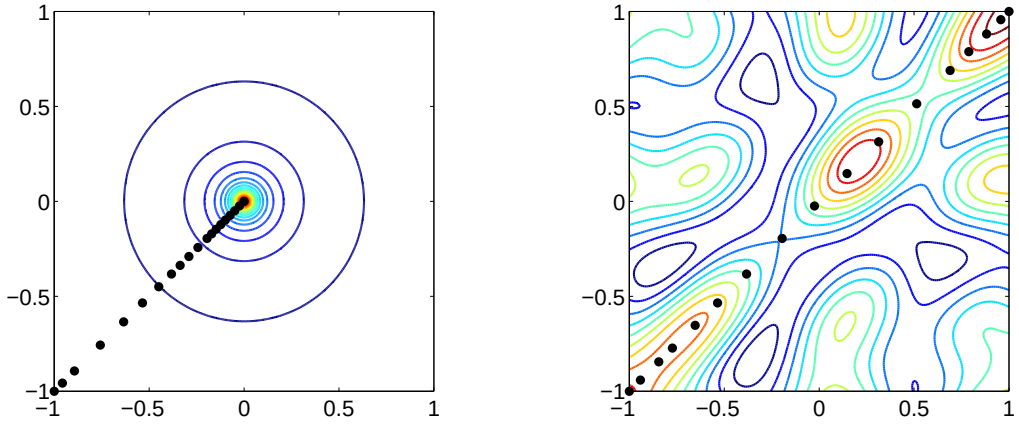


Figure 4.10: The Cholesky algorithm can be used as a numerical test for nonnegative definite cmatrices. Here, it is applied to the inverse multiquadric function in (4.12) with $\epsilon = 1{,}000$ (left) and $\mathcal{A}^*\mathcal{A}$ with $\mathcal{A}(y,x) = \cos(10xy) + y + x^2 + \sin(10xy)$ (right). The black dots are the locations of the pivots taken by the algorithm.

4.12 More on continuous linear algebra

This chapter has concentrated on continuous analogues of matrix factorizations, but many other matrix concepts also have analogues for quasimatrices and cmatrices. Table 4.2 summarizes a selection for cmatrices.

Category	Matrix setting	Cmatrix analogue
Special matrices	Toeplitz	$\mathcal{A}(y,x) = \phi(x-y)$
	Hankel	$\mathcal{A}(y,x) = \phi(x+y)$
	Cauchy	$\mathcal{A}(y,x) = 1/(x+y)$
	Z-matrix	$\mathcal{A}(y,x) \leq 0,\ x \neq y$
	Metzler	$\mathcal{A}(y,x) \geq 0,\ x \neq y$
	Rank 1	$\mathcal{A}(y,x) = g(y)h(x)$
Matrix operations	transpose	$\mathcal{A}^T(y,x) = \mathcal{A}(x,y)$
	conjugate transpose	$\mathcal{A}^*(y,x) = \overline{\mathcal{A}(x,y)}$
	matrix addition	$(\mathcal{A}+\mathcal{B})(y,x) = \mathcal{A}(y,x) + \mathcal{B}(y,x)$
	matrix-vector product	$\mathcal{A}v = \int \mathcal{A}(y,s)v(s)\,ds$
	vector-matrix product	$v^*\mathcal{A} = \int \overline{v(s)}\mathcal{A}(s,x)\,ds$
	matrix-matrix product	$\mathcal{A}\mathcal{B} = \int \mathcal{A}(y,s)\mathcal{B}(s,x)\,ds$
	diagonal	$\operatorname{diag}(\mathcal{A}) = \mathcal{A}(x,x)$
	trace	$\operatorname{tr}(\mathcal{A}) = \int \mathcal{A}(s,s)\,ds$
Matrix properties	symmetric	$\mathcal{A}^T = \mathcal{A}$
	Hermitian	$\mathcal{A}^* = \mathcal{A}$
	normal	$\mathcal{A}^*\mathcal{A} = \mathcal{A}\mathcal{A}^*$
	diagonally dominant	$\lvert\mathcal{A}(y_0,y_0)\rvert \geq \int \lvert\mathcal{A}(y_0,s)\rvert\,ds,\ \forall y_0$
	positive definite	$\int\int v(y)\mathcal{A}(y,x)v(x)dxdy > 0,\ \forall v$
Matrix norms	maximum norm	$\lVert\mathcal{A}\rVert_{\max} = \sup_{(y,x)} \lvert\mathcal{A}(y,x)\rvert$
	Frobenius norm	$\lVert\mathcal{A}\rVert_F^2 = \int\int \lvert\mathcal{A}(y,x)\rvert^2\,dx\,dy$
	∞-norm	$\lVert\mathcal{A}\rVert_\infty = \sup_y \int \lvert\mathcal{A}(y,x)\rvert\,dx$
	2-norm	$\lVert\mathcal{A}\rVert_2 = \sup_{\lVert v\rVert_{L^2}=1} \lVert\mathcal{A}v\rVert_{L^2}$
	1-norm	$\lVert\mathcal{A}\rVert_1 = \sup_x \int \lvert\mathcal{A}(y,x)\rvert\,dy$
Spectral theory	eigenvalues/eigenvectors	$\mathcal{A}v = \lambda v$
	Rayleigh quotient	$R(\mathcal{A},v) = v^*\mathcal{A}v/\lVert v\rVert_{L^2},\ \lVert v\rVert_{L^2} \neq 0$
	field of values	$W(\mathcal{A}) = \{R(\mathcal{A},v) : \lVert v\rVert_{L^2} \neq 0\}$

Table 4.2: Continuous analogues of standard matrix definitions for cmatrices. When the matrix setting requires square matrices, the analogous cmatrix definition requires square cmatrices. If $\mathcal{A}$ is an $[a,b]\times[c,d]$ cmatrix, then $\mathcal{A}v$ requires v to be a function on $[c,d]$, $v^*\mathcal{A}$ requires v to be a function on $[a,b]$, and $\mathcal{A}\mathcal{B}$ requires $\mathcal{B}$ to be an $[c,d]\times I$ cmatrix for some interval $I \subseteq \mathbb{R}$.

Chapter 5

Bivariate rootfinding*

There are two operations on bivariate functions that are commonly referred to as *rootfinding*: (1) finding the zero level curves of $f(x, y)$ and (2) finding the common zeros of $f(x, y)$ and $g(x, y)$. These operations are mathematically quite different. The first, generically, has solutions along curves, while the second, generically, has isolated solutions.

In this chapter, we concentrate on the second problem: find all the solutions to

$$\begin{pmatrix} f(x,y) \\ g(x,y) \end{pmatrix} = 0, \qquad (x,y) \in [-1,1]^2, \tag{5.1}$$

with the assumption that the solutions are isolated. The algorithm we derive is a 2D analogue of the 1D rootfinder in Chebfun, which is based on recursive subdivision and computing the eigenvalues of a colleague matrix [27, 161]. Unlike the tensor product operations discussed in Section 2.2, we do not know how to exploit the low rank structure of f and g, and fundamentally new techniques are required. The 2D rootfinding algorithm we derive still relies on recursive subdivision and eigenvalues of matrices, but also employs additional techniques such as resultants, local refinement, and regularization.

Chebfun2 can also find the zero level curves of $f(x, y)$ via the `roots(f)` command. The underlying algorithm is based on a marching squares technique [96], which is used

*This chapter is based on a paper with Yuji Nakatsukasa and Vanni Noferini [113]. I developed the algorithm for the construction of Chebyshev–Bézout resultant matrices, proposed and analyzed the domain subdivision algorithm, and introduced many practical features. Noferini proposed resultant methods for 2D rootfinding, explained why spurious infinite eigenvalues are usually harmless, and developed the algebraic understanding of rootfinding and resultants. Nakatsukasa proved the conditioning of the eigenvalues of the Bézout matrix polynomial and was the lead programmer. We jointly wrote the paper, though this chapter is substantially different. I have not included Noferini's algebraic discussions or Nakatsukasa's conditioning analysis.

in computer graphics to display contour plots of scalar valued functions. Chebfun2 uses the same algorithm to calculate approximants to them. A short description is given in Section 5.4.2 and more details can be found in [152].

Throughout, we consider the rootfinding problem on $[-1,1]^2$, though the algorithm and its implementation are applicable to any bounded rectangular domain. We assume that f and g are continuously differentiable and the solutions to (5.1) are simple[1].

5.1 A special case

If f or g is of rank 1 then this structure can be exploited when solving (5.1). Without loss of generality suppose that $f(x,y) = c_f(y)r_f(x)$ is of rank 1. Then the real solutions (if any) of $f(x,y) = 0$ form lines parallel to the coordinate axes. (A line either has a y-intercept at a root of c_f or an x-intercept at a root of r_f.) The lines on $[-1,1]^2$ are easily computed by approximating c_f and r_f by Chebyshev interpolants on $[-1,1]$ and employing the 1D rootfinder in Chebfun. Afterwards, further 1D rootfinding problems can be solved involving g restricted to the solutions of $f(x,y) = 0$ on $[-1,1]^2$: $g(\cdot, y_0) = 0$ for each root $y_0 \in [-1,1]$ of c_f and $g(x_0, \cdot)$ for each root $x_0 \in [-1,1]$ of r_f. Thus, when f or g is of rank 1 the 2D rootfinding problem in (5.1) can be solved with existing 1D algorithms.

Moreover, in this special case, if f and g are polynomials then the number of solutions to $f = g = 0$ is generically finite. Suppose that f and g are nonzero polynomials $p = f$ and $q = g$ of degrees[2] (m_p, n_p) and (m_q, n_q), respectively, in addition to $f(x,y) = c_f(y)r_f(x)$. By the fundamental theorem of algebra, c_f has at most n_p roots $y_1, \ldots, y_{n_p}$ in $[-1,1]$ and each $g(\cdot, y_j)$ for $1 \le j \le n_p$ has at most m_q roots (assuming $g(\cdot, y_j) \not\equiv 0$). Hence, there are at most $n_p m_q$ pairs $(x,y) \in [-1,1]^2$ such that $c_f(y) = 0$ and $g(x,y) = 0$, all of which satisfy $f(x,y) = g(x,y) = 0$. Similarly, r_f has at most m_p roots $x_1, \ldots, x_{m_p}$ in $[-1,1]$ so there are at most $m_p n_q$ pairs $(x,y) \in [-1,1]^2$ such that $r_f(x) = 0$, $g(x,y) = 0$, and $f(x,y) = g(x,y) = 0$ (assuming $g(\cdot, x_j) \not\equiv 0$ for $1 \le j \le m_p$). We conclude that the set of solutions in this case to (5.1) either contains a line (when $g(\cdot, y_j) = 0$ or $g(x_j, \cdot) = 0$) or there are at most

$$m_p n_q + n_p m_q \tag{5.2}$$

[1] We say that a solution to $f = g = 0$ is simple if the Jacobian of f and g is invertible at the solution.

[2] A bivariate polynomial is of degree (m_p, n_p) if it is of degree m_p in x and n_p in y.

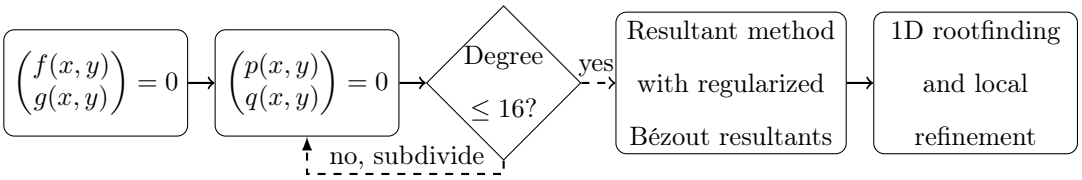


Figure 5.1: Flowchart of our 2D rootfinding algorithm based on a resultant method with Bézout resultants. First, f and g are approximated by polynomials and $[-1,1]^2$ is recursively subdivided until the polynomial degree on each subdomain is at most 16 (Section 5.5). Then, a component of the solutions is computed by a resultant method using regularized Bézout resultant matrices (Section 5.6). The remaining component is computed by a 1D rootfinder. Local refinement is employed to improve the accuracy of the final solutions (Section 5.8).

isolated solutions. The same maximum number of finite solutions is also true more generally (see Theorem 5.1).

5.2 Algorithmic overview

When f and g in (5.1) are of rank > 1 the problem is more challenging. For this case, our numerical algorithm involves several key steps.

First, in keeping with the usual Chebfun2 framework, we replace f and g by global polynomial approximants that are accurate to normwise relative machine precision on $[-1,1]^2$ (see Section 5.3). Then, we recursively subdivide $[-1,1]^2$ into rectangles and represent f and g by low degree piecewise polynomials (see Section 5.5). On each subdomain, one component of the solutions is computed by using a hidden variable resultant method with Bézout resultants (see Section 5.6), which are regularized (see Section 5.8). The remaining component of the solutions is computed by a 1D rootfinder (see Section 5.7). Finally, the accuracy of the computed solutions is improved by a local refinement procedure (see Section 5.8). Figure 5.1 summarizes the key steps of the algorithm, and the rest of this chapter explains each one of them in turn.

The algorithm is implemented in the `roots(f,g)` command in Chebfun2 [148], and works in double precision with floating-point arithmetic. It does not require higher precision or symbolic manipulation. The algorithm is designed to be a backward stable 2D rootfinder for (5.1) under the assumptions that the solutions are simple.

5.3 Polynomialization

Our first step replaces f and g in (5.1) by chebfun2 approximants[3] p and q that are constructed to satisfy $\|f-p\|_\infty \le \epsilon\|f\|_\infty$ and $\|g-q\|_\infty \le \epsilon\|g\|_\infty$ (see Chapter 2). The underlying low rank structure of a chebfun2 that was so readily exploited in Chapter 2 is ignored, and p and q are considered in tensor product form, i.e., for some integers m_p, m_q, n_p, and n_q,

$$p(x,y) = \sum_{j=0}^{m_p}\sum_{i=0}^{n_p} P_{ij}T_i(y)T_j(x), \qquad q(x,y) = \sum_{j=0}^{m_q}\sum_{i=0}^{n_q} Q_{ij}T_i(y)T_j(x), \tag{5.3}$$

where $P \in \mathbb{C}^{(n_p+1)\times(m_p+1)}$ and $Q \in \mathbb{C}^{(n_q+1)\times(m_q+1)}$ are matrices containing the Chebyshev expansion coefficients of p and q, respectively.

We proceed to solve the polynomial system of approximants,

$$\begin{pmatrix} p(x,y) \\ q(x,y) \end{pmatrix} = 0, \qquad (x,y) \in [-1,1]^2, \tag{5.4}$$

under the expectation that its solutions approximate those of the original problem in (5.1). It can be shown with Taylor expansions that if $(x_*, y_*) \in [-1,1]^2$ is a simple solution to $f = g = 0$ and $(\tilde{x}, \tilde{y})$ is a solution to $p(\tilde{x}, \tilde{y}) = q(\tilde{x}, \tilde{y}) = 0$ that is sufficiently close to (x_*, y_*) to allow higher order terms to be ignored, then

$$\left\| \begin{pmatrix} \tilde{x} - x_* \\ \tilde{y} - y_* \end{pmatrix} \right\|_\infty \le \mathcal{O}\left(u \max\left(\|f\|_\infty, \|g\|_\infty\right) \|J(x_*, y_*)^{-1}\|_\infty \right), \quad J = \begin{bmatrix} \partial f/\partial x & \partial f/\partial y \\ \partial g/\partial x & \partial g/\partial y \end{bmatrix}, \tag{5.5}$$

where u is unit machine roundoff. We conclude that in many circumstances the solutions to (5.1) are only slightly perturbed (in the absolute sense) by replacing f and g by accurate approximants p and q. However, if f or g have widely varying magnitudes, then some of the solutions to (5.1) may not be well-approximated by those of (5.4) (see Section 5.9).

In the rootfinding literature it is standard to begin with a system of prescribed low degree (≤ 30) polynomials, usually expressed in the monomial basis, and then to use an algorithm with a complexity of the maximum polynomial degree to the power of 6 [138] or worse [30, 111]. Instead, we start with a system of functions, rather than polynomials, and our first step constructs a nearby polynomial rootfinding problem.

[3] A chebfun2 approximant is also a bivariate polynomial approximant.

Therefore, we have the freedom to select the polynomial basis in which to represent the polynomials, and we choose the Chebyshev polynomial basis for numerical reasons. As a further consequence, rootfinding problems are routinely generated and solved that involve polynomials of high degree (≥ 100).

For the remainder of this chapter we assume that the functions in the original problem (5.1) have been replaced by chebfun2 approximants p and q.

5.3.1 The maximum number of solutions

Replacing f and g by polynomials is not only numerically convenient, it also ensures the rootfinding problem is computationally tractable as polynomial systems have, generically, finitely many solutions. The maximum number of finite solutions that a polynomial system can have depends on the polynomial degrees. In one variable the number of complex solutions equals the polynomial degree. In two or more variables an analogous theorem is known as Bézout's Theorem [91, Ch. 3].

In two variables, the standard formulation of Bézout's Theorem applies to systems of polynomials with *total degree*[4] m and n. The polynomials p and q in (5.3) are of total degree $m_p + n_p$ and $m_q + n_q$, respectively, and a direct application of Bézout's Theorem shows that, generically, the number of solutions to $p = q = 0$ is at most $(m_p + n_p)(m_q + n_q)$. In our case, one can improve this bound to show that (5.4) has, generically, at most $m_p n_q + m_q n_p$ solutions (cf. (5.2)).

Theorem 5.1 (Bézout bound)**.** *Let $p(x, y)$ and $q(x, y)$ be bivariate polynomials with complex coefficients of degrees (m_p, n_p) and (m_q, n_q), respectively. The solution set to $p = q = 0$ is either positive dimensional (containing a curve) or contains at most*

$$m_p n_q + n_p m_q \tag{5.6}$$

isolated solutions.

Proof. Apply Bernstein's Theorem to obtain a mixed volume bound (see [137, p. 139–143]). □

Theorem 5.1 can be seen as a 2D generalization of the Fundamental Theorem of Algebra. There are variants of Theorem 5.1 where the number of solutions is exactly $m_p n_q + n_p m_q$ and to make such a theorem precise one must work in the

[4] A bivariate polynomial is of total degree n if it can be written in the form $\sum_{i+j \leq n} \alpha_{ij} y^i x^j$.

complex projective space $\mathbb{CP}^2$ and be careful about multiple solutions and solutions at infinity [91, Thm. 3.1]. These questions are within the broad subject of algebraic geometry, which quantifies the word "generically" and more intricate structures of the solution sets. An interested reader may like to begin with [137, Chap. 3].

5.4 Existing bivariate rootfinding algorithms

There are many existing numerical algorithms for bivariate rootfinding based on, for example, resultants [21, 44, 89, 101], homotopy continuation [8, 137], and two-parameter eigenvalue problems [3, 111]. In addition, there are symbolic algorithms such as those that use Gröbner bases [30], which are extremely useful for small degree polynomial systems but generally have a high complexity. Here, we briefly review algorithms for bivariate rootfinding. The algorithm described in this chapter is based on the hidden variable resultant method for polynomials expressed in the Chebyshev basis.

5.4.1 Resultant methods

The hidden variable resultant method is based on selecting one variable, say y, and writing bivariate polynomials p and q of degrees (m_p, n_p) and (m_q, n_q) as polynomials in x with coefficients in that are polynomials in y, i.e.,

$$p(x,y) = p_y(x) = \sum_{j=0}^{m_p} \alpha_j(y) x^j, \qquad q(x,y) = q_y(x) = \sum_{j=0}^{m_q} \beta_j(y) x^j. \tag{5.7}$$

Here, the polynomials p and q in (5.7) are represented in the monomial basis because that is how it is commonly given in the literature [41, 49]; however, representing the polynomial in the Chebyshev basis is numerically stable when working on real intervals [159]. For this reason, we always represent the polynomials p and q using the Chebyshev polynomial basis.

Generically, two univariate polynomials do not have a common root, and certain matrices, known as *resultant matrices* (with entries depending on the coefficients of the polynomials), can be constructed that are singular if and only if a common root exists. In particular, the two polynomials $p_y(x)$ and $q_y(x)$ in (5.7), thought of as univariate in x, have a common zero if and only if a resultant matrix is singular [6].

Therefore, the y-values of the solutions to $p = q = 0$ can be computed by finding the y-values such that a resultant matrix is singular.

There are many different types of resultant matrices such as Sylvester [41], Bézout [19], and others [6, 49, 87]. These resultant matrices are usually constructed for polynomials expressed in the monomial basis [21, 101], but they can also be constructed for polynomials expressed in other bases [20]. See Appendix C for the construction of the Bézout resultant matrix for polynomials expressed in the Chebyshev basis.

Finding the y-values such that the resultant matrix is singular is equivalent to a polynomial eigenvalue problem, and many techniques exist for solving such problems, including methods based on contour integrals [2, 18], Newton-type methods, inverse iteration methods [106], the Ehrlich–Aberth method [22], and the standard approach of solving via linearization [58]. We use linearization and replace polynomial eigenvalue problems by generalized eigenvalue problems (see Section 5.6) that have the same eigenvalues and Jordan structure [58].

A related approach is the u-resultant method, which starts by introducing a dummy variable to make the polynomials homogeneous. The hidden variable resultant method is then applied to the new polynomial system of three variables selecting the dummy variable first. This ensures that the x- and y-variables are treated in the same way. Unfortunately, making a polynomial homogeneous when expressed in the Chebyshev basis is numerically unstable.

Some resultant methods, such as the rootfinder in Maple™, first apply a coordinate transformation to ensure that two solutions do not share the same y-value. Such transforms are unnecessary in our algorithm (see Figure 5.5). Other changes of variables can be applied, such as $x = (z + \omega)/2$ and $y = (z - \omega)/2$ [138], but we have found that such changes of variables give little benefit in practice.

5.4.2 Contouring algorithms

Contouring algorithms such as marching squares and marching triangles [83, 96] are employed in computer graphics to generate zero level curves of bivariate functions. These contouring algorithms can be used to solve (5.1) very efficiently. Prior to the development of the algorithm discussed in this chapter, the `roots(f,g)` command in Chebfun2 exclusively employed such a contouring approach [148], where the zero level curves of f and g were computed separately using the MATLAB command[5]

[5]The MATLAB command `contourc` employs an algorithm based on marching squares.

`contourc`, and then the intersections of these zero level curves were used as initial guesses for a Newton iteration.

Contouring algorithms suffer from several drawbacks:

1. The level curves of f and g may not be smooth even for very low degree polynomials, for example, $f(x, y) = y^2 - x^3$.

2. The number of disconnected components of the level curves of f and g can be potentially quite large. The maximum number is given by Harnack's Curve Theorem [79].

3. Problems are caused if the zero level curves of f or g are self-intersecting, leading to the potential for missed solutions.

4. The zero level curves must be discretized and therefore the algorithm requires a fine tuning of parameters to balance efficiency and reliability.

For many practical applications solving (5.1) via a contouring algorithm may be an adequate approach, but not always.

The `roots(f,g)` command in Chebfun2 implements both the algorithm based on a contouring algorithm and the approach described in this chapter. An optional parameter can be supplied by a user to select one of the algorithms. We have decided to keep the contouring approach as an option in Chebfun2 because it can sometimes run faster, but we treat its computed solutions with suspicion.

5.4.3 Other numerical methods

Homotopy continuation methods [137] have the simple underlying idea of solving an initial "easy" polynomial system that can be continuously deformed into the desired "harder" polynomial system. Along the way several polynomial systems are solved with the current solution set being used as an initial guess for the next. These methods have received significant attention in the literature and are a purely numerical approach that can solve multivariate rootfinding problems [8, 137]. A particularly impressive software package for homotopy methods is Bertini [8].

The two-parameter eigenvalue approach constructs polynomial interpolants to f and g and then rewrites $p(x, y) = q(x, y) = 0$ as a two-parameter eigenvalue problem [3],

$$A_1 v = xB_1 v + yC_1 v, \qquad A_2 w = xB_2 w + yC_2 w.$$

This approach has advantages because the two-parameter eigenvalue problem can be solved with the QZ algorithm [84]; however, the construction of the matrices A_i, B_i, C_i for $i = 1, 2$ currently requires the solution of a multivariate polynomial rootfinding problem itself [121]. Alternatively, matrices of much larger size can be constructed using a generalized companion form, but this is quite inefficient [111].

5.5 Recursive subdivision of the domain

Domain subdivision is a simple and powerful technique for solving a high degree rootfinding problem via the solution of several subproblems of low degree[6]. It is a useful step when m_p, n_p, m_q, and n_q are greater than 20 and absolutely essential when they are in the hundreds or thousands.

The methodology we employ is a 2D analogue of Boyd's 1D subdivision technique [27], as utilized by the `roots` command in Chebfun [161]. First, $[-1, 1]^2$ is subdivided from top to bottom if $\max(m_p, m_q) > 16$, i.e., the degrees of p or q in the x-variable are greater than 16, and subdivided from left and right if $\max(n_p, n_q) > 16$, where the number 16 was determined by optimizing the computational time of the overall algorithm on a set of test problems. The process terminates if no subdivision is required. Otherwise, new continuous piecewise approximants to p and q are constructed that are polynomial on each subdomain. The process is applied recursively with further subdivision if a domain contains a polynomial of degree more than 16 in the x- or y-variable. As an example, Figure 5.2 shows how $[-1, 1]^2$ is subdivided for $\sin((x-1/10)y)\cos(1/(x+(y-9/10)+5)) = (y-1/10)\cos((x+(y+9/10)^2/4)) = 0$.

When subdividing, the domain is not exactly bisected, but instead divided asymmetrically to avoid splitting at a solution to $p = q = 0$. Two small arbitrary constants[7] r_x and r_y have been picked so that when $[-1, 1]^2$ is subdivided from top to bottom the domains $[-1, r_x] \times [-1, 1]$ and $[r_x, 1] \times [-1, 1]$ are formed, and when subdivided from left to right the domains $[-1, 1] \times [-1, r_y]$ and $[-1, 1] \times [r_y, 1]$ are formed. This is to avoid accidentally subdividing at a solution since, for example, it is common to have a solution with $x = 0$, but highly contrived to have one with $x = r_x$.

[6]Domain subdivision has similarities to the Interval Projected Polyhedron algorithm [110, 134] except the focus here is on reducing the computational cost of rootfinding rather than determining subregions that are guaranteed to contain the solutions.

[7]We take $r_x \approx -0.004$ and $r_y \approx -0.0005$. The full 16 digits for r_x and r_y are a guarded secret as this knowledge leads to the construction of a example where solutions are double counted. There is no significance to the values of r_x and r_y except that they are nonzero, small, and arbitrary.

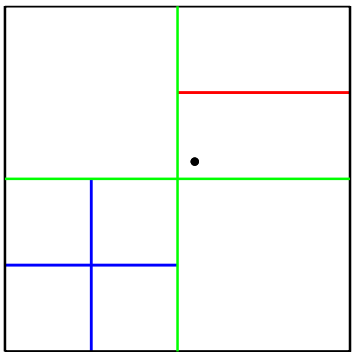


Figure 5.2: Subdivision of $[-1,1]^2$ selected by Chebfun2 when $f = \sin((x - 1/10)y)\cos(1/(x + (y - 9/10) + 5))$ and $g = (y - 1/10)\cos((x + (y + 9/10)^2/4))$. The blue and red lines represent subdivisions used to reduce the degrees of the piecewise polynomial approximant to f and g, respectively, and the green lines are subdivisions used to reduce the degrees of both. The black dot is the only solution to $f = g = 0$ in $[-1,1]^2$.

For domain subdivision to reduce the overall computational cost of rootfinding, it must significantly decrease the degrees of the polynomial approximants. Suppose, for simplicity, that all the degrees of p and q are equal to n, i.e., $n = m_p = n_p = m_q = m_q$, and that on average a subdivision reduces the degrees by a factor $0 < \tau \leq 1$. Then, the number of subdivisions required is the largest integer, k, that satisfies $n\tau^k \leq d = 16$. Hence, $k \approx (\log d - \log n)/(\log \tau)$. One subdivision splits a domain into four and thus k levels of subdivision creates $\mathcal{O}(4^k)$ low degree rootfinding subproblems. Each subproblem can be solved in a computational time independent of n and hence, the resulting cost of our algorithm is

$$\mathcal{O}\left(4^k\right) = \mathcal{O}\left(4^{\frac{\log d - \log n}{\log \tau}}\right) = \mathcal{O}\left(4^{-\frac{\log n}{\log \tau}}\right) = \mathcal{O}\left(n^{-\frac{\log 4}{\log \tau}}\right).$$

Figure 5.3 shows the exponent $-\log 4/\log \tau$ as a function of τ. When $\tau \leq 0.5$ the overall computational cost can be as low as $\mathcal{O}(n^2)$ and is dominated by the initial construction of p and q. When $\tau = 0.79$ the cost is as high as $\mathcal{O}(n^6)$, and if $\tau > 0.79$ subdivision increases the computational cost. For many practical problems we observe that $\tau \approx \sqrt{2}/2 \approx 0.707$, which leads to a computational cost of $\mathcal{O}(n^4)$. At first sight these computational costs may seem unacceptably large, but by Theorem 5.1 there can be as many as $2n^2$ solutions to (5.4) when $n = m_p = n_p = m_q = m_q$ and hence, in that case just to evaluate p and q at all the solutions requires $\mathcal{O}(n^4)$ operations.

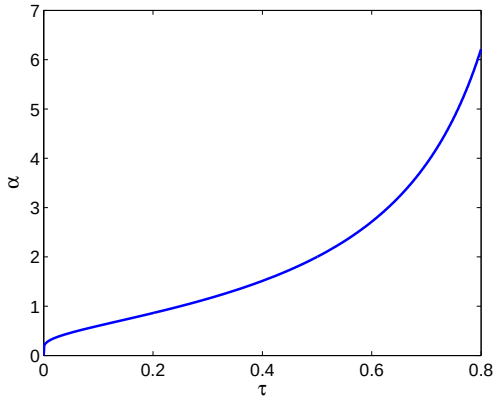


Figure 5.3: Computational cost required after subdivision is $\mathcal{O}(n^\alpha)$, where $\alpha = -\log 4/\log \tau$ and τ is the average reduction in polynomial degrees per subdivision.

The parameter τ can take any value in $(0, 1]$ as the following four examples show:

1. Consider $f(x, y) = c_f(y)r_f(x)$, where c_f and r_f are highly oscillatory. By the Shannon–Nyquist sampling theorem [132], c_f and r_f require at least 2 samples per wavelength to be resolved. A subdivision halves the number of oscillations in the x- and y-variable and hence $\tau \approx 0.5$.

2. Consider the contrived example $f(x, y) = |x - r_x||y - r_y|$, which is of high numerical degree on $[-1, 1]^2$. However, after subdividing in both directions the polynomial approximants are piecewise linear and hence $\tau \approx 0$.

3. Bivariate polynomials of low degree with non-decaying coefficients tend to have $\tau \approx 1$ (see the Hadamard Example in Section 5.10.4).

The parameter τ is closely related to the approximation theory of piecewise polynomials of two variables, which is analyzed in the *hp*-FEM (finite element method) literature [70]. Here, instead of considering the approximation power of piecewise polynomials we are looking at the number of degrees of freedom required to maintain a fixed approximation error.

Figure 5.4 shows the computational time for solving $\sin(\omega(x + y)) = \cos(\omega(x - y)) = 0$ for $1 \leq \omega \leq 50$, with and without subdivision. In our implementation we keep a running estimate of τ and stop the subdivision process if $\tau > .79$ for three consecutive subdivisions.

Once $[-1, 1]^2$ has been sufficiently subdivided we proceed to solve the rootfinding problem on each subdomain. Usually these rootfinding problems involve polynomials of degree at most 16 in both variables, unless τ was estimated to be more than 0.79.

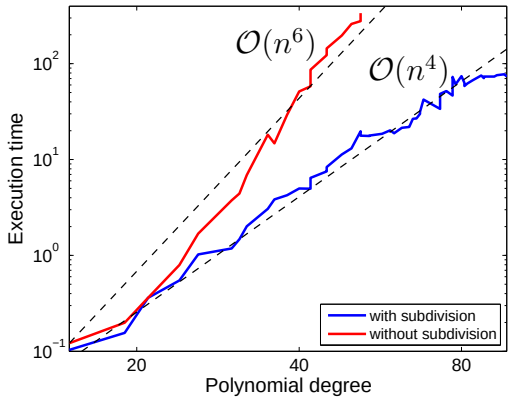


Figure 5.4: Execution time for the `roots(f,g)` command in Chebfun2 to solve $\sin(\omega(x+y)) = \cos(\omega(x-y)) = 0$ for $1 \leq \omega \leq 50$, with and without subdivision. Here, subdivision reduces the cost from $\mathcal{O}(n^6)$ to $\mathcal{O}(n^4)$.

We expect, and it is almost always the case, that the union of solutions to all these subproblems approximate those of the original problem (5.1). Rootfinding problems where this fails to hold usually involve a function with a large dynamic range (see Section 5.9).

5.6 A resultant method with Bézout resultants

After domain subdivision we are left with low degree polynomial bivariate rootfinding problems. In this section we describe the resultant method we use to solve these polynomial systems. Without loss of generality we assume the solutions are desired in $[-1,1]^2$; otherwise, an affine transformation can be used to map the rectangular domain to $[-1,1]^2$.

The resultant method we use is called a *hidden variable resultant method* [137, p. 73], which selects a variable, say y, and rewrites the polynomials $p(x,y)$ and $q(x,y)$ in (5.4) as univariate polynomials in x with coefficients that depend on y, i.e.,

$$p(x,y) = p_y(x) = \sum_{j=0}^{m_p} \alpha_j(y) T_j(x), \qquad q(x,y) = q_y(x) = \sum_{j=0}^{m_q} \beta_j(y) T_j(x).$$

The y-values for the solutions to $p = q = 0$ can then be determined by finding the common roots of p_y and q_y, which in turn can be computed from a resultant matrix. We use a (Chebyshev–)Bézout resultant matrix, as opposed to a Sylvester resultant

matrix, because it is symmetric, which makes it more amenable to theoretical study (see, for instance, [113, Thm. 1]).

5.6.1 Bézout resultants for finding common roots

The Bézout matrix is a special square matrix that is usually associated with two univariate polynomials expressed in the monomial basis [32]. It is useful because its determinant is a nonzero constant multiple of the product of the pairwise differences between the roots of the two polynomials, and hence it is singular if and only if the two polynomials have a common root [56, p. 401].

Bézout matrices have an analogue for polynomials expressed in the Chebyshev polynomial basis, which we call *Chebyshev–Bézout matrices*. Let p and q be two Chebyshev series of degree m and n (with real or complex coefficients) given by

$$p(x) = \sum_{i=0}^{m} \alpha_i T_i(x), \qquad q(x) = \sum_{i=0}^{n} \beta_i T_i(x), \qquad x \in [-1, 1].$$

Then, the Chebyshev–Bézout matrix of size $\max(m, n) \times \max(m, n)$ associated with p and q is defined as $B = (b_{ij})_{1 \le i,j \le \max(m,n)}$, where the entries of B satisfy

$$\frac{p(s)q(t) - p(t)q(s)}{s - t} = \sum_{i,j=1}^{\max(m,n)} b_{ij} T_{i-1}(t) T_{j-1}(s). \tag{5.8}$$

The defining relation in (5.8) differs from the standard definition for Bézout matrices in that monomials have been replaced by Chebyshev polynomials. A description of how to construct Bézout matrices numerically is given in Appendix C.

The Chebyshev–Bézout matrix can be used to compute the common roots of two univariate polynomials.

Theorem 5.2 (Resultant theorem)**.** *Let p and q be two polynomials of exact degree m and n, respectively. Then, the Chebyshev–Bézout matrix defined in* (5.8) *has the following properties:*

1. *The matrix B is symmetric, i.e., $b_{ij} = b_{ji}$.*
2. *If B is a nonzero matrix, then $x_0 \in [-1, 1]$ is a common root of p and q if and only if $\big[T_0(x_0), \ldots, T_{\max(m,n)-1}(x_0)\big]^T$ is an eigenvector of B corresponding to a zero eigenvalue.*

3. The matrix B is singular if and only if p and q have a finite common root.

Proof. The bivariate function $k(s,t) = (p(s)q(t) - p(t)q(s))/(s-t)$ satisfies $k(s,t) = k(t,s)$ so its coefficient matrix, B, is symmetric.

For the second statement, note that (5.8) can be written as

$$B \begin{bmatrix} T_0(s) \\ T_1(s) \\ \vdots \\ T_{\max(m,n)-1}(s) \end{bmatrix} = \begin{bmatrix} \frac{1}{\pi}\int_{-1}^{1} \frac{k(s,t)T_0(t)}{\sqrt{1-t^2}}\,dt \\ \frac{2}{\pi}\int_{-1}^{1} \frac{k(s,t)T_1(t)}{\sqrt{1-t^2}}\,dt \\ \vdots \\ \frac{2}{\pi}\int_{-1}^{1} \frac{k(s,t)T_{\max(m,n)-1}(t)}{\sqrt{1-t^2}}\,dt \end{bmatrix},$$

by using the integral definition of Chebyshev expansion coefficients (see (1.4)). Thus, if $\left[T_0(x_0), \ldots, T_{\max(m,n)-1}(x_0)\right]^T$ with $x_0 \in [-1,1]$ is an eigenvector of B corresponding to a zero eigenvalue, then $\int_{-1}^{1} k(x_0,t)T_j(t)/\sqrt{1-t^2}\,dt = 0$ for $0 \le j \le \max(m,n)-1$. Hence, $k(x_0,\cdot) \equiv 0$ and $p(x_0)q(t) \equiv p(t)q(x_0)$. If $p(x_0) \neq 0$ or $q(x_0) \neq 0$ then p and q are proportional, leading to $B = 0$. By assumption B is a nonzero matrix so $p(x_0) = q(x_0) = 0$ and x_0 is a common root of p and q. Conversely, if $x_0 \in [-1,1]$ is a common root then $k(x_0,\cdot) \equiv 0$ and hence $\left[T_0(x_0), \ldots, T_{\max(m,n)-1}(x_0)\right]^T$ is an eigenvector of B corresponding to a zero eigenvalue.

For the last statement, let X be the $\max(m,n) \times \max(m,n)$ upper-triangular matrix that converts Chebyshev coefficients to monomial coefficients (the jth column of X contains the first $\max(m,n)$ monomial coefficients of T_{j-1}). Then, X is invertible and XBX^{-1} is the standard Bézout matrix associated with p and q expressed in the monomial basis. Therefore, $\det(B) = \det(XBX^{-1})$ is a nonzero constant multiple of the product of the pairwise differences between the roots of p and q [56, p. 401] and the result follows. □

Theorem 5.2 tells us that the invertibility of the Chebyshev–Bézout matrix in (5.8) can be used as a test for common roots of two univariate polynomials expressed in the Chebyshev basis. This fact is the essential ingredient for the resultant method we employ for bivariate rootfinding.

5.6.2 Bézout resultants for bivariate rootfinding

We now describe the hidden variable resultant method for solving $p(x,y) = q(x,y) = 0$, which is derived from the following observation: If $(x_0, y_0) \in [-1,1]^2$ is a solution

to $p = q = 0$, then the univariate polynomials $p(\cdot, y_0)$ and $q(\cdot, y_0)$ have a common root (at x_0) and, by Theorem 5.2, the associated Chebyshev–Bézout matrix is singular. Conversely, if the Chebyshev–Bézout matrix associated with $p(\cdot, y_0)$ and $q(\cdot, y_0)$ is singular, then for some finite x_0 we have[8] $p(x_0, y_0) = q(x_0, y_0) = 0$. This simple observation means we can use the Chebyshev–Bézout matrix to compute the y-values of the solutions to (5.4).

More formally, we can define a function[9] $B(y) : [-1, 1] \to \mathbb{C}^{N\times N}$, where $N = \max(m_p, m_q)$, which takes $y \in [-1, 1]$ to the Chebyshev–Bézout matrix associated with $p(\cdot, y)$ and $q(\cdot, y)$. By Theorem 5.2, if there is a finite x_0 (not necessarily in $[-1, 1]$) such that $p(x_0, y_0) = q(x_0, y_0) = 0$, then $\det(B(y_0)) = 0$. Hence, all the y-values of the solutions satisfy $\det(B(y)) = 0$ (and all the finite roots of $\det(B(y)) = 0$ are such that p_y and q_y have a common root). Thus, the idea is to solve $\det(B(y)) = 0$ and then substitute these y-values into p and q to find the corresponding x-values of the solutions (see Section 5.7).

Since p and q are bivariate polynomials of degrees (m_p, n_p) and (m_q, n_q), $B(y) = (b_{ij}(y))_{1\le i,j\le N}$ satisfies (see (5.8))

$$\frac{p(s, y)q(t, y) - p(t, y)q(s, y)}{s - t} = \sum_{i,j=1}^{N} b_{ij}(y)T_{i-1}(t)T_{j-1}(s).$$

Hence, for each fixed $y \in [-1, 1]$, $B(y)$ is an $N \times N$ matrix and each entry of $B(y)$ is a polynomial of degree at most $M = n_p + n_q$ in y. Matrices with polynomial entries, such as $B(y)$, are known as *matrix polynomials* (see, for example, [58]). We say that $B(y)$ is a matrix polynomial of size $N \times N$ and degree M. The problem of solving $\det(B(y)) = 0$ is an example of a polynomial eigenvalue problem.

The matrix polynomial $B(y)$ can be represented in the Chebyshev basis with matrix coefficients,

$$B(y) = \sum_{i=0}^{M} A_i T_i(y), \qquad A_i \in \mathbb{C}^{N\times N}, \qquad y \in [-1, 1]. \tag{5.9}$$

The matrices A_i can be constructed by calculating the Chebyshev expansions of each

[8]Here, we assume that x_0 is finite, but the Chebyshev–Bézout matrix associated with $p(\cdot, y_0)$ and $q(\cdot, y_0)$ is also singular if x_0 is infinite, i.e., the leading coefficients of both $p(\cdot, y_0)$ and $q(\cdot, y_0)$ are zero.

[9]We always write the function $B(y)$ with its argument in parentheses to distinguish it from a Bézout matrix. For each fixed $y \in [-1, 1]$, $B(y)$ is a Chebyshev–Bézout matrix.

entry of $B(y)$.

By assumption the solutions to $p = q = 0$ are isolated so $\det(B(y)) \not\equiv 0$, i.e., $B(y)$ is a *regular* matrix polynomial. This means that $\det(B(y))$ is a nonzero scalar polynomial and its roots are precisely the finite eigenvalues of $B(y)$ [98]. The eigenvalues of $B(y)$ can be computed by solving the following $MN \times MN$ generalized eigenvalue problem (GEP) [149]:

$$\frac{1}{2}\begin{bmatrix} -A_{M-1} & I_N - A_{M-2} & -A_{M-3} & \cdots & -A_0 \\ I_N & 0 & I_N & & \\ & \ddots & \ddots & \ddots & \\ & & I_N & 0 & I_N \\ & & & 2I_N & 0 \end{bmatrix} v = \lambda \begin{bmatrix} A_M & & & & \\ & I_N & & & \\ & & \ddots & & \\ & & & I_N & \\ & & & & I_N \end{bmatrix} v. \tag{5.10}$$

This GEP is a matrix analogue of the colleague matrix for scalar polynomials [63], and the Chebyshev analogue of the companion matrix for matrix polynomials [98].

The GEP can be solved numerically by the `eig` command in MATLAB, involving a computational cost of about $\mathcal{O}(M^3N^3)$ operations. If $n = m_p = n_p = m_q = n_q$ then $\mathcal{O}(M^3N^3) = \mathcal{O}(n^6)$ and hence, the cost of solving (5.10) becomes prohibitive when $n \geq 30$. To combat this we use domain subdivision (see Section 5.5) to reduce the polynomial degrees in the resultant method, which usually means that the constructed GEPs are not larger than 512×512.

To summarize, the y-values of the solutions are computed by constructing the matrix polynomial $B(y)$ that is singular at those values. Then, the equivalent problem of $\det(B(y)) = 0$ is handled by solving a GEP. Any solutions to the GEP that lie outside of $[-1, 1]$ are filtered out. Finally, the x-values of the solutions are computed (see Section 5.7).

The methodology described above also works when the roles of x and y are interchanged, though this can alter the computational cost of the resulting algorithm. As we have seen, if the y-values are solved for first, then the resulting GEP is of size $\max(m_p, m_q)(n_p + n_q)$, while if the x-values are solved for first then the size is $\max(n_p, n_q)(m_p + m_q)$. These two sizes and resulting computational costs are equal when $n = m_p = n_p = m_q = n_q$; however, the sizes can differ by a factor as large as 2, resulting in a discrepancy as large as 8 in the computational cost. Table 5.1 summarizes the size of the GEP based on whether the x- or y-values are solved for first. In the algorithm we minimize the size of the GEP by solving for the y-values first if $\max(m_p, m_q)(n_p + n_q) \leq \max(n_p, n_q)(m_p + m_q)$; otherwise, we solve for the x-values first.

Bézout resultant	Size of A_i in (5.9)	Degree	Size of GEP in (5.10)
y first	$\max(m_p, m_q)$	$n_p + n_q$	$\max(m_p, m_q)(n_p + n_q)$
x first	$\max(n_p, n_q)$	$m_p + m_q$	$\max(n_p, n_q)(m_p + m_q)$

Table 5.1: Sizes and degrees of the matrix polynomials constructed by the resultant method. The size of the resulting GEP depends on whether the x- or y-values are solved for first. We solve for the y-values first if $\max(n_p, n_q)(m_p + m_q) \leq \max(m_p, m_q)(n_p + n_q)$; otherwise, the x-values are first.

Regardless of which variable is solved for first the GEP is at least of size $m_p n_q + n_p m_q$. However, the number of eigenvalues of the GEP can occasionally be more than the bound in Theorem 5.1. Since the polynomial system has at most $m_p n_q + n_p m_q$ solutions, the resultant method can generate extra solutions that have nothing to do with the rootfinding problem; however, Theorem 5.2 tells us that these extra solutions cannot be finite. Therefore, we have an interesting algebraic phenomenon where the resultant method can generate extra solutions, but these are always at infinity and are usually harmless.[10]

In subsequent sections we assume that the y-values are solved for first.

5.7 Employing a 1D rootfinder

At this stage we assume that the y-values of the solutions to (5.4) have been computed and to obtain the full set of solutions the corresponding x-values are required.

In principle for each y-value of a solution, say $y_0 \in [-1, 1]$, the eigenvectors of Bézout matrix $B(y_0)$ corresponding to zero eigenvalues can be used to compute the corresponding x-values (see Theorem 5.2). However, in practice, the eigenvectors of $B(y_0)$ are not always computed with high accuracy and instead, we employ a 1D rootfinder.

For each y-value of a solution, $y_0 \in [-1, 1]$, two sets of roots for the univariate polynomials $p(\cdot, y_0)$ and $q(\cdot, y_0)$ are computed separately using the `roots` command in Chebfun. Then, we find the roots that are shared by verifying that both $|p|$ and $|q|$ are less than $\mathcal{O}(u^{1/2})$, where u is unit machine roundoff, discarding those that are not. Afterwards, we merge any x-values that are closer than a distance of $\mathcal{O}(u)$ apart by averaging, which prevents double counting a solution due to rounding errors.

[10]To ensure that the spurious infinite solutions are harmless one must also assume that the infinite solutions are simple so that the eigenvalues of $B(y)$ are semisimple. More details are given in [113].

5.8 Further implementation details

A few implementation details remain. Without loss of generality, we assume in this section that the polynomials p and q have been scaled so that $\|p\|_\infty = \|q\|_\infty = 1$. (Otherwise, some of the tolerances need to be scaled by $\|p\|_\infty$ and $\|q\|_\infty$.)

5.8.1 Regularization

In exact arithmetic the polynomial $\det(B(y))$ is zero at precisely the y-values of the solutions to $p = q = 0$, including those outside of $[-1, 1]$. However, $B(y)$ can be numerically singular, i.e., $\|B(y)\|_2\|B(y)^{-1}\|_2 > 10^{16}$, for many other values of y. As a consequence of numerical rounding and this extreme ill-conditioning, an eigenvalue solver applied to the GEP in (5.10) can compute spurious eigenvalues anywhere in the complex plane [4, Sec 8.7.4], and these can cause all the eigenvalues to be computed inaccurately [119, Ch. 13]. As a remedy we apply a regularization step to $B(y)$.

First, we partition $B(y)$ into the following four parts:

$$B(y) = \begin{bmatrix} B_1(y) & E(y)^T \\ E(y) & B_0(y) \end{bmatrix},$$

where $B_0(y)$ and $E(y)$ are $k \times k$ and $k \times (N-k)$, respectively. Here, k is selected to be the smallest integer so that both $\|B_0(y)\|_2 = \mathcal{O}(u)$ and $\|E(y)\|_2 = \mathcal{O}(u^{1/2})$ for all $y \in [-1, 1]$. If $B(y)$ is real, then $B(y)$ is Hermitian and by Weyl's Theorem [166] the eigenvalues of

$$\tilde{B}(y_0) = \begin{bmatrix} B_1(y_0) & E(y_0)^T \\ E(y_0) & 0 \end{bmatrix},$$

for any $y_0 \in [-1, 1]$ are within $\mathcal{O}(u)$ of those of $B(y_0)$. In fact, as argued in [113, Sec. 6.2], the finite eigenvalues of $\tilde{B}(y_0)$ that are small in magnitude are closely approximated by the eigenvalues of $B_1(y_0)$ if the entries of $B(y_0)$ decay sufficiently quickly as the column and row indices increase. (This is usually the case for the polynomials derived from interpolation of smooth functions.) Therefore, the small finite eigenvalues of $B(y_0)$ are well-approximated by those of $B_1(y_0)$. We continue to observe this when $B(y)$ is complex valued.

Instead of computing the eigenvalues of $B(y)$ we find the eigenvalues of $B_1(y)$, setting up the GEP in (5.10) as before. This smaller Bézout matrix polynomial, $B_1(y)$, has better conditioned eigenvalues and often the y-values in $[-1, 1]$ are computed much more accurately.

This regularization step is equivalent to removing the high degree terms in (5.8) that are negligible and has the consequence of removing the eigenvalues of $B(y)$ that have a large absolute magnitude. This step is primarily for improving the numerical stability of the resultant method, but it has an secondary efficiency benefit because it reduces the size of $B(y)$ and the corresponding GEP in (5.10).

5.8.2 Local refinement

Unfortunately, the condition number of a root of $\det(B(y))$ can be as large as the square of the condition number of the corresponding solution to the original problem (5.4) [113, Thm. 1]. Therefore, the computed y-values may be inaccurate, and some sort of local refinement is necessary to improve the accuracy of the solutions. Our rootfinding algorithm thus splits into two parts. First, an initial global resultant method (see Section 5.6) is used to obtain estimates of the solutions with an error that is potentially larger than desired. Then, a local refinement strategy is applied to clean up these estimates as necessary.

A simple option is to use a single iteration of Newton's method to polish the computed solutions, i.e., if $(\tilde{x}, \tilde{y})$ is a computed solution, then update by the procedure

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} - J(\tilde{x}, \tilde{y})^{-1} \begin{pmatrix} p(\tilde{x}, \tilde{y}) \\ q(\tilde{x}, \tilde{y}) \end{pmatrix},$$

where $J(\tilde{x}, \tilde{y})$ is the Jacobian matrix (see (5.5)). Often a single iteration is enough to obtain a solution that is as accurate as can be expected from a backward stable algorithm, though this is not guaranteed [147]. This strategy does have some difficulty when two simple solutions are numerically close as two initial guesses can converge to the same solution, and this can lead to simple solutions being missed.

Instead, we perform local refinement by rerunning the resultant method in a "zoomed-in" region whenever the 2-norm of the Jacobian is small, as this is the case where the Bézout resultant method increases the condition number [113, Thm. 1].

This strategy involves resampling the polynomials p and q in a region close to a computed solution, and repeating our Bézout resultant method in this small region. Usually, these local refinement regions are so small that the polynomial systems are of very low degree, so the extra cost of this vital step is negligible.

5.8.3 Solutions near the boundary of the domain

Some care is required to avoid missing solutions that lie on or near the boundary of $[-1,1]^2$, as they can be easily perturbed to lie outside. Similarly, real solutions can be perturbed to complex solutions with negligible imaginary parts. We deal with the first of these problems by looking for solutions in a slightly larger domain $[-1-u^{1/2}, 1+u^{1/2}]^2$, and any computed solution that is within a distance of $10u$ to $[-1,1]^2$ is perturbed onto the boundary of $[-1,1]^2$. To deal with the second we keep the eigenvalues of the GEP[11] in (5.10) that have a real part in $[-1,1]$ and an imaginary part of size at most $u^{1/2}$. At the end we perturb a complex solution onto $[-1,1]^2$ if it has an imaginary part of size less than $10u$, by setting the imaginary part to 0. The various parameters, that are envitable when designing a numerical rootfinder, are subject to change but are documented in the MATLAB code [148].

5.9 Dynamic range

In some examples, the very first step of the rootfinding algorithm where f and g are replaced by polynomials may cause a numerical problem caused by a large *dynamic range*[12] of f and g [28]. For example, if f and g in (5.1) vary widely in magnitude, then the chebfun2 approximants p and q can have poor relative accuracy (but good absolute accuracy) in regions where $|f| \ll \|f\|_\infty$ and $|g| \ll \|g\|_\infty$. This means that the error bound in (5.5) may be much larger than one would expect by considering the conditioning of a solution. This numerical issue can be easily overcome if the original functions f and g are resampled in the domain subdivision procedure [28].

Since the primary goal of Chebfun2 is to work with approximants rather than function handles and the functionality of Chebfun2 is much broader than just bivariate rootfinding, the implementation of our algorithm in the `roots(f,g)` command does not resample the original functions during subdivision. An implementation of the algorithm described in this chapter that does resample the original functions during subdivision is also publicly available [112].

A typical example for which dynamic range is an issue is when f and g are moderate degree polynomials represented in the monomial basis. This is reflected in the comparisons with the `roots(f,g)` command in [138], and the apparent failure of

[11] The GEP in (5.10) has complex eigenvalues, but we are only interested in the numerically real eigenvalues in $[-1,1]$ as they correspond to solutions to (5.1).

[12] The dynamic range of a complex valued continuous function f is the length of the smallest interval containing the range of $|f|$.

Chebfun2 is due to this dynamic range issue rather than the underlying methodology of the algorithm.

5.10 Numerical examples

In this section we present six examples to illustrate the accuracy and robustness of our algorithm and its ability to solve problems with very high polynomial degree.

In the figures of this section the zero contours of f and g are drawn as blue and red curves, respectively, computed by the command `roots(f)` in Chebfun2 (except for Example 3, which is computed by the `contour` command in MATLAB). The black dots are the solutions computed by the `roots(f,g)` command in Chebfun2 (again, except for Example 3).

5.10.1 Example 1 (Coordinate alignment)

Our first example involves functions f and g approximated by polynomials of degrees $(20, 20)$ and $(24, 30)$, respectively:

$$\begin{pmatrix} T_7(x)T_7(y)\cos(xy) \\ T_{10}(x)T_{10}(y)\cos(x^2y) \end{pmatrix} = 0, \qquad (x, y) \in [-1, 1]^2. \tag{5.11}$$

This problem has 140 solutions. Many of the solutions lie along lines parallel to the coordinate axes, which means that the polynomial $\det(B(y))$ has roots with high multiplicity. This does not cause any numerical difficulties because the roots correspond to semisimple eigenvalues (the algebraic and geometric multiplicities are equal) of the GEP associated with the matrix polynomial $B(y)$, and hence can be calculated to full accuracy [145, 167]. In this case the solutions are computed to an absolute maximum error of 8.88×10^{-16}. Figure 5.5 (left) shows the zero contours and solutions for (5.11).

5.10.2 Example 2 (Face and apple)

In the next example we take functions f and g that are polynomials, i.e., $f = p$ and $g = q$, with zero contours resembling a face and an apple, respectively. These polynomials are from [129] and have degrees $(10, 18)$ and $(8, 8)$. We take the domain to be $[-2, 2] \times [-1.5, 4]$. Figure 5.5 (right) shows the face and apple contours and their intersections.

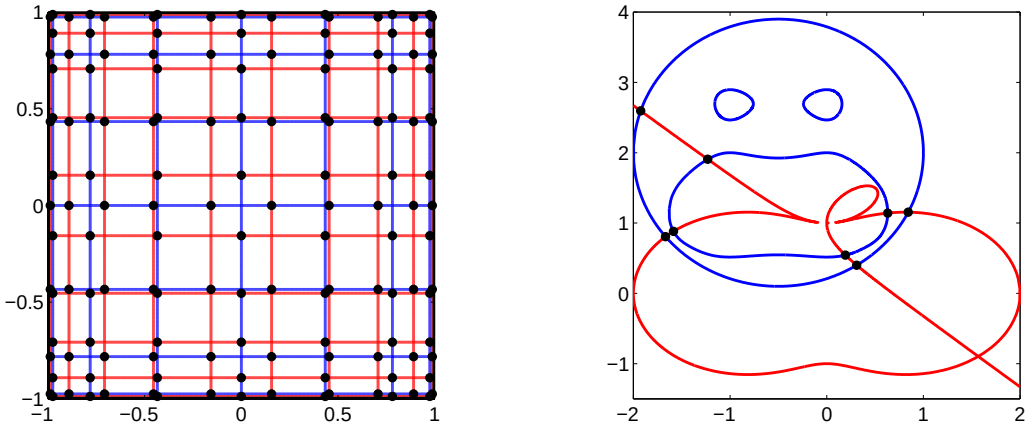


Figure 5.5: The zero contours of f (red) and g (blue), and the common zeros (black dots) computed by the `roots(f,g)` command in Chebfun2. Left: Coordinate alignment (5.11). Simple common zeros are aligned with the coordinate directions, but this causes no numerical difficulties. Right: Face and apple. The polynomials are very small near the origin and local refinement is essential for obtaining accurate solutions.

This example illustrates the importance of local refinement. The polynomials are of low degree, but $|f|$ and $|g|$ are less than 10^{-5} near the origin and initially the solutions are computed inaccurately. However, the ill-conditioned region is detected and the solutions recovered by rerunning the resultant method on a small domain containing the origin (see Section 5.8.2).

5.10.3 Example 3 (Devil's example)

A problem, which due to the dynamic range issue is particularly ill-conditioned, is the following:

$$\begin{pmatrix} \prod_{i=0}^{10}(y^2(4y^2 - i/10) - x^2(4x^2 - 1)) \\ 256(x^2+y^2)^2 + 288(x^2+y^2) - 512(x^3 - 3xy^2) - 27 \end{pmatrix} = 0, \quad (x,y) \in [-1,1]^2. \tag{5.12}$$

We call this the "Devil's example" because of the difficulty it causes bivariate rootfinders when the polynomials are represented in the Chebyshev basis. Here, the functions f and g are polynomials with degrees $(44, 44)$ and $(4, 4)$. It is extremely difficult without resampling the original functions to obtain accurate solutions because the functions vary widely in magnitude. In the `roots(f,g)` command in Chebfun2 we do not resample the original functions f and g when subdividing and for this reason (5.12)

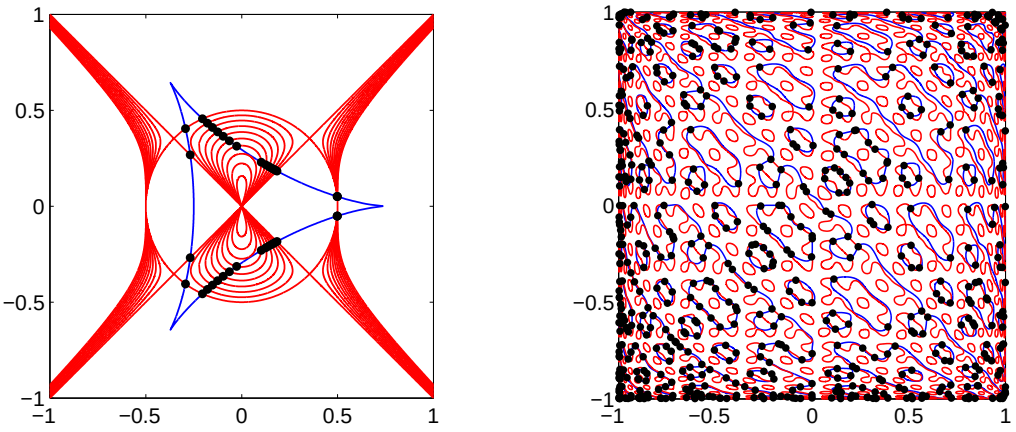


Figure 5.6: Left: Devil's example (5.12). By resampling f and g during the domain subdivision procedure, accurate solutions are obtained despite the dynamic range issue. The function f varies widely in magnitude, and hence Chebfun2 cannot accurately compute the zero level curves of f. Instead, in this figure we use the `contour` command in MATLAB by sampling f on a 2000×2000 equally-spaced tensor product grid. Right: Hadamard example. The polynomials are constructed from interpolation data, and domain subdivision was stopped prematurely as the running estimate for τ was larger than 0.79 (see Section 5.5).

cannot be solved by Chebfun2; however, our implementation that does resample the original functions can compute all the solutions accurately [112].

Figure 5.6 (left) shows the zero contours and solutions.

5.10.4 Example 4 (Hadamard)

Chebfun2 can construct polynomials from interpolation data taken on a Chebyshev tensor product grid. In this example we take the interpolation data to be the Hadamard matrices H_{32} and H_{64} of size 32×32 and 64×64, i.e., we solve $f = g = 0$ on $[-1, 1]^2$, where $f(x_i^{\text{cheb}}, x_j^{\text{cheb}}) = H_{32}(i, j)$, $g(x_i^{\text{cheb}}, x_j^{\text{cheb}}) = H_{64}(i, j)$, and x_i^{cheb} are Chebyshev points on $[-1, 1]$ (see (1.1)). The Hadamard matrices contain ± 1 entries and therefore, f and g (of degrees $(31, 31)$ and $(63, 63)$) have many zero contours. The `roots(f,g)` command in Chebfun2 requires 89 seconds to solve this problem and $|f|$ and $|g|$ are at most 3.98×10^{-13} at the computed solutions. Figure 5.6 (right) shows the zero contours and solutions.

In this example, subdivision was stopped prematurely as it was estimated that $\tau \approx 0.82$ for f and $\tau \approx 0.75$ for g (see Section 5.5). Such high τ are to be expected as f and g are low degree bivariate polynomials with coefficients that do not decay.

5.10.5 Example 5 (Airy and Bessel functions)

As an illustration that the algorithm can solve high degree problems, consider:

$$\begin{pmatrix} \mathrm{Ai}(-13(x^2y+y^2)) \\ J_0(500x)y + xJ_1(500y) \end{pmatrix} = 0, \qquad (x,y) \in [-1,1]^2, \tag{5.13}$$

where Ai is the Airy function and J_0 and J_1 are Bessel functions of the first kind with parameter 0 and 1, respectively. The polynomial degrees required by Chebfun2 are $(171, 120)$ and $(569, 569)$, respectively. The `roots(f,g)` command in Chebfun2 finds all 5,932 solutions in 501 seconds. These solutions were independently verified using a contouring algorithm (see Section 5.4.2). Figure 5.7 (left) shows the computed solutions to (5.13).

Here, $\tau \approx 0.59$ with a corresponding estimated cost of $\mathcal{O}(n^\alpha)$ operations, where $\alpha = -\log 4/\log \tau \approx 2.6$ and n is the maximum polynomial degree in each variable (see Section 5.5).

5.10.6 Example 6 (A SIAM 100-Dollar, 100-Digit Challenge problem)

In 2002, an article in *SIAM News* set a challenge to solve ten problems, each to ten digits (the solution to each problem was a single real number) [155]. The fourth problem was to find the global minimum of the following complicated and highly oscillatory function:

$$\begin{aligned} f(x,y) = &\left(\frac{x^2}{4} + e^{\sin(50x)} + \sin(70\sin(x))\right) + \left(\frac{y^2}{4} + \sin(60e^y) + \sin(\sin(80y))\right) \\ &- \cos(10x)\sin(10y) - \sin(10x)\cos(10y). \end{aligned} \tag{5.14}$$

Since the local extrema of a function, including the global minimum, satisfy $\partial f/\partial x = \partial f/\partial y = 0$, we can solve this problem by computing all the local extrema of f and picking the global minimum from among them. A simple argument shows that the global minimum must occur in $[-1,1]^2$ [25] so we approximate $\partial f/\partial x$ and $\partial f/\partial y$ on $[-1,1]^2$ by polynomials of degrees $(625, 901)$.

The `roots(f,g)` command in Chebfun2 computes all 2,720 local extrema of (5.14) in 257 seconds and obtains the global minimum to an accuracy of 1.12×10^{-15}. The function (5.14) is actually of rank 4 as pointed out in [152] and Chebfun2 correctly calculates the rank, but this structure is not directly exploited in the rootfinding

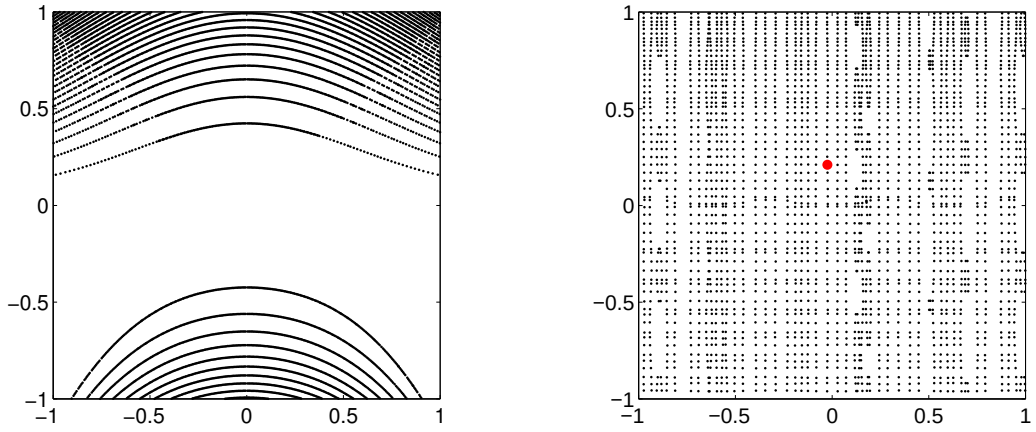


Figure 5.7: Left: Solutions to (5.13). The number of isolated solutions is 5,932. There are so many that they appear as lines rather than dots. Right: The local extrema (black dots) and the global minimum (red dot) of the SIAM 100-Dollar, 100-Digit Challenge problem (5.14). These examples are of very high polynomial degree.

algorithm. Figure 5.7 (right) shows the location of the 2,720 local extrema and the global minimum. In this example $\tau \approx 0.53$ with a corresponding cost of $\mathcal{O}(n^\alpha)$ operations, where $\alpha = -\log 4/\log \tau \approx 2.2$ and n is the maximum polynomial degree in both variables.

Finally, to increase the degrees further, the function in (5.14) was replaced by

$$f(x,y) = \left(\frac{x^2}{4} + e^{\sin(100x)} + \sin(140\sin(x))\right) + \left(\frac{y^2}{4} + \sin(120e^y) + \sin(\sin(160y))\right) \\ - \cos(20x)\sin(20y) - \sin(20x)\cos(20y).$$

Now $\partial f/\partial x$ and $\partial f/\partial y$ are approximated by polynomials of degrees $(1781, 1204)$ on $[-1,1]^2$. Our algorithm computed all 9,318 local extrema in 1,300 seconds with an estimated cost of $\mathcal{O}(n^{2.1})$ operations, where n is the maximum polynomial degree in both variables. These solutions were independently verified by a contouring algorithm (see Section 5.4). The global minimum of $f(x,y)$ on $[-1,1]^2$ computed by Chebfun2 is -3.398166873463248 and we expect the first 14 digits to be correct[13].

[13]We are ready for the next SIAM 100-Dollar, 100-Digit Challenge!

Chapter 6

The automatic solution of linear partial differential equations*

In this chapter we discuss linear partial differential equations (PDEs) defined on rectangular domains with the aim of extending the low rank ideas from functions to partial differential operators. These new ideas lead to an efficient spectral method for solving certain linear PDEs defined on rectangular domains.

Specifically, we consider linear PDEs with variable coefficients defined on $[a, b] \times [c, d]$ of the form:

$$\mathfrak{L}u = f, \qquad \mathfrak{L} = \sum_{i=0}^{N_y} \sum_{j=0}^{N_x} \ell_{ij}(x, y) \frac{\partial^{i+j}}{\partial y^i \partial x^j}, \tag{6.1}$$

where N_x and N_y are the differential orders of $\mathfrak{L}$ in the x- and y-variables, respectively, $f(x, y)$ and $\ell_{ij}(x, y)$ are known functions defined on $[a, b] \times [c, d]$, and $u(x, y)$ is the desired solution. The operator $\mathfrak{L}$ in (6.1) is called a partial differential operator (PDO) [86].

We suppose that the PDE in (6.1) is supplied with $K_x \geq 0$ and $K_y \geq 0$ linear constraints on the solution, i.e.,

$$\mathfrak{B}_x u = \mathbf{g} = \begin{pmatrix} g_1 \\ \vdots \\ g_{K_x} \end{pmatrix}, \qquad \mathfrak{B}_y u = \mathbf{h} = \begin{pmatrix} h_1 \\ \vdots \\ h_{K_y} \end{pmatrix},$$

to ensure that there is a unique solution, where $\mathbf{g}$ and $\mathbf{h}$ are vector valued functions

*This chapter is based on a manuscript in preparation with Sheehan Olver [150]. I proposed the low rank representations of PDEs, showed how to construct such representations, and derived the algorithm for solving PDEs based on these ideas. Olver worked out how to solve the constrained matrix equations and how to deal with general linear constraints.

with components $g_1, \ldots, g_{K_x}$ and $h_1, \ldots, h_{K_y}$, respectively. For example, if $\mathfrak{B}_x$ and $\mathfrak{B}_y$ are Dirichlet boundary conditions, then

$$\mathfrak{B}_x u = \begin{pmatrix} u(a\,,\,\cdot) \\ u(b\,,\,\cdot) \end{pmatrix}, \qquad \mathfrak{B}_y u = \begin{pmatrix} u(\cdot\,,\,c) \\ u(\cdot\,,\,d) \end{pmatrix}.$$

Here, $K_x = K_y = 2$ and the vector valued functions $\mathbf{g}$ and $\mathbf{h}$ represent the Dirichlet data on the boundary of $[a, b] \times [c, d]$. Other examples of linear constraints are Robin conditions, constraints involving high-order derivatives, and integral conditions. Without loss of generality we assume that the constraints are linearly independent in the sense that their number cannot be reduced without relaxing a constraint on the solution.

Throughout this chapter the variable coefficients in (6.1), the right-hand side f, and the solution u, are assumed to be smooth functions, and we restrict our attention to linear PDEs defined on $[-1, 1]^2$, unless stated otherwise. The ideas and implementation of the resulting PDE solver permit general rectangular domains.

6.1 Low rank representations of partial differential operators

A low rank representation of a 2D object is a sum of "outer products" of 1D objects. For linear PDOs those 1D objects are linear *ordinary differential operators* (ODOs). A linear ODO acts on univariate functions and can be written as a finite linear combination of differential and multiplication operators.

A linear PDO is of rank 1 if it can be written as an outer product of two ODOs, i.e., $\mathfrak{L}^y \otimes \mathfrak{L}^x$. For example, $\nabla^2 = \mathcal{D}^2 \otimes \mathcal{I} + \mathcal{I} \otimes \mathcal{D}^2$, where $\mathcal{D}$ is the first-order differential operator and $\mathcal{I}$ is the identity operator. In each case, the operator $\mathfrak{L}^y \otimes \mathfrak{L}^x$ acts on bivariate functions by

$$(\mathfrak{L}^y \otimes \mathfrak{L}^x)\, u = \sum_{j=1}^{\infty} (\mathfrak{L}^y c_j)\, (\mathfrak{L}^x r_j)\,, \qquad u(x, y) = \sum_{j=1}^{\infty} c_j(y) r_j(x).$$

In Chapter 2 a rank 1 PDO would have been referred to as a tensor product operator (see Definition 2.1). A linear PDO $\mathfrak{L}$ of rank k can be written as a sum of k rank 1 PDOs, i.e.,

$$\mathfrak{L} = \sum_{j=1}^{k} \left(\mathfrak{L}_j^y \otimes \mathfrak{L}_j^x\right),$$

and the *rank* of a PDO is the minimum number of terms in such a representation.

Definition 6.1. *Let $\mathfrak{L}$ be a linear PDO in the form* (6.1)*. The rank of $\mathfrak{L}$ is the smallest integer k for which there exist ODOs $\mathfrak{L}_1^y, \ldots, \mathfrak{L}_k^y$ (acting on functions in y) and $\mathfrak{L}_1^x, \ldots, \mathfrak{L}_k^x$ (acting on functions in x) that satisfy*

$$\mathfrak{L} = \sum_{j=1}^{k} \left(\mathfrak{L}_j^y \otimes \mathfrak{L}_j^x \right). \tag{6.2}$$

A linear PDO of finite differential order with variable coefficients of finite rank is itself of finite rank and is usually of infinite rank if one of the variable coefficients is of infinite rank.

6.2 Determining the rank of a partial differential operator

One way to determine the rank of a PDO is directly from Definition 6.1. For example, the rank of the Helmholtz operator $\partial^2/\partial x^2 + \partial^2/\partial y^2 + K^2$ is 2 since

$$\partial^2/\partial x^2 + \partial^2/\partial y^2 + K^2 = \left(\mathfrak{I} \otimes \mathfrak{D}^2 \right) + \left((\mathfrak{D}^2 + K^2 \mathfrak{I}) \otimes \mathfrak{I} \right),$$

where $\mathfrak{I}$ is the identity operator and $\mathfrak{D}$ is the first order differential operator. Furthermore, it can be shown that the rank of $\partial^2/\partial x^2 + \partial^2/\partial x \partial y + \partial^2/\partial y^2$ is 3 and the rank of $(2 + \sin(x+y))\partial^2/\partial x^2 + e^{-(x^2+y^2)}\partial^2/\partial y^2$ is 4.

Another way to calculate the rank of a PDO is given by the following lemma.

Lemma 6.1. *Let $\mathfrak{L}$ be a linear PDO in the form* (6.1) *with variable coefficients of finite rank. Then, the rank of $\mathfrak{L}$ is equal to the minimum number of terms k required in an expression of the form*

$$\sum_{i=0}^{N_y} \sum_{j=0}^{N_x} \ell_{ij}(s,t) y^i x^j = \sum_{j=1}^{k} c_j(t,y) r_j(s,x), \tag{6.3}$$

where c_j and r_j are bivariate functions.

Proof. Let $\mathcal{T}$ be the linear operator[1] defined by

$$\mathcal{T}\left[\ell(s,t)y^i x^j\right] = \ell(x,y)\frac{\partial^{i+j}}{\partial y^i \partial x^j}, \qquad i,j \geq 0,$$

which replaces x and y by s and t and powers of x and y by partial derivatives. Now suppose that $\mathfrak{L}$ is a linear PDO of rank r and k is the minimum number of terms required in (6.3). We show that $r = k$.

Note that the linear operator $\mathcal{T}$ can be used to give the following relation:

$$\mathfrak{L} = \sum_{i=0}^{N_y}\sum_{j=0}^{N_x} \ell_{ij}(x,y)\frac{\partial^{i+j}}{\partial y^i \partial x^j} = \mathcal{T}\left[\sum_{i=0}^{N_y}\sum_{j=0}^{N_x} \ell_{ij}(s,t)y^i x^j\right] = \mathcal{T}\left[H(s,x,t,y)\right].$$

where $H(s,x,t,y) = \sum_{i=0}^{N_y}\sum_{j=0}^{N_x} \ell_{ij}(s,t)y^i x^j$. If the function $H(s,x,t,y)$ can be written as $\sum_{j=1}^{k} c_j(t,y)r_j(s,x)$, then we have

$$\mathfrak{L} = \mathcal{T}\left[\sum_{j=1}^{k} c_j(t,y)r_j(s,x)\right] = \sum_{j=1}^{k}\mathcal{T}\left[c_j(t,y)r_j(s,x)\right] = \sum_{j=1}^{k}\mathcal{T}\left[c_j(t,y)\right]\otimes\mathcal{T}\left[r_j(s,x)\right],$$

where $\mathcal{T}[c_j(t,y)]$ and $\mathcal{T}[r_j(s,x)]$ are ODOs with variable coefficients in y and x, respectively, and hence $r \leq k$. Conversely, a rank r expression for $\mathfrak{L}$ can be converted (using $\mathcal{T}$) to a rank r expression for H, and hence $k \leq r$. We conclude that $r = k$ and the rank of $\mathfrak{L}$ equals the minimum number of terms required in (6.3). □

A special case of Lemma 6.1 gives a connection between constant coefficient PDOs and bivariate polynomials. This has been previously used to investigate polynomial systems of equations [144, Chap. 10]. Thus, if $\mathfrak{L}$ has constant coefficients then a low rank representation can be constructed via the SVD of a bivariate polynomial (see Section 4.8).

More generally, Lemma 6.1 allows us to calculate a low rank representation for a linear PDO via a low rank representation of an associated function. For PDOs with variable coefficients of finite rank we do not make an attempt to find the exact rank,

[1]The definition of this operator is motivated by Blissard's symbolic method (also known as umbral calculus) [17].

PDO	Operator
Laplace	$u_{xx} + u_{yy}$
Helmholtz	$u_{xx} + u_{yy} + K^2 u$
Heat	$u_t - \alpha^2 u_{xx}$
Transport	$u_t - b u_x$
Wave	$u_{tt} - c^2 u_{xx}$
Euler–Tricomi	$u_{xx} - x u_{yy}$
Schrödinger	$\mathrm{i}\epsilon u_t + \frac{1}{2}\epsilon^2 u_{xx} - V(x)u$
Black–Scholes	$u_t + \frac{1}{2}\sigma^2 x^2 u_{xx} + rx u_x - ru$

Table 6.1: Selection of rank 2 PDOs (see Definition 6.1). Not all constant coefficient PDOs are of rank 2. For instance, the biharmonic operator is of rank 3.

but instead we write each variable coefficient in (6.1) in low rank form,

$$\ell_{ij}(x,y) = \sum_{s=1}^{k_{ij}} c_s^{ij}(y) r_s^{ij}(x), \qquad 0 \le i \le N_y, \quad 0 \le j \le N_x,$$

and then use the following low rank representation for $\mathfrak{L}$:

$$\mathfrak{L} = \sum_{i=0}^{N_y} \sum_{j=0}^{N_x} \sum_{s=1}^{k_{ij}} \left(\mathfrak{M}[c_s^{ij}]\mathfrak{D}^i\right) \otimes \left(\mathfrak{M}[r_s^{ij}]\mathfrak{D}^j\right), \tag{6.4}$$

where $\mathfrak{M}[a]$ is the multiplication operator for $a(x)$, i.e., $\mathfrak{M}[a]u = a(x)u(x)$. Fortunately, in practice, it turns out that when a PDO is of rank greater than 2 there is little to be gained by finding a low rank representation of minimal rank (see Section 6.5.1). If a variable coefficient is of infinite rank, then we can replace it by a chebfun2 approximant and proceed to form (6.4).

Surprisingly, several standard linear PDOs are of rank 2 and Table 6.1 presents a selection. Most linear PDOs with variable coefficients are of rank greater than 2, and ODOs are rank 1 PDOs.

6.3 The ultraspherical spectral method for ordinary differential equations*

Once we have a low rank expression for a PDO we proceed to represent the rank 1 terms by a spectral method. Possible choices are the Chebyshev collocation method [51, 154], Petrov–Galerkin methods [133], and the Tau-method [116]. We have chosen the ultraspherical spectral method because it leads to a spectrally accurate discretization and almost banded[2] well-conditioned matrices. This section reviews the ultraspherical spectral method (for further details see [115]).

First, we consider a linear ODE with constant coefficients defined on $[-1, 1]$ with the following form:

$$a_N \frac{d^N u}{dx^N} + \cdots + a_1 \frac{du}{dx} + a_0 u = f, \qquad N \geq 1, \tag{6.5}$$

where $a_0, \ldots, a_N$ are complex numbers, f is a known univariate function, and u is an unknown solution. Furthermore, we assume that the ODE is supplied with K linear constraints, i.e., $\mathfrak{B}u = \mathbf{c}$ where $\mathfrak{B}$ is a linear operator and $\mathbf{c} \in \mathbb{C}^K$, so that the solution to (6.5) is unique. The ultraspherical spectral method aims to find the solution of (6.5) represented in the Chebyshev basis and computes a vector of Chebyshev expansion coefficients of the solution. That is, the spectral method seeks to find an infinite vector $\mathbf{u} = (u_0, u_1, \ldots)^T$ such that

$$u(x) = \sum_{j=0}^{\infty} u_j T_j(x),$$

where T_j is the jth Chebyshev polynomial (see Section 1.1).

Classically, spectral methods represent differential operators by dense matrices [26,

[2] A matrix is *almost banded* if it is banded except for a small number of columns or rows.

*This section is based on a paper with Sheehan Olver [115]. I devised the scheme to construct ultraspherical multiplication operators and the algorithm for solving linear systems involving almost banded matrices. Olver constructed the differentiation and conversion operators and noted that the Chebyshev multiplication operator has a Toeplitz-plus-Hankel plus rank 1 form. The paper contains far more than this short summary. The majority of Section 6.3.1 and the whole of Section 6.3.2 do not appear in the paper.

51, 154], but the ultraspherical spectral method employs a "sparse" recurrence relation

$$\frac{d^k T_n}{dx^k} = \begin{cases} 2^{k-1} n (k-1)!\, C_{n-k}^{(k)}, & n \geq k, \\ 0, & 0 \leq n \leq k-1, \end{cases}$$

where $C_j^{(k)}$ is the ultraspherical polynomial with parameter $k > 0$ of degree j (see Section 1.5). This results in a sparse representation for first and higher-order differential operators. In particular, the differentiation matrix for the kth derivative is given by

$$\mathcal{D}_k = 2^{k-1}(k-1)! \begin{pmatrix} \overbrace{0 \quad \cdots \quad 0}^{k \text{ times}} & k & & & \\ & & k+1 & & \\ & & & k+2 & \\ & & & & \ddots \end{pmatrix}, \qquad k \geq 1.$$

For $k \geq 1$, $\mathcal{D}_k$ maps a vector of Chebyshev expansion coefficients to a vector of $C^{(k)}$ expansion coefficients of the kth derivative. The operator $\mathcal{D}_0$ is the infinite identity matrix.

Since $\mathcal{D}_k$ for $k \geq 1$ returns a vector of ultraspherical expansion coefficients, the ultraspherical spectral method also requires *conversion matrices* denoted by $\mathcal{S}_k$ for $k \geq 0$. The matrix $\mathcal{S}_0$ converts a vector of Chebyshev coefficients to a vector of $C^{(1)}$ coefficients and, more generally, the operator $\mathcal{S}_k$ for $k \geq 1$ converts a vector of $C^{(k)}$ coefficients to a vector of $C^{(k+1)}$ coefficients. Using the relations in [114, (18.9.7) and (18.9.9)] it can be shown that (see [115] for a derivation)

$$\mathcal{S}_0 = \begin{pmatrix} 1 & 0 & -\frac{1}{2} & & \\ & \frac{1}{2} & 0 & -\frac{1}{2} & \\ & & \frac{1}{2} & 0 & \ddots \\ & & & \frac{1}{2} & \ddots \\ & & & & \ddots \end{pmatrix}, \qquad \mathcal{S}_k = \begin{pmatrix} 1 & 0 & -\frac{k}{k+2} & & \\ & \frac{k}{k+1} & 0 & -\frac{k}{k+3} & \\ & & \frac{k}{k+2} & 0 & \ddots \\ & & & \frac{k}{k+3} & \ddots \\ & & & & \ddots \end{pmatrix}, \quad k \geq 1.$$

Note that for $k \geq 1$, the matrix $\mathcal{S}_0^{-1} \cdots \mathcal{S}_{k-1}^{-1} \mathcal{D}_k$ is dense and upper-triangular. This is the matrix that represents kth order differentiation in the Chebyshev basis without converting to ultraspherical bases [116].

We can combine these conversion and differentiation matrices to represent the

ODE in (6.5) as follows:

$$(a_N\mathcal{D}_N + a_{N-1}\mathcal{S}_{N-1}\mathcal{D}_{N-1} + \cdots + a_0\mathcal{S}_{N-1}\cdots\mathcal{S}_0\mathcal{D}_0)\,\mathbf{u} = \mathcal{S}_{N-1}\cdots\mathcal{S}_0\mathbf{f}, \tag{6.6}$$

where $\mathbf{u}$ and $\mathbf{f}$ are vectors of Chebyshev expansion coefficients of u and f, respectively. The conversion matrices are used in (6.6) to ensure that the resulting linear combination maps Chebyshev coefficients to $C^{(N)}$ coefficients, and the right-hand side f is represented by a vector of $C^{(N)}$ expansion coefficients.

To make the solution to (6.6) unique we must impose the K prescribed linear constraints in $\mathcal{B}$ on $\mathbf{u}$. That is, we must represent the action of the linear constraints on a vector of Chebyshev coefficients. For example, Dirichlet boundary conditions take the form

$$\mathcal{B} = \begin{pmatrix} T_0(-1) & T_1(-1) & T_2(-1) & T_3(-1) & \cdots \\ T_0(1) & T_1(1) & T_2(1) & T_3(1) & \cdots \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 & -1 & \cdots \\ 1 & 1 & 1 & 1 & \cdots \end{pmatrix},$$

since then $\mathcal{B}\mathbf{u} = (u(-1), u(1))^T$, and Neumann conditions at $x = \pm 1$ take the form

$$\mathcal{B} = \begin{pmatrix} T_0'(-1) & T_1'(-1) & T_2'(-1) & T_3'(-1) & \cdots \\ T_0'(1) & T_1'(1) & T_2'(1) & T_3'(1) & \cdots \end{pmatrix} = \begin{pmatrix} 0 & -1 & 4 & -9 & \cdots \\ 0 & 1 & 4 & 9 & \cdots \end{pmatrix},$$

since then $\mathcal{B}\mathbf{u} = (u'(-1), u'(1))^T$. In general, any linear constraint can be represented by its action on a vector of Chebyshev coefficients.

Finally, to construct a linear system that can be solved for the first n Chebyshev coefficients of u we take the $n \times n$ finite section of various matrices. Let $\mathcal{P}_n$ be the *truncation matrix* that maps $\mathbb{C}^\infty$ to $\mathbb{C}^n$ such that $\mathcal{P}_n\mathbf{u} = (u_0, \ldots, u_{n-1})^T$. We take the first n columns of $\mathcal{B}$, $B = \mathcal{B}\mathcal{P}_n^T$, the $(n-K) \times n$ principal submatrix of $\mathcal{L}$, $L = \mathcal{P}_{n-K}\mathcal{L}\mathcal{P}_n^T$, and form the linear system:

$$\begin{pmatrix} B \\ L \end{pmatrix} \mathcal{P}_n\mathbf{u} = \begin{pmatrix} \mathbf{c} \\ \mathcal{P}_{n-K}\mathcal{S}_{N-1}\cdots\mathcal{S}_0\mathbf{f} \end{pmatrix}. \tag{6.7}$$

Since the matrices $\mathcal{D}_k$ and $\mathcal{S}_k$ are banded, the matrix L is banded, and the resulting linear system is almost banded, i.e., banded except for K rows imposing the linear constraints on $\mathbf{u}$. The K rows in (6.7) that impose the linear constraints could also be placed below L, but we have decided to make the linear system have a structure that is as close as possible to upper-triangular.

6.3.1 Multiplication matrices

For ODEs with variable coefficients we need to be able to represent the multiplication operation $\mathcal{M}[a]u = a(x)u(x)$, and since the ultraspherical spectral method converts between different ultraspherical bases, we need to construct multiplication matrices for each ultraspherical basis.

Suppose we wish to represent $\mathcal{M}[a]u$ where $a(x)$ and $u(x)$ have Chebyshev expansions

$$a(x) = \sum_{j=0}^{\infty} a_j T_j(x), \qquad u(x) = \sum_{j=0}^{\infty} u_j T_j(x),$$

and we desire the Chebyshev expansion coefficients of $a(x)u(x)$. Define $\mathcal{M}_0[a]$ to be the matrix that takes a vector of Chebyshev expansion coefficients of $u(x)$ and returns the vector of Chebyshev expansion coefficients of $a(x)u(x)$. It is shown in [115] that $\mathcal{M}_0[a]$ can be written as the following Toeplitz-plus-Hankel plus rank 1 matrix:

$$\mathcal{M}_0[a] = \frac{1}{2}\left[\begin{pmatrix} 2a_0 & a_1 & a_2 & a_3 & \dots \\ a_1 & 2a_0 & a_1 & a_2 & \ddots \\ a_2 & a_1 & 2a_0 & a_1 & \ddots \\ a_3 & a_2 & a_1 & 2a_0 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & \dots \\ a_1 & a_2 & a_3 & a_4 & \iddots \\ a_2 & a_3 & a_4 & a_5 & \iddots \\ a_3 & a_4 & a_5 & a_6 & \iddots \\ \vdots & \iddots & \iddots & \iddots & \iddots \end{pmatrix}\right].$$

This multiplication operator looks dense. However, if $a(x)$ is approximated by a polynomial of degree m then $\mathcal{M}_0[a]$ is banded with bandwidth of m. Moreover, the matrix-vector product $\mathcal{M}_0[a]\mathbf{u}$ returns the Chebyshev expansion coefficients of $a(x)u(x)$.

We also require multiplication operations $\mathcal{M}_k[a]$ representing multiplication of two $C^{(k)}$ series. That is, if $\mathbf{u}$ is a vector of Chebyshev expansion coefficients of u, then the sequence of matrices $\mathcal{M}_k[a]\mathcal{S}_{k-1}\cdots\mathcal{S}_0\mathbf{u}$ returns the $C^{(k)}$ expansion coefficients of $a(x)u(x)$. In [115] an explicit formula for the entries of $\mathcal{M}_k[a]$ for $k \geq 1$ is given. Here, we show that the matrices $\mathcal{M}_k[a]$ can also be constructed from a 3-term recurrence relation.

Suppose we wish to construct $\mathcal{M}_k[a]$, where $a(x)$ has a uniformly and absolute convergent $C^{(k)}$ expansion given by

$$a(x) = \sum_{j=0}^{\infty} a_j C_j^{(k)}(x).$$

Since $\mathcal{M}_k[a]$ is a linear operator we know that

$$\mathcal{M}_k[a] = \sum_{j=0}^{\infty} a_j \mathcal{M}_k[C_j^{(k)}], \tag{6.8}$$

and furthermore, since the ultraspherical polynomials satisfy a 3-term recurrence relation (see (1.7)) and multiplication is associative we have

$$\mathcal{M}_k[C_{j+1}^{(k)}] = \frac{2(j+k)}{j+1}\mathcal{M}_k[x]\mathcal{M}_k[C_j^{(k)}] - \frac{j+2k-1}{j+1}\mathcal{M}_k[C_{j-1}^{(k)}], \qquad j \geq 1. \tag{6.9}$$

Therefore, the terms in the series (6.8) can be constructed recursively from $\mathcal{M}_k[C_1^{(k)}]$ and $\mathcal{M}_k[C_0^{(k)}]$. The matrix $\mathcal{M}_k[C_0^{(k)}]$ is the infinite identity operator because $C_0^{(k)} = 1$ and the construction of $\mathcal{M}_k[C_1^{(k)}]$ is all that remains. Since $C_1^{(k)} = 2kx$ we have $\mathcal{M}_k[C_1^{(k)}] = 2k\mathcal{M}_k[x]$. To derive $\mathcal{M}_k[x]$ we rewrite the 3-term recurrence relation satisfied by ultraspherical polynomials as

$$xC_j^{(k)}(x) = \begin{cases} \frac{j+1}{2(j+k)}C_{j+1}^{(k)}(x) + \frac{j+2k-1}{2(j+k)}C_{j-1}^{(k)}(x), & j \geq 1, \\ \frac{1}{2k}C_1^{(k)}(x), & j = 0, \end{cases}$$

and then use it to derive a $C^{(k)}$ expansion for $xu(x)$, where $u(x) = \sum_{j=0}^{\infty} u_j C_j^{(k)}(x)$. We have

$$\begin{aligned} xu(x) &= \sum_{j=0}^{\infty} u_j x C_j^{(k)}(x) = \sum_{j=1}^{\infty} u_j x C_j^{(k)}(x) + u_0 x C_0^{(k)}(x) \\ &= \sum_{j=1}^{\infty} u_j \frac{j+1}{2(j+k)} C_{j+1}^{(k)}(x) + \sum_{j=1}^{\infty} u_j \frac{j+2k-1}{2(j+k)} C_{j-1}^{(k)}(x) + u_0 \frac{1}{2k} C_1^{(k)}(x) \\ &= \sum_{j=1}^{\infty} \left(u_{j-1} \frac{j}{2(j+k-1)} + u_{j+1} \frac{j+2k}{2(j+k+1)} \right) C_j^{(k)}(x) + u_1 \frac{2k}{2(k+1)} C_0^{(k)}(x), \end{aligned}$$

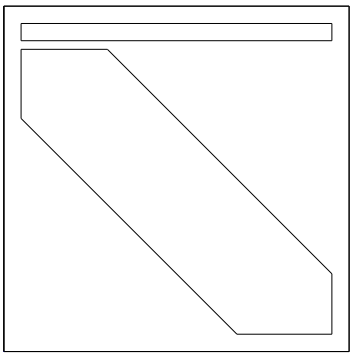


Figure 6.1: Typical structure of the matrices constructed by the ultraspherical spectral method in [115].

which calculates the columns of $\mathcal{M}_k[x]$, i.e.,

$$\mathcal{M}_k[x] = \begin{pmatrix} 0 & \frac{2k}{2(k+1)} & & & \\ \frac{1}{2k} & 0 & \frac{2k+1}{2(k+2)} & & \\ & \frac{2}{2(k+1)} & 0 & \frac{2k+2}{2(k+3)} & \\ & & \frac{2}{2(k+2)} & 0 & \ddots \\ & & & \ddots & \ddots \end{pmatrix}.$$

Thus, $\mathcal{M}_k[a]$ in (6.8) can be constructed from the 3-term recurrence relation in (6.9).

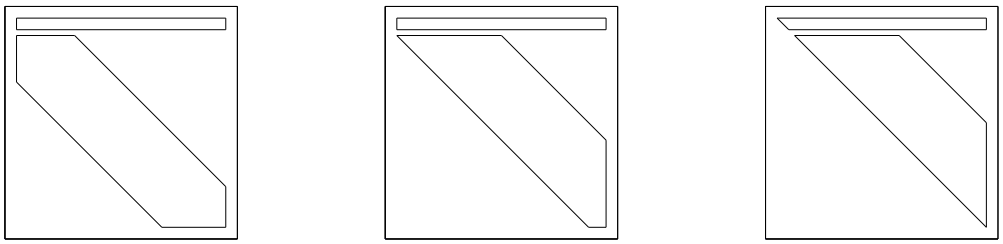
These multiplication operators can be used to represent ODEs with variable coefficients. For example, if

$$a_N \frac{d^N u}{dx^N} + \cdots + a_1 \frac{du}{dx} + a_0 u = f, \qquad N \geq 1,$$

where $a_0, \ldots, a_N$ are univariate functions, then the ODE can be represented as

$$\left(\mathcal{M}_N[a_N]\mathcal{D}_N + \cdots + \mathcal{S}_{N-1}\cdots\mathcal{S}_1\mathcal{M}_1[a_1]\mathcal{D}_1 + \mathcal{S}_{N-1}\cdots\mathcal{S}_0\mathcal{M}_0[a_0]\mathcal{D}_0\right)\mathbf{u} = \mathcal{S}_{N-1}\cdots\mathcal{S}_0\mathbf{f},$$

where the conversion matrices are used to ensure that the range is represented in a $C^{(N)}$ basis. The K linear constraints can be incorporated into the linear system in the same way as before (see (6.7)). Figure 6.1 shows the typical structure of the nonzero entries in a linear system constructed by the ultraspherical spectral method.



(a) Original matrix (b) After left Givens rotations (c) After right Givens rotations

Figure 6.2: Typical structure during the computation of the QRP* factorization of an almost banded matrix.

6.3.2 Fast linear algebra for almost banded matrices

The ultraspherical spectral method requires the solution of a linear system $Ax = b$ where $A \in \mathbb{C}^{n\times n}$ is a banded matrix with $m_{\mathrm{R}} = \mathcal{O}(m)$ nonzero superdiagonals and $m_{\mathrm{L}} = \mathcal{O}(m)$ nonzero subdiagonals (so that $m = m_{\mathrm{L}} + m_{\mathrm{R}} + 1$), except for the first K dense boundary rows. The typical structure of A is shown in Figure 6.2a. Here, we describe a stable algorithm[3] to solve $Ax = b$ in $\mathcal{O}(m^2 n)$ operations and a storage complexity of $\mathcal{O}(mn)$.

Since A is nearly upper-triangular, except for m_1 subdiagonals, a first attempt to solve $Ax = b$ is to compute the QR factorization by applying Givens rotations on the left. Unfortunately, whether we apply the rotations on the left or the right, the resulting upper-triangular part will become dense because of the boundary rows. Instead, by applying a partial factorization on the left followed by another partial factorization on the right we can obtain a factorization $A = QRP^*$, where P and Q are orthogonal and R is upper-triangular with no more nonzero entries than A.

We first apply Givens rotations on the left of A to introduce zero entries in the m_1th subdiagonal. These rotations introduce zeros starting in the $(m_1, 1)$ entry and successively eliminate down the subdiagonal to the $(n, n - m_1 + 1)$ entry. Each rotation applied on the left of A results in a linear combination of two neighboring rows. One of the rows contains the nonzero entry to be eliminated and the other is the row immediately above this. After the first $n - m_1$ rotations, the resulting matrix has a zero m_1th subdiagonal and, typically, nonzero entries along the $(m_2 + 1)$th superdiagonal have been introduced.

[3]Similar techniques have been employed by Chandrasekaran and Gu for solving linear systems involving banded plus semiseparable linear systems [33]. An alternative algorithm for solving almost banded linear systems is the QR factorization applied to a "filled-in" representation [115].

In the same way, we then use Givens rotations to eliminate, in turn, the (m_1-1)th, (m_1-2)th, ..., $(K+1)$th subdiagonals. The typical structure of the resulting matrix is shown in Figure 6.2b and has lower bandwidth of K and an upper bandwidth of m_2+m_1-K, except for the K boundary rows.

We now apply Givens rotations on the right. The first sequence of rotations eliminate the Kth subdiagonal by starting in the last row and successively eliminate to the Kth row. Each rotation applied on the right performs a linear combination of two neighboring columns. One of the columns contains the nonzero entry to be eliminated and the other is the column immediately to the right. After the first $n-K$ rotations the resulting matrix is zero along the Kth subdiagonal and, typically, nonzero entries along the (m_2+m_1-K+1)th superdiagonal have been introduced. Finally, we use Givens rotations to eliminate, in turn, the $(K-1)$th, $(K-2)$th,..., 1st subdiagonals. The resulting matrix is upper-triangular, with upper bandwidth of m_1+m_2 (except for the first K rows) as shown in Figure 6.2c.

In summary, each Givens rotation, whether applied on the left or right, eliminates one zero in a subdiagonal and introduces at most one nonzero in a superdiagonal. Thus, the final upper-triangular matrix usually has no more nonzeros than the original matrix A.

As mentioned above, this factorization can be written as $A = QRP^*$, where $Q, P \in \mathbb{C}^{n\times n}$ are orthogonal and $R \in \mathbb{C}^{n\times n}$ is upper-triangular. To reduce storage requirements we overwrite A with Q, R, and P as we perform the factorization using a technique described by Demmel [43, p. 123]. Note that, if we were solving $Ax = b$ just once, we would not store any information about Q, because we could apply the left Givens rotations directly to the vector b. However, in practice our method progressively increases n, until the solution of the differential equation is well-resolved, and in this case we can reapply the left Givens rotation used on the smaller system when computing the factorization for the new larger system.

In summary the system $Ax = b$ is solved with the following four steps:

1. Factorize $A = QRP^*$ by applying left and right Givens rotations,
2. Compute Q^*b,
3. Solve $Ry = Q^*b$ for y by back substitution,
4. Compute $x = Py$.

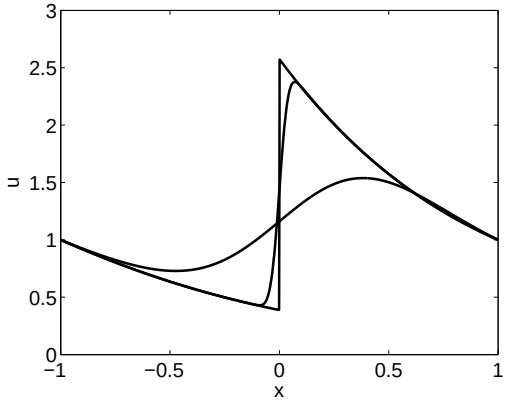


Figure 6.3: The solution of $eu''(x) + xu'(x) + \sin(x)u(x) = 0$, $u(\pm 1) = 1$, for $e = 10^{-1}, 10^{-3}, 10^{-7}$. For $e = 10^{-7}$ the solution requires a polynomial of degree 22,950 to be globally resolved to machine precision. Typically, the ultraspherical spectral method constructs well-conditioned matrices and hence, can resolve solutions that require large linear systems.

The factorization requires $\mathcal{O}(mn)$ Givens rotations with each one involving a linear combination of $\mathcal{O}(m)$ nonzero entries. Hence, this factorization can be computed in $\mathcal{O}(m^2 n)$ operations. Back substitution takes $\mathcal{O}(m^2 n)$ operations, since R is banded except for the first $K = \mathcal{O}(1)$ rows. Moreover, $Q^* b$ and Py are computed by applying the $\mathcal{O}(mn)$ rotations to vectors and hence can be computed in $\mathcal{O}(mn)$ operations. In total, an almost banded linear system constructed by the ultraspherical spectral method can be solved in $\mathcal{O}(m^2 n)$ operations.

The storage requirement is asymptotically minimal since A has $\mathcal{O}(mn)$ nonzero entries, and R also has $\mathcal{O}(mn)$ nonzero entries.

Typically, the linear systems constructed by the ultraspherical spectral method are well-conditioned [115]. Therefore, the accuracy of the computed solution is usually close to machine precision. Figure 6.3 shows the solution to a boundary layer problem $eu''(x)+xu'(x)+\sin(x)u(x) = 0$, $u(\pm 1) = 1$, for $e = 10^{-1}, 10^{-3}, 10^{-7}$. For $e = 10^{-7}$ the solution requires a Chebyshev expansion of degree 22,950 to be resolved to machine precision.

6.4 Discretization of partial differential operators in low rank form

Once discretized a low rank representation for a PDO (see (6.2)) becomes a generalized Sylvester matrix equation, i.e., $A_1 X B_1^T + \cdots + A_k X B_k^T$, where the matrices

$A_1, \ldots, A_k$ and $B_1, \ldots, B_k$ are ultraspherical spectral discretizations of ODOs and X is a matrix containing the bivariate Chebyshev coefficients for the solution.

Specifically, suppose we seek to compute an $n_y \times n_x$ matrix X of bivariate Chebyshev expansion coefficients of the solution $u(x, y)$ to (6.1) satisfying

$$\left| u(x,y) \approx \sum_{i=0}^{n_y} \sum_{j=0}^{n_x} X_{ij} T_i(y) T_j(x) \right| \leq \mathcal{O}(\epsilon \|u\|_\infty), \qquad (x,y) \in [-1,1]^2,$$

where ϵ is machine precision. The ultraspherical spectral method can be used to represent the ODOs $\mathfrak{L}_1^y, \ldots, \mathfrak{L}_k^y$ and $\mathfrak{L}_1^x, \ldots, \mathfrak{L}_k^x$ in (6.2) as matrices $\mathcal{L}_1^y, \ldots, \mathcal{L}_k^y$ and $\mathcal{L}_1^x, \ldots, \mathcal{L}_k^x$. These matrices can be truncated to form a generalized Sylvester matrix equation

$$A_1 X B_1^T + \cdots + A_k X B_k^T = F, \tag{6.10}$$

where $A_j = \mathcal{P}_{n_y} \mathcal{L}_j^y \mathcal{P}_{n_y}^T$ and $B_j = \mathcal{P}_{n_x} \mathcal{L}_j^x \mathcal{P}_{n_x}^T$ for $1 \leq j \leq k$, and F is the $n_y \times n_x$ matrix of bivariate Chebyshev expansion coefficients of the right-hand side f in (6.1).

Typically, the matrix equation (6.10) does not have a unique solution as the prescribed linear constraints $\mathfrak{B}_x$ and $\mathfrak{B}_y$ must also be incorporated. By investigating the action of $\mathfrak{B}_x$ on the basis $\{T_0(x), \ldots, T_{n_x-1}(x)\}$, we can discretize any linear constraint as

$$X B_x^T = G^T,$$

where B_x is an $K_x \times n_x$ matrix and G is an $K_x \times n_y$ matrix containing the first n_y Chebyshev coefficients of each component of $\mathbf{g}$. Similarly, by investigating the action of $\mathcal{B}_y$ on the basis $\{T_0(y), \ldots, T_{n_y-1}(y)\}$ we can discretize $\mathcal{B}_y u = \mathbf{h}$ as

$$B_y X = H,$$

where H is an $K_y \times n_x$ matrix containing the first n_x Chebyshev coefficients of each component of $\mathbf{h}$.

For the constraints to be consistent the matrices B_x and B_y must satisfy the following *compatibility condition*:

$$H B_x^T = (B_y X) B_x^T = B_y (X B_x^T) = B_y G^T. \tag{6.11}$$

For example, in order that Dirichlet conditions satisfy the compatibility conditions

the boundary data must agree at the four corners of $[-1,1]^2$.

In practice, the solver determines the parameters n_x and n_y by progressively discretizing the PDE on finer grids until the solution is resolved. First, we discretize the PDE with $n_x = n_y = 9$ and solve the resulting matrix equation (6.10) under linear constraints (see the next section for details). Then, we check if the Chebyshev coefficients in X decay to below machine precision. Roughly speaking, if the last few columns of X are above machine precision, then the solution has not been resolved in the x-variable and n_x is increased to $17, 33, 65$, and so on, and if the last few rows in X are above machine precision, then n_y is increased to $17, 33, 65$, and so on. The exact resolution tests we employ are the same as those employed by Chebfun2, which are heuristic in nature, but based on a significant amount of practical experience. The discretization parameters n_x and n_y are independently increased and the resolution test is performed in both directions after each solve.

6.5 Solving matrix equations with linear constraints

In this section we describe how to solve the following matrix equation with linear constraints:

$$\sum_{j=1}^{k} A_j X B_j^T = F, \quad X \in \mathbb{C}^{n_y \times n_x}, \qquad B_y X = H, \qquad X B_x^T = G^T, \tag{6.12}$$

where $A_j \in \mathbb{C}^{n_y \times n_y}$, $B_j \in \mathbb{C}^{n_x \times n_x}$, $F \in \mathbb{C}^{n_y \times n_x}$, $B_y \in \mathbb{C}^{K_y \times n_y}$, $B_x \in \mathbb{C}^{K_y \times n_x}$, $H \in \mathbb{C}^{K_y \times n_x}$, and $G \in \mathbb{C}^{K_x \times n_y}$. Our approach is to use the linear constraints to remove degrees of freedom in X and thus obtain a generalized Sylvester matrix equation with a unique solution without constraints.

By assumption the prescribed linear constraints are linearly independent so the column ranks of B_x and B_y are K_x and K_y, respectively. Without loss of generality, we further assume that the principal $K_x \times K_x$ and $K_y \times K_y$ submatrices of B_x and B_y are the identity matrices[4] I_{K_x} and I_{K_y}. Then, we can modify the matrix equation in (6.12) to

$$\sum_{j=1}^{k} A_j X B_j^T - \sum_{j=1}^{k} (A_j)_{1:n_y,1:K_y} B_y X B_j^T = F - \sum_{j=1}^{k} (A_j)_{1:n_y,1:K_y} H B_j^T,$$

[4]Otherwise, permute the columns of B_x and B_y so the principal $K_x \times K_x$ and $K_y \times K_y$ matrices $\hat{B}_x$ and $\hat{B}_y$ are invertible and redefine as $B_x = \hat{B}_x^{-1} B_x$, $G = \hat{B}_x^{-1} G$, $B_y = \hat{B}_y^{-1} B_y$, and $H = \hat{B}_y^{-1} H$.

where we have used the constraint $B_y X = H$. Moreover, by rearranging we have

$$\sum_{j=1}^{k} A_j X B_j^T - \sum_{j=1}^{k} (A_j)_{1:n_y,1:K_y} B_y X B_j^T = \sum_{j=1}^{k} \left(A_j - (A_j)_{1:n_y,1:K_y} B_y\right) X B_j^T,$$

and since the $K_y \times K_y$ principal matrix of B_y is the identity matrix, each matrix $A_j - (A_j)_{1:n_y,1:K_y} B_y$ for $1 \leq j \leq k$ is zero in the first K_y columns. Similarly, the condition $X B_x^T = G^T$ can be used to further modify the matrix equation as follows:

$$\begin{aligned} &\sum_{j=1}^{k} \left(A_j - (A_j)_{1:n_y,1:K_y} B_y\right) X \left(B_j - B_x (B_j)_{1:n_x,1:K_x}\right)^T \\ &\quad = F - \sum_{j=1}^{k} (A_j)_{1:n_y,1:K_y} H B_j^T - \sum_{j=1}^{k} \left(A_j - (A_j)_{1:n_y,1:K_y} B_y\right) G^T (B_j)_{1:n_x,1:K_x}^T, \end{aligned} \tag{6.13}$$

so that the matrices $(B_j - B_x (B_j)_{1:n_x,1:K_x})^T$ for $1 \leq j \leq k$ are zero in the first K_x rows.

Now, the first K_y columns of $A_j - (A_j)_{1:n_y,1:K_y} B_y$ and the first K_x rows of $(B_j - B_x (B_j)_{1:n_x,1:K_x})^T$ are zero in (6.13) and hence, the matrix equation is independent of the first K_y rows and K_x columns of X. Therefore, the matrix equation in (6.13) can be reduced by removing these zero columns and rows and then solved for a matrix $X_{22} \in \mathbb{C}^{(n_y - K_y) \times (n_x - K_x)}$, where

$$X = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix}, \qquad X_{11} \in \mathbb{C}^{K_y \times K_x}, \quad X_{12} \in \mathbb{C}^{K_y \times (n_x - K_x)}, \quad X_{21} \in \mathbb{C}^{(n_y - K_y) \times K_x}.$$

The solution of the resulting generalized Sylvester equation is given in Section 6.5.1.

Once we have computed X_{22} we can recover the remaining parts of X using the linear constraints. For instance, since $B_y X = H$ and the $K_y \times K_y$ principal submatrix of B_y is the identity matrix, we have

$$X_{12} = H_2 - B_y^{(2)} X_{22},$$

where $H = [H_1, H_2]$ with $H_1 \in \mathbb{C}^{K_y \times K_x}$ and $H_2 \in \mathbb{C}^{K_y \times (n_x - K_x)}$, and $B_y = [I_{K_y}, B_y^{(2)}]$ with $B_y^{(2)} \in \mathbb{C}^{K_y \times (n_y - K_y)}$. Furthermore, since $X B_x^T = G^T$ and the $K_x \times K_x$ principal

submatrix of B_x is the identity matrix, we have

$$X_{21} = G_2^T - X_{22}(B_x^{(2)})^T,$$

where $G = [G_1, G_2]$ with $G_1 \in \mathbb{C}^{K_x \times K_y}$ and $G_2 \in \mathbb{C}^{K_x \times (n_y - K_y)}$, and $B_x = [I_{K_x}, B_x^{(2)}]$ with $B_x^{(2)} \in \mathbb{C}^{K_x \times (n_x - K_x)}$. Lastly, we can recover X_{11} using either of the two formulas

$$X_{11} = H_1 - B_y^{(2)} X_{21}, \qquad X_{11} = G_1^T - X_{12}(B_x^{(2)})^T,$$

since the compatibility condition (6.11) ensures that both these formulas are equivalent.

6.5.1 Solving the matrix equation

We are left with a standard generalized Sylvester matrix equation of the form

$$\sum_{j=1}^{k} \tilde{A}_j X_{22} \tilde{B}_j^T = \tilde{F}, \tag{6.14}$$

and the exact algorithm we use to solve for X_{22} depends on the integer k.

If $k = 1$, the matrix equation takes the form $\tilde{A}_1 X_{22} \tilde{B}_1^T = \tilde{F}$, and since we are using the ultraspherical spectral method (see Section 6.3) the matrices $\tilde{A}_1$ and $\tilde{B}_1$ are almost banded. Therefore, we can solve $\tilde{A}_1 Y = \tilde{F}$ for $Y \in \mathbb{C}^{(n_y - K_y) \times (n_x - K_x)}$ in $\mathcal{O}(n_x n_y)$ operations and then solve $\tilde{B}_1 X_{22}^T = Y^T$ for X_{22} in $\mathcal{O}(n_x n_y)$ operations using the QRP^* factorization described in Section 6.3.2.

If $k = 2$, the matrix equation takes the form

$$\tilde{A}_1 X_{22} \tilde{B}_1^T + \tilde{A}_2 X_{22} \tilde{B}_2^T = \tilde{F}. \tag{6.15}$$

To solve (6.15) we use the generalized Bartels–Stewart algorithm [7, 55], which requires $\mathcal{O}(n_x^3 + n_y^3)$ operations. Alternatively, the generalized Hessenberg–Schur algorithm can be used [55]. It turns out that many standard PDOs with constant coefficients are of rank 2 (see Table 6.1 for a list of examples).

For $k \geq 3$, we are not aware of an efficient algorithm for solving (6.14). Instead,

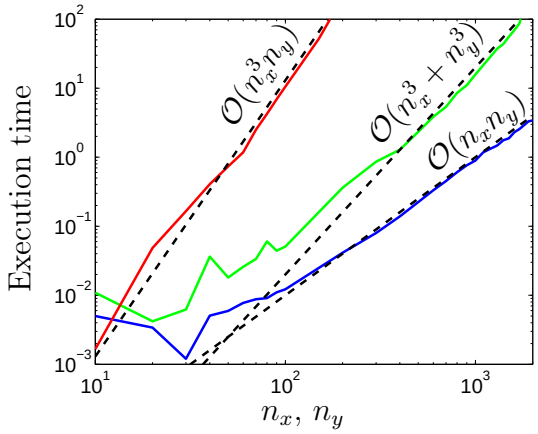


Figure 6.4: Typical computation cost for solving the matrix equations $\tilde{A}_1 X_{22} \tilde{B}_1^T = \tilde{F}$, (6.15), and (6.16). For rank 1 PDOs the complexity of the solver is $\mathcal{O}(n_x n_y)$, for rank 2 it is $\mathcal{O}(n_x^3 + n_y^3)$, and for rank 3 it is $\mathcal{O}(n_x^3 n_y)$.

we expand the matrix equation into an $n_x n_y \times n_x n_y$ linear system

$$\left(\sum_{j=1}^{k} (\tilde{B}_j \otimes \tilde{A}_j) \right) \mathrm{vec}(X_{22}) = \mathrm{vec}(\tilde{F}), \tag{6.16}$$

where '$\otimes$' denotes the Kronecker product operator for matrices and $\mathrm{vec}(C)$ denotes the vectorization of the matrix C formed by stacking the columns of C into a single column vector.

Naïvely solving the resulting linear system (6.16) requires $\mathcal{O}((n_x n_y)^3)$ operations. However, the matrices $\tilde{A}_j$ and $\tilde{B}_j$ are almost banded and hence the matrix $\sum_{j=1}^{k} (\tilde{B}_j \otimes \tilde{A}_j)$ is also almost banded with bandwidth $\mathcal{O}(n_x)$ except for the first $\mathcal{O}(n_x)$ rows that are dense. Thus, the linear system can be solved in $\mathcal{O}(n_x^2 (n_x n_y)) = \mathcal{O}(n_x^3 n_y)$ operations using the QRP^* factorization described in Section 6.3.2.

Figure 6.4 shows the computational time for solving $\tilde{A}_1 X_{22} \tilde{B}_1^T = \tilde{F}$, (6.15), and (6.16), where the matrices are almost banded with bandwidth of 10. This shows the typical dominating computational cost of the solver for rank 1, rank 2, and higher rank PDOs. In particular, it shows the substantial efficiency gain that can be achieved when the rank 1 or rank 2 structure of a PDO is exploited.

6.5.2 Solving subproblems

In some special cases the computational cost of solving the matrix equation can be reduced if the even and odd modes decouple and can be solved for separately.

For example, Laplace's equation with Dirichlet conditions can be split into four subproblems since the PDO contains only even derivatives in x and y and the boundary conditions can be equivalently written as the following operators:

$$\mathcal{B}_x = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & \cdots \\ 0 & 1 & 0 & 1 & 0 & \cdots \end{pmatrix}, \qquad \mathcal{B}_y = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & \cdots \\ 0 & 1 & 0 & 1 & 0 & \cdots \end{pmatrix}.$$

In this case since Laplace's equation is of rank 2 the computational time is reduced by a factor of 8.

In fact, any PDO with constant coefficients that contains only even order derivatives in one variable accompanied with Dirichlet or Neumann boundary conditions decouples into two subproblems. Moreover, if it contains only even (or odd) order derivatives in both variables then it decouples into four subproblems. Our implementation automatically detects these cases and splits the problem into two or four subproblems when possible.

6.6 Numerical examples

In this section we demonstrate the effectiveness of the algorithm on several examples.[5] We note that since our algorithm is based on a dense and direct solver, it does not matter to the stability of the approach if the PDE is elliptic, hyperbolic, or parabolic.

First, we consider Helmholtz's equation $u_{xx}+u_{yy}+K^2u = 0$ with Dirichlet boundary conditions on $[-1,1]^2$, where K is the wavenumber. Here, we take

$$u_{xx} + u_{yy} + (\sqrt{2}\omega)^2 u = 0, \qquad u(\pm 1, y) = f(\pm 1, y), \quad u(x, \pm 1) = f(x, \pm 1), \quad (6.17)$$

where $\omega \in \mathbb{R}$ and $f(x,y) = \cos(\omega x)\cos(\omega y)$. The exact solution to (6.17) is $u = \cos(\omega x)\cos(\omega y)$, which is highly oscillatory for large ω. In Figure 6.5 we plot the solution for $\omega = 50$ and plot the Cauchy error for $\omega = 10\pi, 50\pi, 100\pi$. The Cauchy error shows that the solution is rapidly resolved once we have computed an $\lceil\omega\rceil \times \lceil\omega\rceil$ matrix of Chebyshev coefficients (in agreement with the Shannon–Nyquist sampling rate [132]).

[5]The PDE solver described in this chapter has been implemented in MATLAB. It is publicly available as part of Chebfun via the `chebop2` command.

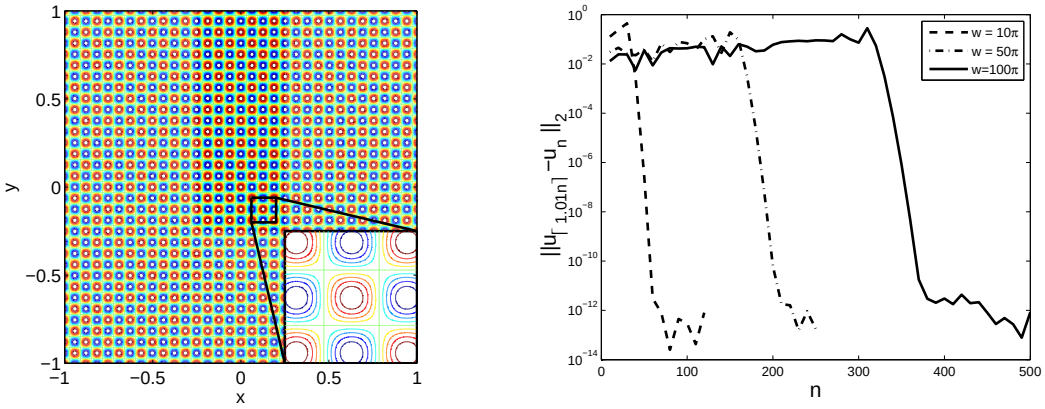


Figure 6.5: Left: Contour plot of the solution of (6.17) for $\omega = 50$. Right: Cauchy error for the solution coefficients for $\omega = 10\pi$ (dashed), $\omega = 50\pi$ (dot-dashed) and $\omega = 100\pi$ (solid), which shows the 2-norm difference between the coefficients of the approximate solution when solving an $n \times n$ matrix equation and an $\lceil 1.01n \rceil \times \lceil 1.01n \rceil$ matrix equation.

In particular, for $\omega = 100\pi$ in (6.17) we have

$$\left(\int_{-1}^{1} \int_{-1}^{1} \left(\tilde{u}(x,y) - u(x,y) \right)^2 dxdy \right)^{\frac{1}{2}} = 5.44 \times 10^{-10},$$

where $\tilde{u}$ is the computed solution and u is the exact solution. This error is relatively small considering that the solution has about 20,000 local extrema in the domain $[-1,1]^2$. The solution $\tilde{u}$ was computed in[6] 7.58 seconds. The implementation automatically detected and set up subproblems, which reduced the computational time by a factor close to 8.

As a second example, we compare the evolution of a Gaussian bump e^{-10x^2} when its propagation is governed by the wave equation and Klein–Gordon equation

$$u_{tt} = u_{xx}, \qquad u_{tt} = u_{xx} - 5u,$$

respectively. A similar comparison is done in [158]. In order to work on a bounded rectangular domain we restrict the x-variable to $[-15, 15]$ and Figure 6.6 shows the propagation of e^{-10x^2} for both equations for $0 \le t \le 10$. In these equations we impose zero Dirichlet conditions along $x = \pm 15$ with an initial pulse of e^{-10x^2} at $t = 0$. To

[6]Experiments were performed on a 2012 1.8GHz Intel Core i7 MacBook Air with MATLAB 2012a.

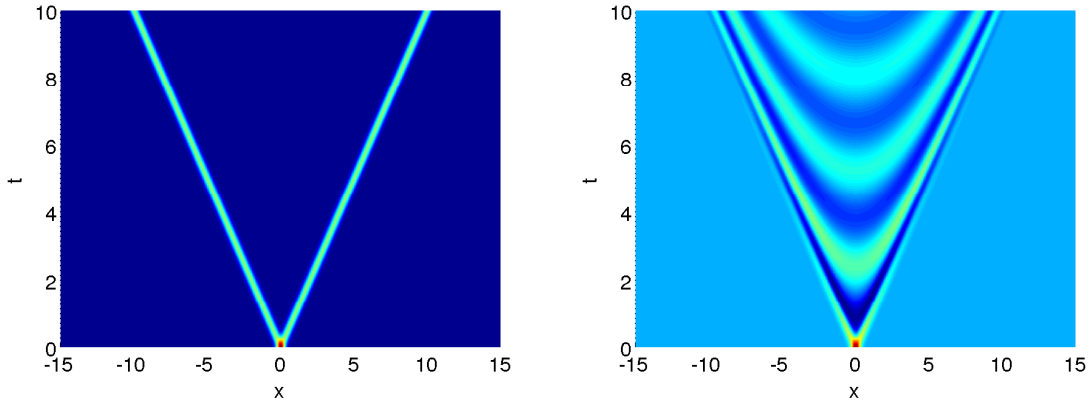


Figure 6.6: Comparison of the evolution of the Gaussian e^{-10x^2} on $[-15, 15]$ when propagation is governed by the wave equation (left) and the Klein–Gordon equation (right).

investigate the error for the solution of the wave equation we check if the wave's energy

$$E(t) = \frac{1}{2}\int_{-15}^{15} u_t^2 dx + \frac{1}{2}\int_{-15}^{15} u_x^2 dx$$

is conserved throughout $t \in [0, 10]$. (The algorithm does not impose that this quantity is conserved.) We find that $\|E - E(0)\|_\infty = 9.48 \times 10^{-10}$. In the wave equation $u_{tt} - u_{xx}$ all the modes travel at a speed of 1, while in the Klein–Gordon equation $u_{tt} - u_{xx} + 5u$ different frequencies travel at different speeds.

For a third example we take the time-dependent Schrödinger equation on $[0, 1] \times [0, 0.54]$,

$$\mathrm{i}\epsilon u_t = -\frac{1}{2}\epsilon^2 u_{xx} + V(x)u,$$

with $\epsilon = 0.0256$, $u(0, t) = 0$, $u(1, t) = 0$, and an initial condition $u(x, 0) = u_0(x)$, where

$$u_0(x) = e^{-25(x-1/2)^2} e^{-\mathrm{i}/(5\epsilon)\log(2\cosh(5(x-1/2)))}.$$

In Figure 6.7 (left) for $V(x) = 10$ and (right) for $V(x) = x^2$, we solve for the solution u and plot its real part. In both cases, we see the formation of caustic. In Figure 6.8, we plot $|u(x, 0.54)|^2$ for $V(x) = 10$ and two values of ϵ. The results are consistent with Fig. 2a/b of [5], which used periodic boundary conditions in place of Dirichlet.

The last example we consider is the biharmonic equation, a fourth order elliptic

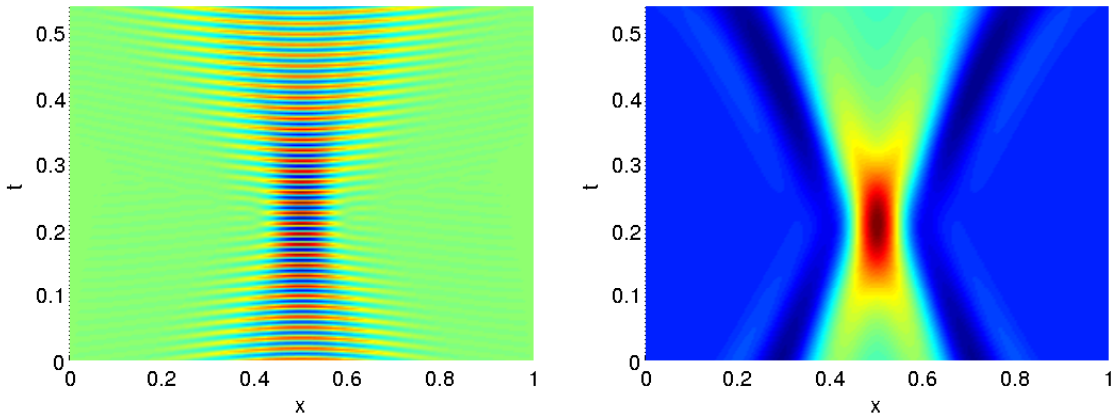


Figure 6.7: Real part of the solution to the time-dependent Schrödinger equation on $[0,1] \times [0, 0.54]$ with $\epsilon = 0.0256$ for $V(x) = 10$ (left) and $V(x) = x^2$ (right).

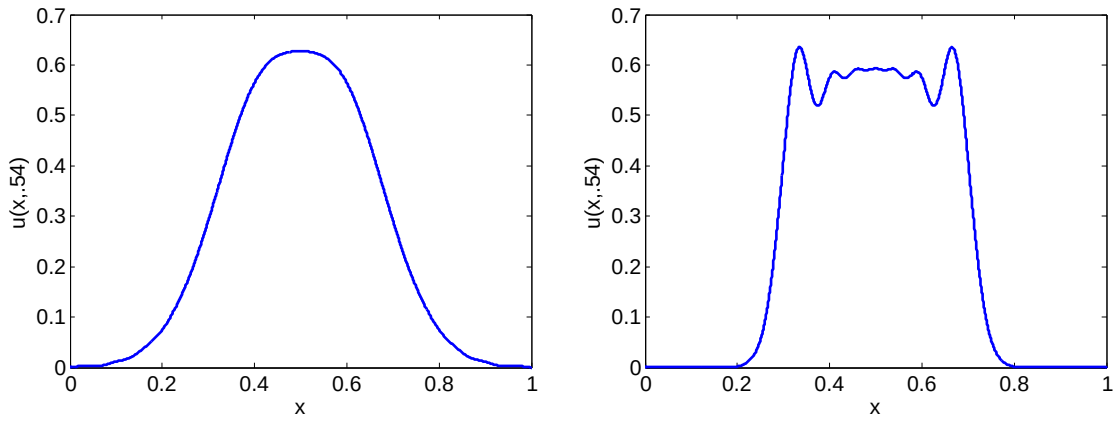


Figure 6.8: Solution to the Schrödinger equation with $V(x) = 10$ at $t = .54$ for $\epsilon = 0.0256$ (left) and $\epsilon = 0.0064$ (right).

PDE, given by

$$u_{xxxx} + u_{yyyy} + 2u_{xxyy} = 0, \qquad (x, y) \in [-1, 1]^2$$

with Dirichlet and Neumann boundary data corresponding to the function

$$v(x, y) = \operatorname{Im}\left((x - \mathrm{i}y)e^{-2(x+\mathrm{i}y)} + \cos(\cos(x + \mathrm{i}y))\right).$$

The exact solution is v itself. The implementation adaptively finds that a bivariate Chebyshev expansion of degree 30 in x and y is sufficient to resolve the solution to a maximum absolute error of 4.59×10^{-13}. This is a PDO of rank 3 and hence the algorithm solves a matrix linear system rather than a Sylvester matrix equation (see Section 6.5.1). Therefore, the computational time to solve this problem is 2.52 seconds, compared to a fraction of a second for a rank 2 PDO with the same discretization.

Conclusions

The original motivation for this thesis was to extend Chebfun to two dimensions, and at the start of 2012 such a project seemed daunting and quite frankly a little unrealistic. Yet, since then new mathematical ideas have come into fruition, many of them in this thesis, that have made this project a reality. Here, in 2014, we have a fully integrated 2D component of Chebfun — consisting of thousands of lines of MATLAB code — with dreams of conquering 3D next. This thesis has focused on the mathematical advances that have taken place to make all this possible.

One of the most important advances was how we turned the continuous idealization of Gaussian elimination for functions into a practical, efficient, and adaptive algorithm. This allowed us to represent scalar valued functions by low rank approximants, i.e., sums of functions of the form $g(y)h(x)$, where $g(y)$ and $h(x)$ are univariate functions. In doing so we were able to build on 1D Chebyshev technology in a powerful way and exploit decades of practical algorithmic experience in Chebfun. Chapter 2 discussed these advances and showed how they have been extended to vector calculus operations on vector fields and parametric surfaces.

Later we wanted to better understand the class of functions that are amenable to low rank approximation. In Chapter 3 we first developed the results necessary to make such a discussion precise and defined the concepts of numerical degree, numerical rank, and functions that are numerically of low rank. Then, we proved theorems to show when best low rank approximants converge algebraically and geometrically, before providing further insight by investigating three examples. In the final part we discussed the singular value decomposition for functions and its connection to best low rank function approximation.

In the original Chebfun thesis, Battles briefly mentioned (in the third from last paragraph) a vision of linear algebra for "matrices" that are indexed by continuous variables. In Chapter 4 we realized this vision by defining "cmatrices" to be those objects. We went on to define and analyze a full set of standard cmatrix factorizations,

and we demonstrated the striking similarities and subtle differences to matrix algebra. New mathematical problems arise for cmatrix factorizations related to compactness and convergence of infinite series. In a systematic way we proved convergence of each factorization under certain assumptions on the cmatrices.

Initially, a significant problem in extending Chebfun to two dimensions was the lack of a robust global bivariate rootfinder. Many different bivariate rootfinders were proposed, but each one left us wishing for a more robust alternative. In Chapter 5 we described such a robust alternative based on a resultant method, Bézout resultant matrices, and domain subdivision. This algorithm not only meets our needs but fits into the theme of computing 2D operations with 1D technology as it is based on using resultant matrices for finding the common roots of two univariate polynomials. The final algorithm is able to solve bivariate rootfinding problems for polynomials of much higher degree than algorithms previously proposed. It is one of the landmarks in our 2D extension of Chebfun.

Our latest project has been to extend low rank ideas to the solution of PDEs defined on rectangular domains, and in Chapter 6 we showed that low rank representations for partial differential operators can be easily calculated. We went on to exploit these representations to develop a framework for solving PDEs. We coupled this with the well-conditioned 1D spectral method by Olver and the author [115] and were able to efficiently solve PDEs in an automated manner using generalized matrix equations with almost banded matrices.

Even in just two dimensions, many challenges remain. We have not discussed the approximation of piecewise smooth functions that have discontinuities along curves, or the approximation of functions on arbitrary domains in $\mathbb{R}^2$, or imagined how to compute with the resulting approximants. Such problems present tantalizing prospects for future work.

Appendix A

Explicit Chebyshev expansions

A Lipschitz continuous function $f : [-1, 1] \to \mathbb{C}$ has an absolutely and uniformly convergent Chebyshev expansion (see Section 1.2),

$$f(x) = \sum_{j=0}^{\infty} a_j T_j(x), \qquad x \in [-1, 1].$$

The expansion coefficients $\{a_j\}_{j\geq 0}$ can be numerically calculated from their integral definition (see (1.4)); however, in some special cases they can be derived in closed form. The results in this appendix are used in Section 3.7 to calculate the numerical degree of functions.

One special case is when a function f has a pole on the real axis outside of $[-1, 1]$.

Lemma A.1. *Let $a > 1$. Then*

$$\frac{1}{x-a} = \frac{-2}{\sqrt{a^2-1}} {\sum_{j=0}^{\infty}}' \frac{T_j(x)}{(a+\sqrt{a^2-1})^j},$$
$$\frac{1}{x+a} = \frac{2}{\sqrt{a^2-1}} {\sum_{j=0}^{\infty}}' \frac{(-1)^j T_j(x)}{(a+\sqrt{a^2-1})^j},$$

where the prime indicates that the first term is halved.

Proof. From Mason and Handscomb [104, (5.14)] we have for $a > 1$

$$\frac{1}{x-a} = \frac{-2}{\sqrt{a^2-1}} {\sum_{j=0}^{\infty}}' (a-\sqrt{a^2-1})^j T_j(x),$$

and the first expansion follows by noting that $(a-\sqrt{a^2-1}) = (a+\sqrt{a^2-1})^{-1}$. The Chebyshev expansion of $1/(x+a)$ can also be immediately derived since

$$\frac{1}{x+a} = \frac{-1}{(-x)-a} = \frac{2}{\sqrt{a^2-1}} \sum_{j=0}^{\infty}{}' \frac{T_j(-x)}{(a+\sqrt{a^2-1})^j}$$

and $T_j(-x) = (-1)^j T_j(x)$. □

More generally, explicit Chebyshev expansions are known for certain meromorphic functions [48].

Another special case is when the function is a Gaussian.

Lemma A.2. *The Chebyshev series expansion of $e^{-\gamma x^2}$ is*

$$e^{-\gamma x^2} = \sum_{j=0}^{\infty}{}' (-1)^j \frac{2I_j(\gamma/2)}{e^{\gamma/2}} T_{2j}(x),$$

where $I_j(z)$ is the modified Bessel function of the first kind with parameter j and the prime indicates that the first term is halved.

Proof. See equation (14) on page 32 of [97]. □

Finally, the expansion coefficients for $e^{\mathrm{i}M\pi xy}$ can be derived.

Lemma A.3. *The bivariate Chebyshev expansion of $e^{\mathrm{i}M\pi xy}$ is*

$$e^{\mathrm{i}M\pi xy} = \sum_{p=0}^{\infty}{}' \sum_{q=0}^{\infty}{}' a_{pq} T_p(y) T_q(y),$$

$$a_{pq} = \begin{cases} 4\mathrm{i}^q J_{(q+p)/2}(M\pi/2) J_{(q-p)/2}(M\pi/2), & \mathrm{mod}(|p-q|,2) = 0, \\ 0, & \textit{otherwise}, \end{cases}$$

where the primes indicate that the $p=0$ and $q=0$ terms are halved. In particular, the $p=q=0$ term should be divided by four.

Proof. The bivariate Chebyshev expansion of $e^{\mathrm{i}M\pi xy}$ is given by

$$e^{\mathrm{i}M\pi xy} = \sum_{p=0}^{\infty}{}' \sum_{q=0}^{\infty}{}' a_{pq} T_p(y) T_q(y), \qquad a_{pq} = \frac{4}{\pi^2} \int_{-1}^{1} \int_{-1}^{1} \frac{e^{\mathrm{i}M\pi xy} T_p(y) T_q(x)}{\sqrt{1-y^2}\sqrt{1-x^2}} dx dy,$$

where the primes indicate that the $p = 0$ and $q = 0$ terms are halved. By the change of variables $x = \cos\theta$ and [114, (10.9.2)] we have

$$\int_{-1}^{1} \frac{e^{\mathrm{i}M\pi xy} T_q(x)}{\sqrt{1-x^2}} dx = \int_0^{\pi} e^{\mathrm{i}M\pi y\cos\theta} \cos(q\theta) d\theta = \pi \mathrm{i}^q J_q(M\pi y).$$

Therefore, by the change of variables $y = \cos\theta$ we have

$$a_{pq} = \frac{4\mathrm{i}^q}{\pi} \int_{-1}^{1} \frac{T_p(y) J_q(M\pi y)}{\sqrt{1-y^2}} dy = \frac{4\mathrm{i}^q}{\pi} \int_0^{\pi} J_q(M\pi\cos\theta)\cos(p\theta) d\theta.$$

If p and q are either both even or odd, then the integrand is an even function about $\pi/2$ and we have

$$a_{pq} = \frac{8\mathrm{i}^q}{\pi} \int_0^{\pi/2} J_q(M\pi\cos\theta)\cos(p\theta) d\theta = 4\mathrm{i}^q J_{(q+p)/2}(M\pi/2) J_{(q-p)/2}(M\pi/2),$$

where the last equality is from [114, (10.22.13)]. The result follows since if p and q are of different parity then the integrand is odd and hence, $a_{pq} = 0$. □

Appendix B

The Gagliardo–Nirenberg interpolation inequality

The Gagliardo–Nirenberg interpolation inequality is a result usually stated for multivariate functions defined on Lipschitz smooth domains [95, Thm. 1.4.5]. We used this result in Section 4.9 to obtain estimates of a function in the L^∞-norm from those in the L^2-norm. In one dimension the inequality becomes easier to prove and the constants can be calculated explicitly. First, in the 1D setting we derive an inequality that is slightly stronger than the more general Gagliardo–Nirenberg result.

Lemma B.1. *Let $f : [a, b] \to \mathbb{C}$ be a Lipschitz continuous function with Lipschitz constant $C < \infty$. Then, for any $1 \leq p < \infty$ we have*

$$\|f\|_{L^\infty([a,b])} \leq \max\left(2C^{1/(p+1)}\|f\|_{L^p}^{p/(p+1)}, \frac{2}{(b-a)^{1/p}}\|f\|_{L^p}\right).$$

Proof. Since f is continuous, it attains its supremum. Pick $x_0 \in [a, b]$ such that $|f(x_0)| = \|f\|_{L^\infty([a,b])}$. Then, for any $x \in I \cap [a, b]$ with

$$I = [x_0 - |f(x_0)|/(2C), x_0 + |f(x_0)|/(2C)]$$

we have, by the reverse triangle inequality,

$$|f(x)| \geq |f(x_0)| - C\,|x_0 - x| \geq \frac{1}{2}\,|f(x_0)|\,.$$

Thus, by basic estimates, for $1 \leq p < \infty$

$$\begin{aligned}\int_a^b |f(x)|^p dx &\geq \int_{\max(x_0-|f(x_0)|/(2C),a)}^{\min(x_0+|f(x_0)|/(2C),b)} |f(x)|^p dx \\ &\geq \left(\frac{1}{2}|f(x_0)|\right)^p \min(|f(x_0)|/(2C), b-a).\end{aligned}$$

The result follows by rearranging this inequality. □

For a continuously differentiable function, f, the Lipschitz constant is bounded by $\|f'\|_\infty$ and hence

$$\begin{aligned}\|f\|_{L^\infty([a,b])} &\leq \max\left(2\|f'\|_{L^\infty}^{1/(p+1)}\|f\|_{L^p}^{p/(p+1)}, \frac{2}{(b-a)^{1/p}}\|f\|_{L^p}\right) \\ &\leq 2\|f'\|_{L^\infty}^{1/(p+1)}\|f\|_{L^p}^{p/(p+1)} + \frac{2}{(b-a)^{1/p}}\|f\|_{L^p},\end{aligned}$$

where the last inequality holds because both arguments in the maximum are nonnegative. This is a special case of the Gagliardo–Nirenberg interpolation inequality.

Appendix C

The construction of Chebyshev Bézout resultant matrices*

Let p and q be two Chebyshev series of degree m and n (with real or complex coefficients) given by

$$p(x) = \sum_{i=0}^{m} \alpha_i T_i(x), \qquad q(x) = \sum_{i=0}^{n} \beta_i T_i(x), \qquad x \in [-1, 1].$$

The Chebyshev Bézout matrix of order $\max(m, n)$ associated with p and q is defined as $B = (b_{ij})_{1 \le i,j \le \max(m,n)}$, where the matrix entries satisfy

$$\frac{p(s)q(t) - p(t)q(s)}{s - t} = \sum_{i,j=1}^{\max(m,n)} b_{ij} T_{i-1}(t) T_{j-1}(s).$$

By multiplying both sides by $s - t$ we obtain an equivalent relation

$$s \left(\sum_{i,j=1}^{\max(m,n)} b_{ij} T_{i-1}(t) T_{j-1}(s) \right) - t \left(\sum_{i,j=1}^{\max(m,n)} b_{ij} T_{i-1}(t) T_{j-1}(s) \right) = p(s)q(t) - p(t)q(s). \tag{C.1}$$

The equation in (C.1) involves polynomials expressed in the Chebyshev basis and hence, can be written as a relation between Chebyshev coefficients

$$MB - BM^T = \mathbf{p}\mathbf{q}^T - \mathbf{q}\mathbf{p}^T, \qquad M = \begin{pmatrix} 0 & 1 & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{2} & 0 & \frac{1}{2} \\ & & & \frac{1}{2} & 0 \end{pmatrix}. \tag{C.2}$$

*This appendix is adapted from a paper with Vanni Noferini and Yuji Nakatsukasa [149]. I derived the algorithm (for the construction of linearizations in the *Double Ansatz Space*, see [149]) and Noferini pointed out that it could also be used for the construction of Bézout resultant matrices.

where **p** and **q** are the vectors of the first $\max(m,n)+1$ Chebyshev coefficients for p and q, respectively. (Note that M represents multiplication, when applied on the left it represents multiplication by 't' and when its transpose is applied on the right it represents multiplication by 's'.)

The Bézout matrix can then be constructed in $\mathcal{O}(\max(m,n)^2)$ operations by solving (C.2) using the Bartels–Stewart algorithm [7]. Note that as M is upper-Hessenberg, the matrix equation is already in reduced form. This results in the following MATLAB code:

```
function B = ChebyshevBezoutResultant(p, q)
% Construct the Chebyshev Bezout resultant matrix associated to two
% polynomials.
%   B = CHEBYSHEVBEZOUTRESULTANT(P, Q) returns the Chebyshev Bezout
%   resultant matrix associated to the Chebyshev series P and Q.
%   P and Q are column vectors of the Chebyshev coefficients.

m = length(p); n = length(q); N = max(m,n) - 1;
p(m+1:N+1) = 0; q(n+1:N+1) = 0;% Pad p and q to the same degree
B = zeros(N);
E = p*q.' - q*p.';              % B satisfies M*B-B*M' = p*q.'-q*p.'
B(N,:) = 2*E(N+1,1:N);
B(N-1,:) = 2*E(N,1:N) + [0 2*B(N,1) B(N,2:N-1)] + [B(N,2:N) 0];
for i = N-2:-1:1                % back subs for M*B-B*M' = p*q.'-q*p.'
    B(i,:) = 2*E(i+1,1:N)-B(i+2,1:N) + ...
                          [0 2*B(i+1,1) B(i+1,2:N-1)]+[B(i+1,2:N) 0];
end
B(1,:) = B(1,:)/2;              % T_0 coefficients are halved
```

This algorithm can be generalized to efficiently construct Bézout matrices in other polynomial degree-graded bases[1] including the monomial basis.

[1] A polynomial basis $\phi_0, \phi_1, \ldots,$ is degree-graded if ϕ_k is of exact degree k for $k \geq 0$.

Bibliography

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 10th edition, Dover Publications, New York, 1972.

[2] J. Asakura, T. Sakurai, H. Tadano, T. Ikegami, and K. Kimura, *A numerical method for nonlinear eigenvalue problems using contour integrals*, JSIAM Letters, 1 (2009), pp. 52–55.

[3] F. V. Atkinson, Multiparameter Eigenvalue Problems, Academic Press, New York, 1972.

[4] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[5] W. Bao, S. Jin, and P. A. Markowich, *On time-splitting spectral approximations for the Schrödinger equation in the semiclassical regime*, J. Comp. Phys., 175 (2002), pp. 487–524.

[6] S. Barnett, *Greatest common divisors from generalized Sylvester resultant matrices*, Linear Multilinear Algebra, 8 (1980), pp. 271–279.

[7] R. H. Bartels and G. W. Stewart, *Solution of the matrix equation $AX + XB = C$*, Comm. ACM, 15 (1972), pp. 820–826.

[8] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler, *Numerically Solving Polynomial Systems with Bertini*, SIAM, Philadelphia, 2013.

[9] Z. Battles, *Numerical Linear Algebra for Continuous Functions*, DPhil thesis, University of Oxford, 2005.

[10] Z. Battles and L. N. Trefethen, *An extension of MATLAB to continuous functions and operators*, SIAM J. Sci. Comput., 25 (2004), pp. 1743–1770.

[11] M. Bebendorf, *Approximation of boundary element matrices*, Numer. Math., 86 (2000), pp. 565–589.

[12] M. Bebendorf, *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, Springer-Verlag Berlin Heidelberg, 2008.

[13] M. Bebendorf, *Adaptive cross approximation of multivariate functions*, Constr. Approx., 34 (2011), pp. 149–179.

[14] M. Bebendorf and S. Kunis, *Recompression techniques for adaptive cross approximation*, J. Integral Equations Appl., 21 (2009), pp. 329–470.

[15] M. Bebendorf, Y. Maday, and B. Stamm, *Comparison of some reduced representation approximations*, submitted, 2013.

[16] B. Beckermann, *The condition number of real Vandermonde, Krylov and positive definite Hankel matrices*, Numer. Math., 85 (2000), pp. 553–577.

[17] E. T. Bell, *The history of Blissard's symbolic method, with a sketch of its inventor's life*, Amer. Math. Monthly, 45 (1938), pp. 414–421.

[18] W.-J. Beyn, *An integral method for solving nonlinear eigenvalue problems*, Linear Algebra Appl., 436 (2012), pp. 3839–3863.

[19] É. Bézout, *Théorie Générale des Équations Algébriques*, PhD Thesis, Pierres, Paris, 1779.

[20] D. A. Bini and L. Gemignani, *Bernstein–Bézoutian matrices*, Theoret. Comput. Sci., 315 (2004), pp. 319–333.

[21] D. Bini and A. Marco, *Computing curve intersection by means of simultaneous iterations*, Numer. Algorithms, 43 (2006), pp. 151–175.

[22] D. Bini and V. Noferini, *Solving polynomial eigenvalue problems by means of the Ehrlich–Aberth method*, Linear Algebra Appl., 439 (2013), pp. 1130–1149.

[23] A. Birkisson and T. A. Driscoll, *Automatic Fréchet differentiation for the numerical solution of boundary-value problems*, ACM Trans. Math. Software, 38 (2012), pp. 26:1–26:29.

[24] V. I. Bogachev, *Measure Theory, Volume 1*, Springer-Verlag Berlin Heidelberg, 2007.

[25] F. Bornemann, D. Laurie, S. Wagon and J. Waldvogel, *The SIAM* 100*-Digit Challenge: A Study in High-Accuracy Numerical Computing*, SIAM, Philadelphia, 2004.

[26] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, 2nd edition, Dover Publications, New York, (2001).

[27] J. P. Boyd, *Computing zeros on a real interval through Chebyshev expansion and polynomial rootfinding*, SIAM J. Numer. Anal., 40 (2002), pp. 1666–1682.

[28] J. P. Boyd, *Finding the zeros of a univariate equation: proxy rootfinders, Chebyshev interpolation, and the companion matrix*, SIAM Rev., 55 (2013), pp. 375–396.

[29] L. Brutman, *On the Lebesgue function for polynomial interpolation*, SIAM J. Numer. Anal., 15 (1978), pp. 694–704.

[30] B. Buchberger and F. Winkler, *Gröbner Basis and Applications*, Cambridge University Press, 1998.

[31] O. A. Carvajal, F. W. Chapman and K. O. Geddes, *Hybrid symbolic-numeric integration in multiple dimensions via tensor-product series*, Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation, Manuel Kauers (ed.), ACM Press, New York, 2005, pp. 84–91.

[32] A. Cayley, Note sur la méthode d'élimination de Bézout, J. Reine Angew. Math., 53 (1857), p. 366–367.

[33] S. Chandrasekaran and M. Gu, *Fast and stable algorithms for banded plus semiseparable systems of linear equations*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 373-384.

[34] S. Chandrasekaran and I. C. F. Ipsen, *Analysis of a QR algorithm for computing singular values*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 520–535.

[35] F. W. Chapman, *Generalized orthogonal series for natural tensor product interpolation*, PhD Thesis, University of Waterloo, 2003.

[36] H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM J. Sci. Comput., 26 (2005), pp. 1389–1404.

[37] C. W. Clenshaw, *A note on the summation of Chebyshev series*, Math. Comp., 9 (1955), pp. 118–120.

[38] C. W. Clenshaw and A. R. Curtis, *A method for numerical integration on an automatic computer*, Numer. Math., 2 (1960), pp. 197–205.

[39] J. A. Cochran, *The nuclearity of operators generated by Hölder continuous kernels*, Math. Proc. Cambridge Philos. Soc., 75 (1974), pp. 351–356.

[40] J. A. Cochran, *Growth estimates for the singular values of square-integrable kernels*, Pacific J. Math., 56 (1975), pp. 51–58.

[41] D. A. Cox, J. B. Little, D. O'Shea, *Ideals, Varieties, and Algorithms*, 3rd edition, Springer-Verlag, 2007.

[42] C. De Boor, *An alternative approach to (the teaching of) rank, basis, and dimension*, Linear Algebra Appl., 146 (1991), pp. 221–229.

[43] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[44] P. Dreesen, K. Batselier, and B. De Moor, *Back to the roots: Polynomial system solving, linear algebra, systems theory*, Proc. 16th IFAC Symposium on System Identification, pp. 1203–1208, 2012.

[45] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition*, SIAM J. Comput., 36 (2006), pp. 184–206.

[46] T. A. Driscoll, F. Bornemann, and L. N. Trefethen, *The chebop system for automatic solution of differential equations*, BIT Numerical Mathematics, 48 (2008), pp. 701–723.

[47] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[48] D. Elliott, *The evaluation and estimation of the coefficients in the Chebyshev series expansion of a function*, Math. Comp., 18 (1964), pp. 274–284.

[49] I. Z. Emiris and B. Mourrain, *Matrices in elimination theory*, J. Symbolic Comput., 28 (1999), pp. 3–43.

[50] J. C. Ferreira and V. A. Menegatto, *Eigenvalues of integral operators defined by smooth positive definite kernels*, Integral Equations Operator Theory, 64 (2009), pp. 61–81.

[51] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, 1998.

[52] L. Fox and I. B. Parker, *Chebyshev Polynomials in Numerical Analysis*, Oxford University Press, Oxford, 1968.

[53] R. Franke, *A critical comparison of some methods for interpolation of scattered data*, Naval Postgraduate School, Tech. Report, March 1979.

[54] D. Gaier, *Lectures on Complex Approximation*, Birkhäuser, Basel, 1987.

[55] J. D. Gardiner, A. J. Laub, J. J. Amato, C. B. Moler, *Solution of the Sylvester matrix equation* $AXB^T + CXD^T = E$, ACM Trans. Math. Software, 18 (1992), pp. 223–231.

[56] I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky, *Discriminants, Resultants, and Determinants*, Birkhäuser, Boston, 2008.

[57] W. M. Gentleman, *Implementing Clenshaw–Curtis quadrature II: Computing the cosine transformation*, Comm. ACM, 15 (1972), pp. 343–346.

[58] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, SIAM, Philadelphia, USA, 2009, (unabridged republication of book first published by Academic Press in 1982).

[59] G. H. Golub, *Numerical methods for solving least squares problems*, Numer. Math., 7 (1965), pp. 206–216.

[60] G. H. Golub and C. Reinsch, *Singular value decomposition and least squares solutions*, Numer. Math., 14 (1970), pp. 403–420.

[61] G. H. Golub and C. F. Van Loan, *Matrix Computation*, 4th edition, Johns Hopkins University Press, Maryland, 2013.

[62] P. Gonnet, S. Güttel, and L. N. Trefethen, *Robust Padé approximation via SVD*, SIAM Review, 55 (2013), pp. 101–117.

[63] I. J. Good, *The colleague matrix, a Chebyshev analogue of the companion matrix*, Q. J. Math., 12 (1961), pp. 61–68.

[64] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *How to find a good submatrix*, in Matrix Methods: Theory, Algorithms and Applications, V. Olshevsky and E. Tyrtyshnikov, eds., World Scientific, Hackensack, NJ, 2010, pp. 247–256.

[65] , S. A. Goreinov, E. E. Tyrtyshnikov, *The maximal-volume concept in approximation by low-rank matrices*, Contemporary Mathematics 280 (2001), pp. 47–52.

[66] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *A theory of pseudoskeleton approximations*, Linear Algebra Appl., 261 (1997), pp. 1–21.

[67] T. Gowers, J. Barrow-Green, and I. Leader, eds., *The Princeton Companion to Mathematics*, Princeton University Press, New Jersey, 2008.

[68] L. Grasedyck, *Singular value bounds for the Cauchy matrix and solutions of Sylvester equations*, Technical report 13, University of Kiel, 2001.

[69] M. Griebel and H. Harbrecht, *Approximation of bi-variable functions: singular value decomposition versus sparse grids*, IMA J. Numer. Anal., 34 (2014), pp. 28–54.

[70] B. Guo and I. Babuška, *The hp version of the finite element method*, Comput. Mech., 1 (1986), pp. 21–41.

[71] A. Haar, *Die Minkowskische Geometrie und die Annäherung an stetige Funktionen*, Mathematische Annalen, 78 (1917), pp. 294–311.

[72] W. Hackbusch, *Hierarchische Matrizen: Algorithmen und Analysis*, Springer-Verlag Berlin Heidelberg, 2009.

[73] W. Hackbusch, *Tensor Spaces and Numerical Tensor Calculus*, Springer-Verlag Berlin Heidelberg, 2012.

[74] N. Hale and A. Townsend, *Fast and accurate computation of Gauss–Legendre and Gauss–Jacobi quadrature nodes and weights*, SIAM J. Sci. Comput., 35 (2013), A652–A674.

[75] N. Hale and A. Townsend, *A fast, simple, and stable Chebyshev–Legendre transform using an asymptotic formula*, SIAM J. Sci. Comput., 36 (2014), pp. A148-A167.

[76] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[77] A. Hammerstein, *Über die Entwickelung des Kernes linearer Integralgleichungen nach Eigenfunktionen*, Sitzungsberichte Preuss. Akad. Wiss., (1923), pp. 181–184.

[78] H. Harbrecht, M. Peters, and R. Schneider, *On the low-rank approximation by the pivoted Cholesky decomposition*, Appl. Numer. Math., 62 (2012), pp. 428–440.

[79] A. Harnack, *Über Vieltheiligkeit der ebenen algebraischen Curven*, Math. Ann., 10 (1876), pp. 189–198.

[80] N. J. Higham, *A survey of condition number estimation for triangular matrices*, SIAM Rev., 29 (1987), pp. 575–596.

[81] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd edition, SIAM, Philadelphia, 2002.

[82] E. Hille and J. D. Tamarkin, *On the characteristic values of linear integral equations*, Acta Math., 57 (1931), pp. 1–76.

[83] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt, *Marching triangles: range image fusion for complex object modelling*, IEEE International Conference on Image Processing, pp. 381–384, 1996.

[84] M. E. Hochstenbach, T. Košir, and B. Plestenjak, A Jacobi–Davidson type method for the two-parameter eigenvalue problem, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 477–497.

[85] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1990.

[86] L. Hörmander, *The Analysis of Linear Partial Differential Operators I: Distribution Theory and Fourier Analysis*, Springer-Verlag, Berlin, 2003.

[87] D. Kapur and T. Saxena, *Comparison of various multivariate resultant formulations*, In Levelt, A., ed., Proc. Int. Symp. on Symbolic and Algebraic Computation, Montreal, pp. 187–194, 1995.

[88] M. Javed, *Numerical computation with delta functions in Chebfun*, MSc. Thesis, University of Oxford, 2011.

[89] G. Jónsson and S. Vavasis, *Accurate solution of polynomial equations using Macaulay resultant matrices*, Math. Comp., 74 (2005), pp. 221–262.

[90] O. D. Kellogg, *Orthogonal function sets arising from integral equations*, Amer. J. Math., 40 (1918), pp. 145–154.

[91] F. C. Kirwan, *Complex Algebraic Curves*, Cambridge University Press, Cambridge, 1992.

[92] G. Klein, *Applications of Linear Barycentric Rational Interpolation*, PhD thesis, University of Fribourg, 2012.

[93] S. Knapek, *Hyperbolic cross approximation of integral operators with smooth kernel*, Tech. Report 665, SFB 256, University of Bonn, 2000.

[94] G. Little and J. B. Reade, *Eigenvalues of analytic kernels*, SIAM J. Math. Anal., 15 (1984), pp. 133–136.

[95] Z. Liu and S. Zheng, *Semigroups Associated with Dissipative Systems*, CRC Press, USA, 1999.

[96] W. E. Lorensen and H. E. Cline, *Marching cubes: A high resolution 3D surface construction algorithm*, ACM Comput. Graph., 21 (1987), pp. 163–169.

[97] Y. L. Luke, *The special functions and their approximations*, Vol. II, Academic Press, New York, 1969.

[98] D. S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann, *Vector spaces of linearizations for matrix polynomials*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 971–1004.

[99] M. W. Mahoney and P. Drineas, *CUR matrix decompositions for improved data analysis*, Proc. Natl. Acad. Sci. USA, 106 (2009), pp. 697–702.

[100] J. C. Mairhuber, *On Haar's theorem concerning Chebychev approximation problems having unique solutions*, Proc. Amer. Math. Soc., 7 (1956), pp. 609–615.

[101] D. Manocha and J. Demmel, *Algorithms for intersecting parametric and algebraic curves I: simple intersections*, ACM Trans. Graphics, 13 (1994), pp. 73–100.

[102] P. G. Martinsson, V. Rokhlin, Y. Shkolnisky, and M. Tygert, *ID: a software package for low-rank approximation of matrices via interpolative decompositions*, version 0.2.

[103] P. G. Martinsson, V. Rokhlin, and M. Tygert, *A randomized algorithm for the decomposition of matrices*, Appl. Comput. Harmon. Anal., 30 (2011), pp. 47–68.

[104] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*, CRC Press, Florida, 2003.

[105] P. C. Matthews, *Vector Calculus (Springer Undergraduate Mathematics Series)*, Springer-Verlag London, 1998.

[106] V. Mehrmann and H. Voss, *Nonlinear Eigenvalue Problems: A Challenge for Modern Eigenvalue Methods*, Mitt. der Ges. für Angewandte Mathematik and Mechanik, 27 (2005), pp. 121–151.

[107] J. Mercer, *Functions of positive and negative type and their connection with the theory of integral equations*, Philos. Trans. R. Soc. Lond. Ser. A, 209 (1909), pp. 415–446.

[108] J. Milnor, *Analytic proofs of the "Hairy Ball theorem" and the Brouwer fixed point theorem*, Amer. Math. Monthly, 85 (1978), pp. 521–524.

[109] L. Mirsky, *Symmetric gauge functions and unitarily invariant norms*, Quart. J. Math., 11 (1960), pp. 50–59.

[110] B. Mourrain and J. P. Pavone, *Subdivision methods for solving polynomial equations*, J. Symbolic Comput., 44 (2009), pp. 292–306.

[111] A. Muhič and B. Plestenjak, *On the quadratic two-parameter eigenvalue problem and its linearization*, Linear Algebra Appl., 432 (2010), pp. 2529–2542.

[112] Y. Nakatsukasa, V. Noferini, and A. Townsend, *Computing common zeros of two bivariate functions*, MATLAB Central File Exchange, `http://www.mathworks.com/matlabcentral/fileexchange/44084`, 2013.

[113] Y. Nakatsukasa, V. Noferini, and A. Townsend, *Computing the common zeros of two bivariate functions via Bézout resultants*, to appear in Numer. Math., 2014.

[114] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, *NIST Handbook of Mathematical Functions*, Cambridge University Press, Cambridge, 2010.

[115] S. Olver and A. Townsend, *A fast and well-conditioned spectral method*, SIAM Rev., 55 (2013), pp. 462–489.

[116] E. L. Ortiz and H. Samara, *An operational approach to the Tau method for the numerical solution of non-linear differential equations*, Computing, 27 (1981), pp. 15–25.

[117] R. Pachón, R. B. Platte, and L. N. Trefethen, *Piecewise-smooth chebfuns*, IMA J. Numer. Anal., 30 (2010), pp. 898–916.

[118] R. Pachón and L. N. Trefethen, *Barycentric-Remez algorithms for best polynomial approximation in the chebfun system*, BIT Numer. Math., 49 (2009), pp. 721–741.

[119] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.

[120] R. B. Platte and L. N. Trefethen, *Chebfun: a new kind of numerical computing*, Progress in Industrial Mathematics at ECMI 2008, Springer-Verlag Berlin Heidelberg, 15 (2010), pp. 69–87.

[121] D. Plaumann, B. Sturmfels, and C. Vinzant, *Computing linear matrix representations of Helton–Vinnikov curves*, Mathematical Methods in Systems, Optimization, and Control Operator Theory, 222 (2012), pp. 259–277.

[122] M. J. D. Powell, *Approximation Theory and Methods*, Cambridge University Press, Cambridge, 1981.

[123] J. B. Reade, *Eigenvalues of positive definite kernels*, SIAM J. Math. Anal., 14 (1983), pp. 152–157.

[124] J. B. Reade, *Eigenvalues of positive definite kernels II*, SIAM J. Math. Anal., 15 (1984), pp. 137–142.

[125] J. R. Rice, *Tchebycheff approximation in several variables*, Trans. Amer. Math. Soc., 109 (1963), pp. 444–466.

[126] M. Richardson, *Approximating functions with endpoint singularities*, PhD thesis, University of Oxford, 2013.

[127] T. J. Rivlin and H. S. Shapiro, *Some uniqueness problems in approximation theory*, Comm. Pure Appl. Math., 13 (1960), pp. 35–47.

[128] T. J. Rivlin, *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory*, 2nd edition, J. Wiley, 1990.

[129] M. Sagraloff and others, *Gallery of algebraic curves and their arrangements*, `http://exacus.mpi-inf.mpg.de/gallery.html`.

[130] E. Schmidt, *Zur Theorie der linearen und nichtlinearen Integralgleichungen. I Teil. Entwicklung willkürlichen Funktionen nach Systemen vorgeschriebener*, Math. Ann., 63 (1907), pp. 433–476.

[131] J. Schneider, *Error estimates for two-dimensional cross approximation*, J. Approx. Theory, 162 (2010), pp. 1685–1700.

[132] C. E. Shannon, *Communication in the presence of noise*, Proc. IEEE, 86 (1998), pp. 447–457.

[133] J. Shen, *Efficient spectral-Galerkin method II. Direct solvers of second- and fourth-order equations using Chebyshev polynomials*, SIAM J. Sci. Comput., 16 (1995), pp. 74–87.

[134] E. C. Sherbrooke and N. M. Patrikalakis, *Computation of the solutions of nonlinear polynomial systems*, Comput. Aided Geom. Design, 10 (1993), pp. 379–405.

[135] F. Smithies, *The eigenvalues and singular values of integral equations*, Proc. Lond. Math. Soc., 2 (1938), pp. 255–279.

[136] S. A. Smolyak, *Quadrature and interpolation formulas for tensor products of certain classes of functions*, Dokl. Akad. Nauk SSSR, 4 (1963), pp. 240–243.

[137] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*, World Scientific, Singapore, 2005.

[138] L. Sorber, M. Van Barel, and L. De Lathauwer, *Numerical solution of bivariate and polyanalytic polynomial systems*, submitted, 2013.

[139] G. W. Stewart, *On the early history of the singular value decomposition*, SIAM Review, 35 (1993), pp. 551–566.

[140] G. W. Stewart, *Afternotes Goes to Graduate School*, SIAM, Philadelphia, 1998.

[141] G. W. Stewart, *Matrix Algorithms Volume 1: Basic Decompositions*, SIAM, Philadelphia, 1998.

[142] G. W. Stewart, *Fredholm, Hilbert, Schmidt: Three fundamental papers on integral equations*, `www.cs.umd.edu/~stewart/FHS.pdf`, 2011.

[143] M. H. Stone, *The generalized Weierstrass approximation theorem*, Math. Mag., 21 (1948), pp. 237–254.

[144] B. Sturmfels, *Solving Systems of Polynomial Equations*, American Mathematical Society, 2002.

[145] J.-G. Sun, *On condition numbers of nondefective multiple eigenvalue*, Numer. Math., 61 (1992), pp. 265–275.

[146] G. Szegő, *Orthogonal Polynomials*, Amer. Math. Soc., 1939.

[147] F. Tisseur, *Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1038–1057.

[148] A. Townsend, *Chebfun2 software*, `http://www.chebfun.org/chebfun2`, 2013.

[149] A. Townsend, V. Noferini, and Y. Nakatsukasa, *Vector spaces of linearizations for matrix polynomials: A bivariate polynomial approach*, submitted.

[150] A. Townsend and S. Olver, *The automatic solution of partial differential equations using a global spectral method*, unpublished manuscript, 2014.

[151] A. Townsend and L. N. Trefethen, *Gaussian elimination as an iterative algorithm*, SIAM News, 46, March 2013.

[152] A. Townsend and L. N. Trefethen, *An extension of Chebfun to two dimensions*, SIAM J. Sci. Comput., 35 (2013), pp. C495–C518.

[153] A. Townsend and L. N. Trefethen, *Continuous analogues of matrix factorizations*, submitted to SIAM Review, 2014.

[154] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000.

[155] L. N. Trefethen, *A Hundred-Dollar, Hundred-Digit Challenge*, SIAM News, 35, Jan./Feb. 2002.

[156] L. N. Trefethen, *Computing numerically with functions instead of numbers*, Math. Comput. Sci., 1 (2007), pp. 9–19.

[157] L. N. Trefethen, *Householder triangularization of a quasimatrix*, IMA J. Numer. Anal., 30 (2010), pp. 887–897.

[158] L. N. Trefethen and K. Embree, *The PDE Coffee Table Book*, unpublished book, `http://people.maths.ox.ac.uk/trefethen/pdectb.html`, 2011.

[159] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, Philadelphia, 2013.

[160] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[161] L. N. Trefethen et al., Chebfun Version 5, The Chebfun Development Team (2014), `http://www.chebfun.org/`.

[162] L. N. Trefethen and J. A. C. Weideman, *The exponentially convergent trapezoidal rule*, SIAM Rev., to appear in 2014.

[163] J. Van Deun and L. N. Trefethen, *A robust implementation of the Carathéodory–Fejér method for rational approximation*, BIT Numer. Math., 51 (2011), pp. 1039–1050.

[164] J. Waldvogel, *Fast construction of the Fejér and Clenshaw–Curtis quadrature rules*, BIT Numer. Math., 46 (2006), pp. 195–202.

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

[166] H. Weyl, *Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Andwendung auf die Theorie der Hohlraumstrahlung)*, Math. Ann., 71 (1912), pp. 441–479.

[167] H. Xie and H. Dai, *On the sensitivity of multiple eigenvalues of nonsymmetric matrix pencils*, Linear Algebra Appl., 374 (2003), pp. 143–158.